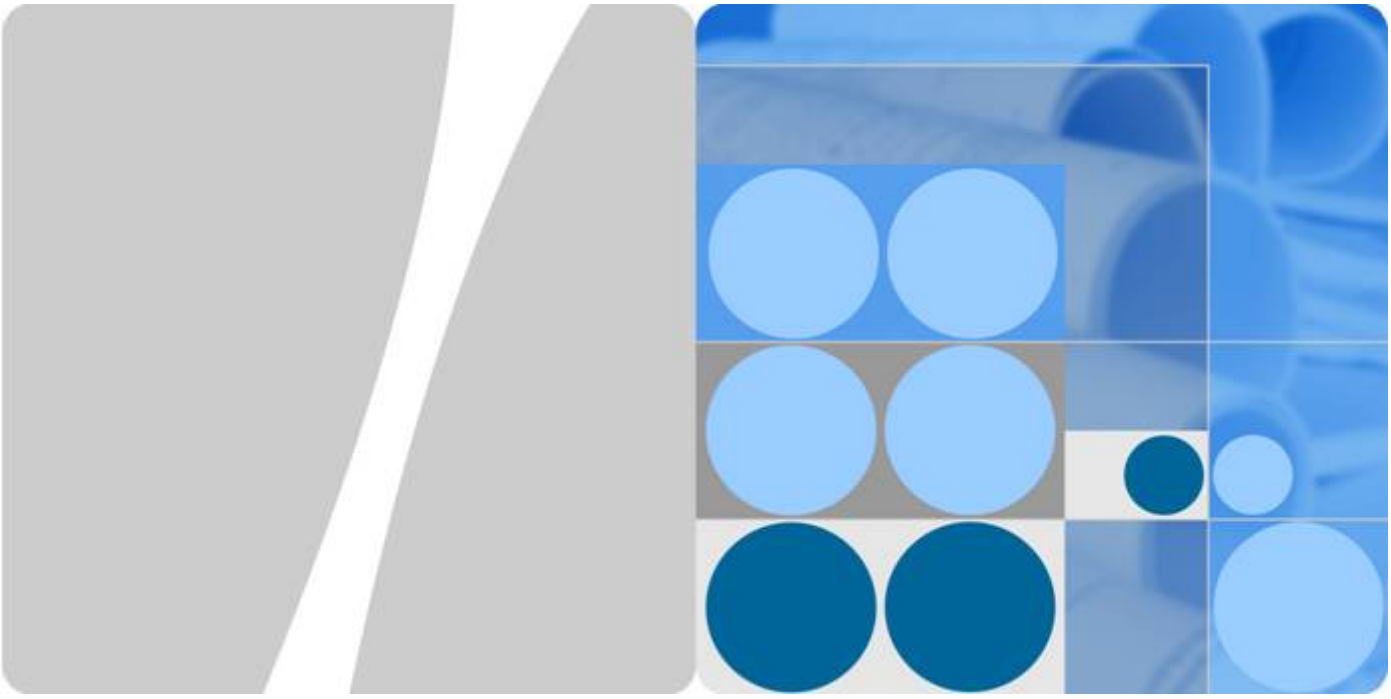


OceanConnect

V100R001C30

API 参考

文档版本　01

发布日期　2017-08-19

华为技术有限公司

HUAWEI

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

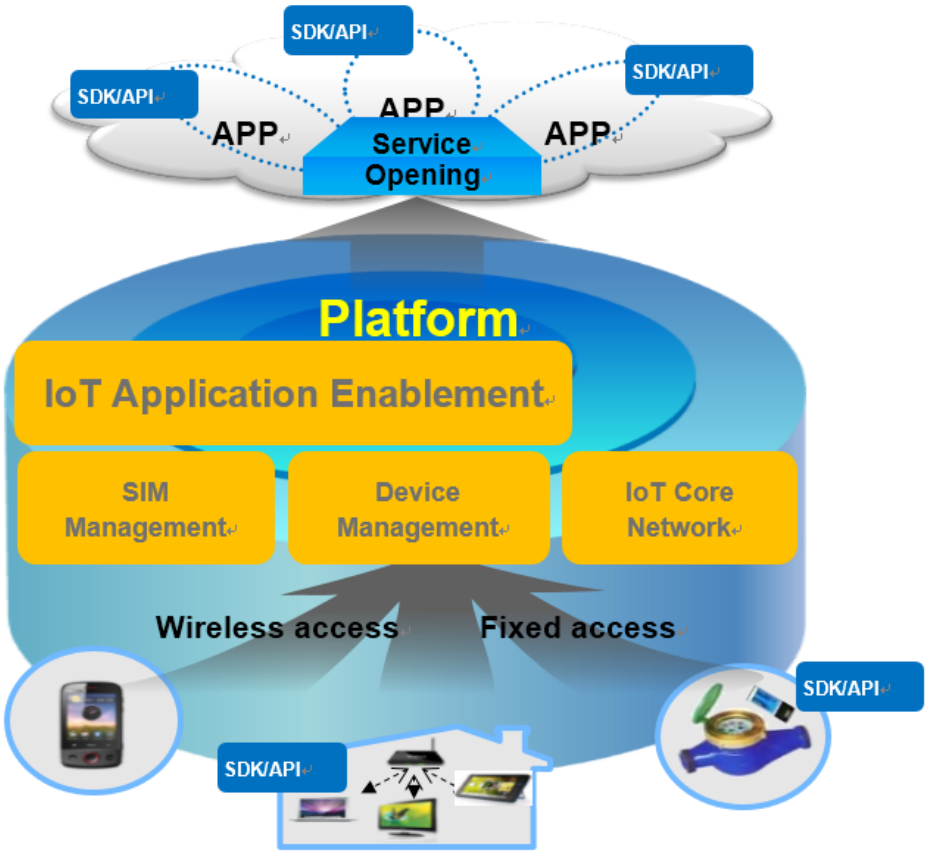
目 录

1 Agent API 参考（网关）

1.1 开发者必读

1. 概述

华为Agent作为华为IOT解决方案的一部分，主要实现了各种物联网通信协议接入、各种传感器类型的抽象、与平台之间的登录/认证/数据上报/命令下发、Bundle的部署和管理。



华为Agent会提供一个基础的能力包（中间件），并通过开放相关API，实现合作伙伴（第三方传感器，第三方物联网协议等）通过此Agent中间件快速接入到IOT解决方案中。第三方开发者只需要调用并实现相关接口，即可完成快速的集成和扩展。

华为Agent是基于OSGi框架开发，，目前已经支持Zwave、ZigBee协议传感器、Onvif摄像头的接入，并提供这些协议的应用层API接口，第三方通过这些API接口可以快速完成传感器的接入。

本文档主要介绍将IOT Agent部署到第三方网关的集成场景：

1. Agent集成到第三方网关的适配开发：

场景一：简易集成

Agent集成到第三方网关不需要支持下面功能时，建议采用简易集成方式，该方式只需要设置环境变量无需进行适配开发。

1. 不需要支持IPC
2. 不需要通过LED灯显示Agent状态
3. 不需要在用户APP上提示网关升级等硬件状态

场景二：深度集成

当Agent集成后需要支持上述任一种功能时，采用深度集成方式；需要根据**硬件定制扩展接口**开发一个OSGi Bundle，实现与硬件接口的适配。

2. 基本概念

- **OSGI**：Open Service Gateway Initiative，开放服务网关协议Java动态化模块化系统的一系列规范。
- **bundle**：OSGI框架中的应用插件。
- **Zwave协议**：一种新兴的基于射频的、低成本、低功耗、高可靠、适于网络的短距离无线通信技术。
- **ZigBee协议**：基于IEEE802.15.4标准的低功耗局域网协议。

3. 接口全景图

集成开发者通过全景图了解到：华为Agent提供了哪些功能？每个功能包含哪些接口？接口之间的逻辑关系如何？从而更快速的找到正确的接口来实现具体业务。

接口类别	功能	接口	说明
硬件定制扩展接口	硬件信息查询接口	`GatewayInfoQueryService`	查询网关升级状态、局域网MAC地址列表等信息
	硬件状态上报接口	`GatewayNotifyListener`	IOT Agent在固件升级、WIFI设备上线时会触发对应的业务流程，该接口用于将相关硬件状态通知到IOT Agent业务层
	WPS接口	`GatewayWpsService`	IOT Agent支持接入WPS设备，通过调用该接口实现的WPS硬件交互，当需要支持WPS设备时必须实现该接口
	控制LED灯接口	`LEDControlInterface`	在与云平台中断、bundle升级等场景中IOT Agent会调用LED灯接口，通过LED灯状态变化来提示当前业务状态

1.2 接口列表

1.2.1 配置项定制

业务配置

配置IOT Agent运行所需的环境变量，采用Key-Value形式。

配置文件名：IOTAgentConfig.properties

配置文件路径：在Agent运行目录下，如果运行目录不存在则取系统默认值。

参数名	必选/可选	描述
NetworkInterface	必选	IOT Agent在云平台注册时需要有一个唯一标识，当前以MAC地址作为唯一标识，需要配置对应的网卡名，默认值br0。
ZwavePort	可选	与ZWave硬件模块的通讯的串口号，支持ZWave设备是必须配置，默认值/dev/ttyAMA0。
ZigBeePort	可选	与ZigBee硬件模块的通讯的串口号，支持ZigBee设备是必须配置，默认值/dev/ttyAMA0。
ManuFacturer	可选	网关设备厂商的默认值，获取设备厂商定制接口未实现或获取失败时，取该默认值Huawei。
FirewareVersion	可选	网关固件版本的默认值，**硬件信息查询接口**未实现或获取失败时，取该配置项的值。
HardwareVersion	可选	网关硬件版本的默认值，**硬件信息查询接口**未实现或获取失败时，取该配置项的值。
HardwareModel	可选	网关硬件版本的默认值，**硬件信息查询接口**未实现或获取失败时，取该配置项的值。
supportFrequentErase	可选	标识网关是否支持频繁擦写Flash，默认值为True，低端Flash频繁擦写会显著降低寿命，建议设置为False。
ESHThingConf	可选	Eclipse Smarthome 开源框架things配置文件名，Agent支持自动加载Smarthome开源框架的things配置，文件格式内容与Smarthome开源框架该配置完全一致，默认值demo.things

1.2.2 硬件定制扩展接口

1.2.2.1 硬件信息查询接口

接口功能

用于Agent查询网关升级状态和查询LAN口客户端IP信息，比如在网关绑定等业务流程中会调用该接口查询网关的升级状态，该接口可选实现。

接口描述

接口名GatewayInfoQueryService，需要实现下面2个接口函数：

```
Map<String, String>queryLanInfo()
UpgradeStatus queryUpgradeStatus()
```

参数说明

字段/方法名	参数	返回值	描述
queryLanInfo	无	Map<String, String>	该接口返回当前接入该网关局域网的客户端地址列表，其中Key为ip地址，value为Mac地址；如key等于192.168.164.1，value等于025DC7169988
queryUpgradeStatus	无	UpgradeStatus	该接口返回查询网关升级状态，返回值为UpgradeStatus类型，如果正在升级返回Upgrading，已经升级完成或没有升级返回upgraded。

返回结果

无

注意事项

无

示例

下面示例中假设该网关上的硬件信息保存在系统文件中，硬件信息查询接口中从系统文件中读取配置并返回。

```
public class OntInfoQueryServiceImpl implements GatewayInfoQueryService
{
    //查询网关升级状态的硬件接口
    private ICommonService mCommonService;

    /*实现查询网关升级状态的接口 */
    public UpgradeStatus queryUpgradeStatus()
    {
        if (null == mCommonService)
        {
               return null;
        }
```

```
        int status = mCommonService.getDeviceUpdateState();
        switch (status)
        {
            case 1:
                return UpgradeStatus.upgrading;
            case 2:
                   return UpgradeStatus.upgraded;
            default:
                logger.error("Invaild DeviceUpdateState: {}", status);
                return null;
        }
}}
```

1.2.2.2 硬件状态上报接口

接口功能

当硬件状态变化时通过该接口通知IOT Agent业务模块，当前支持的硬件事件包括：固件升级、Bundle升级、WIFI设备上线；

IOT Agent中有多个业务模块都会监听该接口事件，在本接口的实现类中需要获得所有的监听者并向所有的监听者发送通知。

接口描述

接口类名GatewayNotifyListener，对应的接口函数如下：

```
public void notifyListener(EventTopic topic, Map<String, String> props)
```

在硬件状态变化时，调用该接口将事件发送给业务层。

参数说明

字段	类型	描述
Topic	EventTopic	表示发生的事件类型： 固件升级 TOPIC_GATEWAY_UPGRADE_STATUS 局域网设备上线 TOPIC_GATEWAY_WLAN_DEV_ONLINE Bundle升级 TOPIC_BUNDLE_UPGRADE_STATUS

字段	类型	描述
props	Map<String, String>	表示事件的详细信息，KEY-VALUE形式： 固件升级或Bundle升级事件，Key等于“status”，Value等于“upgrading”或“upgraded” 局域网设备上线时间，Key等于“MacAddr”，Value表示上线的设备MAC地址

返回结果

无

注意事项

无

示例

```
public class OntNotifyService
{
    /** 所有关心硬件状态的监听者 */
    private List<GatewayNotifyListener> mListeners = new ArrayList<GatewayNotifyListener>();

    private final class OntMsgHandler extends AbsMsgHandler
    {
        /**接收底层的硬件事件，并发送给监听者*/
        public String OnRecvMsg(String msg)
        {
            topic=
            props=
            for (final GatewayNotifyListener listener : mListeners)
            {
                listener.notifyListener(topic, props);
            }
         }
     }
     /*有监听者启动时，获得监听者实例*/
     public void addNotifyListener(GatewayNotifyListener service)
     {
         if (null != service)
         {
             mListeners.add(service);
         }
      }
      /*有监听者停止时，删除监听者实例*/
      public void removeNotifyListener(GatewayNotifyListener service)
      {
          if (null != service)
          {
              mListeners.remove(service);
          }
      }
}
```

1.2.2.3 WPS 接口

接口功能

IOT Agent支持接入WPS设备，通过调用该接口实现与WPS硬件的交互，当需要支持WPS设备时必须实现该接口。

接口描述

接口类GatewayWpsService，需要实现下面2个接口函数：

```
public boolean startWPS();
public boolean stopWPS();
```

参数说明

字段/方法名	参数	返回值	描述
startWPS	无	boolean	启动WPS服务，启动成功后可以与WPS设备进行对码，返回true表示启动成功，false表示启动失败
stopWPS	无	boolean	停止WPS服务，终止对码，返回true表示停止成功，false表示停止失败。

返回结果

无

注意事项

无

示例

```
public class OntWpsServiceImpl implements GatewayWpsService
{
    /** 网关提供的控制WPS的服务 */
    private IWLANService mWlanService;

    /**调用硬件接口实现启动WPS接口*/
    public boolean startWPS()
    {
           iRet = mWlanService.wlanStartWPS(SSID_INDEX);
           if(0 != iRet)
              {
                     Log.error("StartWPS failed, iRet={}.",iRet);
                     return false;
              }
```

```
        return true;
    }
    /**调用硬件接口实现停止WPS接口*/
    public boolean stopWPS()
    {
            int iRet = mWlanService.wlanStopWPS(SSID_INDEX);
            if(0 != iRet)
            {
                Log.error("stopWPS failed, iRet={}.",iRet);
                return false;
            }
        return true;
    }
}
```

1.2.2.4 控制 LED 灯接口

控制 LED 灯接口

接口功能

网关上LED灯的控制接口，在与云平台中断、bundle升级、与传感器对码等场景中IOT Agent会调用LED灯接口，通过LED灯状态变化来提示当前业务状态。

接口描述

接口类LEDControlInterface，需要实现下面3个接口函数：

```
boolean alwaysOn(LEDLightID id);
boolean alwaysOff(LEDLightID id);
boolean setFlash(LEDLightID id, int flashTime);
```

参数说明

字段/方法名	参数	返回值	描述
alwaysOn	LEDLightID	boolean	该接口将对应的LED灯设置为常亮，返回ture表示设置成功，false表示设置失败。
alwaysOff	LEDLightID	boolean	该接口将对应的LED灯设置为常灭，返回ture表示设置成功，false表示设置失败。

字段/方法名	参数	返回值	描述
setFlash	LEDLightID flashTime	boolean	该接口将对应的LED灯设置为闪烁状态，flashTime表示LED灯每次闪烁时亮的时间，单位为ms，返回ture表示设置成功，false表示设置失败。

返回结果

无

注意事项

无

示例

```
public class OntLedControlImpl implements LEDControlInterface
{
    /*网关提供的控制LED灯的服务*/
    private IHardwareControlService mHardwareControlService;

    /*调用硬件接口将LED灯设置为常亮 */
    public boolean alwaysOn(int id)
    {
        if ( null != mHardwareControlService)
        {
            mHardwareControlService.setLedMode(id, -1, 0);
            return true;
        }
        return false;
    }
    /**调用硬件接口将LED灯设置为常灭*/
    public boolean alwaysOff(int id)
    {
        if ( null != mHardwareControlService)
        {
            mHardwareControlService.setLedMode(id, 0, -1);
            return true;
        }
        return false;
    }
    /**调用硬件接口将LED灯设置为闪烁*/
    public boolean setFlash(int id, int flashTime)
    {
        if ( null != mHardwareControlService)
        {
            mHardwareControlService.setLedMode(id, flashTime, flashTime);
            return true;
        }
        return false;
    }
}
```

1.3 常用数据结构定义

1. LEDLightID 类说明

说明：枚举类用于表示LED灯的类型。

字段/方法名	描述
SERVICE	表示Agent业务状态的灯，如网关与平台的连接状态等
ZWAVE	表示ZWave模块状态的灯，比如当ZWave模块业务正常时该灯会变成常亮状态

2. UpgradeStatus 类说明

说明：枚举类用于升级状态。

字段/方法名	描述
upgrading	表示正在升级中，如固件升级中或Bundle升级中等
upgraded	表示已经升级完成

1.4 集成场景举例

将华为 Agent 集成到第三方网关

- 根据第三方网关的环境信息完成**配置项定制**；
- 根据业务功能分析需要实现的硬件扩展接口，如果没有对应的业务需求所有硬件信息接口都不需要实现：

  比如：当需要通过网关的LED灯显示Agent状态时，需要开发代码实现**控制LED灯接口**；

  需要支持IPC摄像头时，需要开发代码实现**WPS接口**和**硬件信息查询接口**；

  需要支持WIFI-Repeater时，需要开发代码实现实现**WPS接口**和**硬件信息查询接口**中的局域网设备上线通知接口。

1.5 Demo 示例说明

下面Demo是IOT Agent集成到第三方网关的一个实例，用户可参考借鉴。

场景一：简易集成场景；

1. 将IOT Agent的安装包上传到第三方网关安装目录下。

2. 解压IOT Agent安装包；
3. 按照**配置项定制**说明修改IOT Agent解压目录下的配置文件IOTAgentConfig.properties
4. 执行Agent启动脚本start.sh

场景二：深度集成场景：

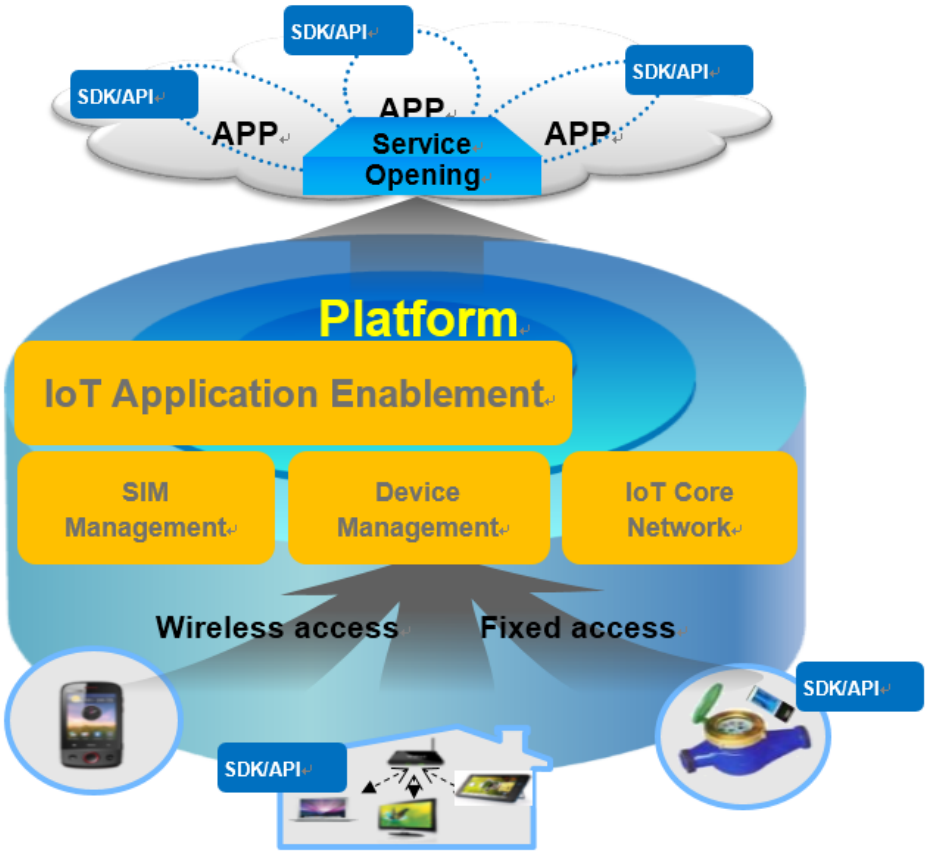
待补充。

2 Agent API 参考(普通传感器及 Eclipse Smarthome 传感器)

2.1 开发者必读

1. 概述

华为Agent作为华为IOT解决方案的一部分，主要实现了各种物联网通信协议接入、各种传感器类型的抽象、与平台之间的登录/认证/数据上报/命令下发、Bundle的部署和管理。



华为Agent会提供一个基础的能力包（中间件），并通过开放相关API，实现合作伙伴（第三方传感器，第三方物联网协议等）通过此Agent中间件快速接入到IOT解决方案中。第三方开发者只需要调用并实现相关接口，即可完成快速的集成和扩展。

华为Agent是基于OSGi框架开发，目前已经支持Zwave、ZigBee协议传感器、Onvif摄像头的接入，并提供这些协议的应用层API接口，第三方通过这些API接口可以快速完成传感器的接入。

本文档主要介绍将新协议传感器和Eclipse Smarthome传感器接入到华为IOT解决方案的方法：

1. 新协议传感器的接入开发

   新协议传感器是指该传感器与网关之间的通讯协议当前华为IOT Agent不支持，如采用蓝牙协议的传感器。

2. Eclipse Smarthome社区传感器的接入开发。

将Eclipse Smarthome社区支持的传感器接入到华为Agent，如飞利浦Hue、贝尔金Wemo等。

华为IOT Agent基于Eclipse Smarthome开源框架开发，支持除org.eclipse.smarthome.model模块以外的核心功能，Eclipse Smarthome社区传感器接入华为IOT Agent需要进行下面配置：

- 增加Smarthome社区传感器必选配置项的配置；

- 增加Smarthome社区传感器指令与华为IOT指令对应关系的配置。

2. 基本概念

- **OSGI**：Open Service Gateway Initiative，开放服务网关协议Java动态化模块化系统的一系列规范。
- **bundle**：OSGI框架中的应用插件。
- **ZWave协议**：一种新兴的基于射频的、低成本、低功耗、高可靠、适于网络的短距离无线通信技术。
- **ZigBee协议**：基于IEEE802.15.4标准的低功耗局域网协议。
- **Thing**：Smarthome框架中对现实物理设备的一种表示。
- **Channel**：作为thing的一部分，一个channel表示thing的一个功能。

3. 接口全景图

集成开发者通过全景图了解到：华为Agent提供了哪些功能？每个功能包含哪些接口？接口之间的逻辑关系如何？从而更快速的找到正确的接口来实现具体业务。

接口类别	功能	接口	说明
软件扩展接口	扩展其他协议	AbstractDiscoveryService	实现新协议传感器添加和删除接口
		BaseThingHandler	传感器命令处理接口
		BaseThingHandlerFactory	生成ThingHandler的工厂类，用于建立传感器及其命令处理类的对应关系

2.2 接口列表

2.2.1 传感器添加/删除接口

接口功能

AbstractDiscoveryService接口是设备发现和删除的接口类。

接口描述

支持其他协议传感器接入时需要实现的接口类，其中startScan接口必须实现，startRemove接口可选实现。

参数说明

字段/方法名	参数	返回值	描述
startScan	无	void	开始添加传感器的接口
startRemove	无	void	开始删除传感器的接口

返回结果

无

注意事项

无

示例

```
public class ZWaveDiscoveryService extends AbstractDiscoveryService
{
@Override
public synchronized void startScan()
                  {
                      stopFirst();
                      ZWaveController.requestAddNodeStart();
                      addingNode = true;
                   }

                  @Override
                  public synchronized void startRemove()
                 {
                    stopFirst();
                    ZWaveController.requestRemoveNodeStart();
                    removingNode = true;
                }
}
```

2.2.2 命令处理接口

接口功能

BaseThingHandler，传感器的命令处理程序，需要实现命令处理业务逻辑，并提供上报传感器在线离线状态和业务状态的接口。

接口描述

新增物联网协议或服务时必须实现该接口类，根据不同协议业务逻辑上报状态，并处理接收到指令。

参数说明

字段/方法名	参数	返回值	描述
handleCommand	ChannelUID channelUID, Command command	void	处理平台下发命令接口，需要由第三方Bundle实现该函数，处理下发命令的业务逻辑，其中channelUID是指传感器提供的一个功能的标示，比如彩灯具有调节亮度和颜色等功能，下发调节亮度的command时，对应的channelUID为brightness。
updateState	String channelID, State state	void	BaseThingHandler提供的更新传感器业务状态的接口，比如温度传感器温度变化时通过该接口，将最新的温度上报到平台；其中channelID为temperature。
updateStatus	ThingStatus status	void	

返回结果

无

注意事项

无

示例

```
//ZwaveThingHandler继承BaseThingHandler的部分代码实现
public class ZWaveThingHandler extends BaseThingHandler
{
//数据上报接口
@Override
    public void updateState(String channelID, State state)
    {
        super.updateState(channelID, state);
    }

//状态上报接口
    @Override
    protected void updateStatus(ThingStatus status)
    {
```

```
        if (status == null)
        {
            throw new IllegalArgumentException("Thing status can not be null.");
        }

        this.updateStatus(status, ThingStatusDetail.NONE);
}
//命令处理接口
public void handleCommand(ChannelUID channelUID, Command command)
    {
        if (command instanceof GetCommand)
        {
            handleGetCommand(channelUID, (GetCommand) command);
            return;
        }
...
...
}
}
```

2.2.3 命令处理工厂接口

接口功能

BaseThingHandlerFactory，生成ThingHandler的工厂类，用于建立Thing与Thinghandler的对应关系

接口描述

新增物联网协议或服务时必须实现该类，并重写supportsThingType和createHandler方法，在新增传感器时通过该工厂类创建命令处理程序。

参数说明

字段/方法名	参数	返回值	描述
createHandler	Thing thing	ThingHandler	创建并返回特定传感器的命令处理类Thinghandler
supportsThingType	ThingTypeUID thingTypeUID	boolean	是否支持的某个ThingType的处理，如果thingTypeUID为本工厂类支持的ThingType则返回ture，否则返回false。

返回结果

无

注意事项

无

示例

```
//示例为zwave协议实现此接口时的部分代码
public class ZWaveThingHandlerFactory extends BaseThingHandlerFactory
{
@Override
protected ThingHandler createHandler(Thing thing)
{
return new ZWaveThingHandler(thing);
}
}
```

3 Agent API 参考(Z-Wave)

3.1 开发者必读

1. 概述

华为Agent作为华为IOT解决方案的一部分，主要提供各种物联网通信协议栈、传感器数据的抽象、与平台云服务的通讯、Bundle的部署和管理等功能。

华为Agent是基于OSGi框架开发，目前已经支持Z-Wave、ZigBee协议传感器、Onvif摄像头的接入，并提供这些协议的应用层API接口；通过这些API接口，合作伙伴（第三方传感器或第三方家庭网关）可快速将设备接入到IOT解决方案中。

本文档主要介绍Z-Wave传感器集成到华为IOT解决方案的API，存在三种不同的集成场景：

场景一：免开发

对于已经支持的传感器类型，且控制命令和数据上报都符合协议的，则免开发直接接入。

场景二：配置开发

对于已经支持的传感器类型，但控制命令或者数据上报需要特定格式转换的，则通过配置转换规则即可接入，参考**传感器配置**。

场景三：定制开发

对于当前系统不支持的传感器，或者已经支持的传感器类型，但控制命令或者数据上报通过规则转换实现不了的，则需要开发定制Bundle接入，参考**Z-Wave协议传感器业务处理接口**。

2. 接口全景图

Z-Wave传感器接入的API全景图：

功能	接口	说明
传感器配置	ThingType.xml	传感器型号配置
	StateReport.xml	传感器数据上报规则配置
	Command.xml	传感器控制命令规则配置
Z-Wave协议传感器业务处理	BasicCommandFactory	ZWaveCommandHandler的工厂接口，根据特定传感器返回对应的命令处理类
	getNodeByNodeKey	获取传感器的设备数据的接口，传感器的静态数据主要包含支持的command、在Z-Wave网络中的nodeid、ThingType等信息
	getNodeDataByNodeKey	获取传感器状态数据的接口，状态数据是指传感器提供的各功能的state
	sendCommand	向传感器发送命令的接口
	reportData	将传感器数据上报到平台的接口
	SensorChangeListener	支持监听传感器业务状态改变化，实现业务定制和扩展，当前支持添加传感器时、添加传感器后、传感器状态变化时、传感器删除后4个业务触发点的回调

3.2 接口列表

本章节描述传感器的接入、数据上报、控制命令下发接口的形式、各字段的作用和使用方法。在“传感器配置”一节中主要描述通过配置XML方式实现对IoT平台已经支持的类型实现不同厂家或批次传感器的接入。在“Z-Wave协议传感器业务处理接口”一节中描述如何通过使用Java开发语言实现数据上报、控制命令下发的实现，从而实现IoT平台尚未支持的传感器类型的接入。

3.2.1 传感器配置

3.2.1.1 传感器型号配置

接口功能

根据传感器的厂商ID、型号等识别传感器类型；当传感器不支持上报传感器的业务类型时，系统无法自动识别传感器类型，需要在该配置文件增加传感器的设备信息与传感器类型的对应关系。

接口描述

```
ThingType.xml
```

配置需求：根据传感器basic上报设备信息识别传感器类型时配置此xml文件。

配置方式：根据传感器的设备信息在ThingType.xml配置新增传感器的标示，然后在CreateDevice.xml中定义传感器的类型；将xml放在新增bundle的src/mian/resources/TRANSFORMER-INF的目录下，在bundle启动的时候，华为Agent会自动去扫描此目录下的配置文件并将其加载。

参数说明

字段	是否可选	描述	举例
specificName	否	协议+传感器设备名称	ZWave:PowerSwitchMultilevel 其中PowerSwitchMultilevel为要加入传感器的SpecificDeviceClass的名称。
manufacturerId	否	传感器厂商ID	Z-Wave协议中：Manufacturer Specific Report Command码流中第一个值。如0086。
productType	否	传感器产品类型ID	Z-Wave协议中：Manufacturer Specific Report Command码流中第二个值，如0003。
productId	否	传感器产品ID	Z-Wave协议中：Manufacturer Specific Report Command码流中第三个值，如0062。

接口返回值

无

返回结果

无

注意事项

无

示例

1. 在ThingType.xml中根据传感器的厂商ID等设备信息，定义传感器的标示

```
<?xml version="1.0" encoding="UTF-8"?>
<thingTypes>
<aeon.doorwindow.eu>ZWave:RoutingSensorBinary-0086-0002-001D</aeon.doorwindow.eu> <!-- Aeon 欧洲门磁 -->
</thingTypes>
```

   示例说明：

   比如门磁需要通过basic通道上报数据，则需要首先在ThingType.xml中配置该门磁的型号。

   其中：

   ZWave表示协议名；

   RoutingSensorBinary表示设备的specificName；

   0086表示设备的厂商ID。

   0002表示设备的传感器类型ID。

   001D表示设备的产品ID。

   后续通过${aeon.doorwindow.eu}来表示该传感器；

2. 然后在CreateDevice.xml中将该传感器配置为ContactSensor门磁

```
<deviceType>
<thingType>${aeon.doorwindow.eu}</thingType>
<deviceType>ContactSensor</deviceType>
</deviceType>
```

3.2.1.2 传感器数据上报规则配置

接口功能

配置传感器上报数据的转化规则；当传感器上报状态采用非标准消息或存在私有定义时，需要配置转换规则；比如某恒温器通过协议中标准的ThermostatSetpointReport命令的上报水温，此时不需要增加转换规则；如果某传感器通过basicset上报烟感时，由于basicset在协议中不代表特性的业务状态，一般作为通用状态上报的命令，所以需要根据传感器的设备信息新增如下转换规则：

```
<stateReport channelId="basicset">
<thingType>${aeon.smoke.eu}</thingType> //该传感器的设备信息，在ThingType.xml中定义
<serviceId>Smoke</serviceId>       //serviceid为该传感器在平台profile中定义的服务id
<dataType>integer</dataType>
<transform>
<Smoke if="value !=0 ">"DETECTED"</Smoke>  //转换规则1，Smoke为平台profile中对应服务类型下的属性名称，"DETECTED"和"UNDETECTED"为Smoke属性的枚举值
<Smoke if="value ==0 ">"UNDETECTED"</Smoke> //转换规则2
</transform>
</stateReport>
```

接口描述

```
StateReport.xml
```

配置需求：新增传感器的数据上报状态采用非标准消息或存在私有定义时，需要在StateReport.xml中增加规则。

将此xml放在新增bundle的src/mian/resources/TRANSFORMER-INF的目录下。bundle启动的时候，华为Agent会自动去扫描此目录下的配置文件，并将其加载。

目前已经支持的数据上报通道如下图。

channel	描述
battery-level	电量
temperature	温度
humidity	湿度
ultraviolet	紫外线
motion	动感
doorwindow	门窗开关
smoke	烟
water	水
basicset	通用业务状态，需要根据传感器类型来识别对应的业务类型。
color	颜色，支持RGB和HSB格式
doorlock	门锁
garagedoor	车库卷闸门
curtain	窗帘
switch	开关
brightness	亮度

参数说明

字段	是否可选	描述	举例
channelId	否	数据上报通道，即传感器的一种功能通道。	如在Z-Wave协议中，通过basic上报数据时，可将其channelId配置为basic。 上报温度时，可将其配置为temperature

字段	是否可选	描述	举例
thingType	是	传感器型号，此参数可选。 1. 如果某个channelId的数据上报中不配置thingType，则表明对所有传感器有效。 2. 如果配置，则此channelId只针对当前配置的thingType有效	如某个channel（功能）只对某个型号的门磁有效，则需要配置<thingType>${aeon.doorwindow.eu}</thingType>
serviceId	否	和平台协商定义的service名称	如门磁的serviceid为DoorWindow
dataType	否	上报的数据类型	如门磁上报的basic数据为数据，因此需要配置dataType为integer类型
transform	否	转换规则	<transform> <status if="value !=0 ">"OPEN"</status> <status if="value ==0 ">"CLOSED"</status> </transform> 注：比如上报的0转换为”CLOSED”，非0转换为”OPEN”
pattern	是	数据转换中的模式	<pattern>%.1f</pattern> 注：比如讲字符串转换为int时保留一位小数

返回结果

无

注意事项

无

示例

```
<stateReport channelId="basic">
<thingType>${aeon.doorwindow.eu}</thingType>
<thingType>${aeon.doorwindow.eu2}</thingType>
<serviceId>DoorWindow</serviceId>
<dataType>integer</dataType>
<transform>
<status if="value !=0 ">"OPEN"</status>
<status if="value ==0 ">"CLOSED"</status>
</transform>
```

```
<mute>true</mute>
</stateReport>
```

示例说明：

比如aeon.doorwindow.eu，aeon.doorwindow.eu2这两款门磁门磁需要通过basic通道上报数据，同时平台需要根据上报的basic数据，将数据转换为平台的数据。

如：传感器通过basic上报0，在平台上将被转换为”CLOSED”;传感器通过basic上报非0，则在平台上将被转化为为”OPEN”。

3.2.1.3 传感器控制命令规则配置

接口功能

定义传感器控制命令的转换规则，将华为IOT定义的设备属性转换为可被传感器识别的配置，如果传感器不支持控制命令本节忽略；比如我们定义了serviceId=Swtich表示开关服务，其中status 等于“ON”表示开，status等于“OFF”表示关，而AEON的开关传感器0x63表示开，0x00表示关，则需要配置如下规则：

```
<command method="SWITCH" serviceId="Switch">
<thingType>${aeon.smart.switch.eu}</thingType>
<commandClass>com.huawei.m2m.gateway.core.thing.SwitchCommand</commandClass>
    <transform>
        <status if="command.body.status == 'ON'">0x63</status>  //转换关系1
        <status if="command.body.status == 'OFF'">0x00</status> //转换关系2
    </transform>
</command>
```

第一行的method和serviceID，参考华为IOT设备属性定义文件IoT device profile.xlsx中开关服务命令的定义；

第二行表示该规则只对特定传感器类型起作用，${aeon.smart.switch.eu}在ThingType.xml中定义

第三行是Agent提供的控制命令类，与ServiceID对应，对应关系如下：

ServiceID	Agent提供的控制命令类
Switch	com.huawei.m2m.gateway.core.thing.SwitchCommand
Brightness	com.huawei.m2m.gateway.core.thing.BrightnessCommand
DoorLock	com.huawei.m2m.gateway.core.thing.DoorLockCommand
GarageDoor	com.huawei.m2m.gateway.core.thing.GarageDoorCommand
Color	com.huawei.m2m.gateway.core.thing.ChangeColorCommand
WindowCovering	com.huawei.m2m.gateway.core.thing.CurtainCommand

接口描述

```
Command.xml
```

配置需求：如果传感器的控制命令存在私有定义时，需要在Command.xml中增加规则。

将此xml放在新增bundle的src/mian/resources/TRANSFORMER-INF的目录下。bundle启动的时候，华为Agent会自动去扫描此目录下的配置文件，并将其加载。

参数说明

字段	是否可选	描述	举例
method	否	华为IOT设备属性中定义的method。	如WindowCovering服务中定义了OPEN、CLOSE、STOP三个method
thingType	是	传感器型号，此参数可选。 1. 如果某个channelId的数据上报中不配置thingType，则表明对所有传感器有效。 2. 如果配置，则此channelId只针对当前配置的thingType有效	如某个channel（功能）只对某个型号的门磁有效，则需要配置<thingType>${aeon.doorwindow.eu}</thingType>
serviceId	否	华为IOT设备属性中定义的service名称	如开关的serviceid为Switch
commandClass	否	Agent提供的控制命令类	参考上面ServiceID与命令控制类的对应关系。
transform	否	转换规则	<status if="command.body.status == 'ON'">0x63</status>注：status为“ON”时转换为0x63

返回结果

无

注意事项

无

示例

```
<command method="SWITCH" serviceId="Switch">
<thingType>${aeon.smart.switch.eu}</thingType>
<commandClass>com.huawei.m2m.gateway.core.thing.SwitchCommand</commandClass>
    <transform>
        <status if="command.body.status == 'ON'">0x63</status>  //转换关系1
        <status if="command.body.status == 'OFF'">0x00</status> //转换关系2
    </transform>
</command>
```

3.2.2 Z-Wave 协议传感器业务处理接口

3.2.2.1 传感器命令处理接口

接口功能

命令处理接口创建并返回特定传感器类型的命令处理类，用于处理传感器发送给网关的command；Agent支持对所有Z-Wave command的消息解析，但Agent只实现了部分Z-Wave command的业务处理逻辑，当收到传感器的command属于Agent未实现的Z-Wave command时才会调用该命令处理类的handle接口。

接口描述

```
BasicCommandFactory
```

当新增传感器的command当前Agent不支持时需要实现该接口，并通过osgi声明式服务的方式注册到Agent框架，注册时声明该接口对应的传感器类型，通过thingTypeUID来标识，如下所示：

```
<property name="thingTypeUID" type="String" value="Z-Wave:SetpointThermostat-0002-0005-0004"/>
```

参数说明

字段/方法名	参数	返回值	描述
createCommandHandler	ThingTypeUID thingtypeUID	ZWaveCommandHandler	根据thingtypeUID创建ZWaveCommandHandler类的实例。Agent收到传感器的command后调用该ZwaveCommandHandler的handle函数进行业务处理。

返回结果

无

注意事项

无

示例

```
//新增一个BasicCommandFactory接口的实现类：
public class CustomizedSensorCommandFactory implements BasicCommandFactory
{
    private Map<String, ZWaveCommandHandler> commandHandlerFactory = new ConcurrentHashMap<String,
ZWaveCommandHandler>();

    public CustomizedSensorCommandFactory()
    {
        commandHandlerFactory.put("zwave:SetpointThermostat-0002-0005-0004", new
ThermostatSetpointReportHandler());
    }

    public ZWaveCommandHandler createCommandHandler(ThingTypeUID thingtypeUID)
    {
        return commandHandlerFactory.get(thingtypeUID.toString());
    }
}
```

然后通过下面XML将该类作为服务声明，在bundle启动时框架会根据该XML的配置来自动加载该服务，该XML使用标准的OSGi声明式服务，详细的字段描述请参考OSGi规范：

```
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" immediate="true"
name="com.huawei.m2m.gateway.zwave.customizedsensor.CustomizedSensorCommandFactory">
   <implementation
class="com.huawei.m2m.gateway.zwave.customizedsensor.CustomizedSensorCommandFactory"/>
   <service>
      <provide interface="com.huawei.m2m.gateway.zwave.service.BasicCommandFactory"/>
   </service>
   <property name="thingTypeUID" type="String" value="zwave:SetpointThermostat-0002-0005-0004"/>
</scr:component>
```

3.2.2.2 获取传感器的设备数据

接口功能

提供给第三方传感器Bundle获取传感器的设备数据的接口，传感器的静态数据主要包含支持的command、在Z-Wave网络中的nodeid、ThingType等信息。

接口描述

```
public ZWaveNode getNodeByNodeKey(String nodeKey);
```

属于ZWaveControllerService接口类提供的函数，使用时需要先获得ZWaveControllerService的实例。

参数说明

字段	类型	描述
nodeKey	String	传感器的唯一标识

返回结果

ZWaveNode

ZWave传感器的数据类，包含该传感器支持的command、在Z-Wave网络中的nodeid、ThingType等信息，详情参考数据结构定义。

注意事项

无

示例

```
// Z-WaveController，它继承于Z-WaveCommand
ZWaveNode node = ZWaveController.getNodeByNodeKey(nodeKey);
ThingTypeUID thingTypeUID = node.getThingTypeUID();
```

3.2.2.3 获取传感器的状态数据

接口功能

提供给第三方传感器Bundle获取传感器状态数据的接口，状态数据是指传感器提供的各功能的state，如门磁传感器的状态数据包含门的开关状态和电池的电量。

接口描述

```
public Map<String, ZWaveUpdateValueType> getNodeDataByNodeKey(String nodeKey);
```

属于ZWaveControllerService接口类提供的函数，使用时需要先获得ZWaveControllerService的实例。

参数说明

字段	类型	描述
nodeKey	String	传感器的唯一标识

返回结果

Map<String, ZWaveUpdateValueType>

状态数据的MAP表，key为传感器功能名，等于channelId；value为对应功能的state数据，支持数值类型数据和二进制类型数据。

注意事项

无

示例

```
// 获取传感器的电量数据，并上报。
Map<String, ZWaveUpdateValueType> dataMap = ZWaveController.getNodeDataByNodeKey (nodeKey);

ZWaveUpdateValueType data = dataMap.get(ChannelId.BATTERY_LEVEL);
```

```
//调用reportData数据将数据上报
ZWaveController.reportData(nodeKey, ChannelId.BATTERY_LEVEL, data);
```

3.2.2.4 发送指令

接口功能

提供给第三方传感器Bundle向传感器发送命令的接口。

接口描述

```
public void sendCommand(String nodeKey, ZWaveCommand command);
```

当需要改变传感器的状态或者查询传感器状态时使用；

属于ZWaveControllerService接口类提供的函数，使用时需要先获得ZWaveControllerService的实例。。

参数说明

字段	类型	描述
nodeKey	String	传感器的唯一标识
command	ZWaveCommand	ZWave Command实体类

返回结果

无

注意事项

无

示例

```
// DoorLockOperationSet为门锁支持的一个command，它继承于Z-WaveCommand
public class DoorLockOperationSet extends ZWaveCommand
{
   ...
}
//创建一个要发送的command的对象
DoorLockOperationSet doorLockOperationSet = new DoorLockOperationSet();
//调用接口，将此command发送给相应的传感器
ZWaveController.sendCommand(nodeKey, doorLockOperationSet);
```

3.2.2.5 上报数据

接口功能

将传感器的状态上报到IOT平台。

接口描述

```
public void reportData(String nodeKey, String channelId, State data);
```

传感器状态变化时调用该接口将状态上报到IOT平台；

属于ZWaveControllerService接口类提供的函数，使用时需要先获得ZWaveControllerService的实例

参数说明

字段	类型	描述
nodeKey	String	传感器的唯一标识
channelId	String	上报数据通道
data	State	上报的数据

返回结果

无

注意事项

无

示例

比如要在COMMAND_DOOR_LOCK_CONFIGURATION_REPORT命令的handle（业务处理）代码中上报数据。

```
//收到COMMAND_DOOR_LOCK_CONFIGURATION_REPORT命令后，将此report命令中携带的信息通过
ChannelId.DOORLOCK通道上报上去。
public void handle(String nodeKey, ZWaveCommand command)
    {
        DoorLockOperationReport report = (DoorLockOperationReport)command;
        //获取ZWaveController实例

        //获取report命令中携带的需要上报到的数据
        String doorCondition = getDoorCondition(report.getDoorCondition());
        //调用reportData数据将数据上报
        ZWaveController.reportData(nodeKey, ChannelId.DOORLOCK, new StringType(doorCondition));

    }
```

3.2.2.6 传感器业务触发点

接口功能

SensorChangeListener在传感器状态变化时，通知第三方bundle，以便进行业务扩展，采用注册通知的方法。

接口描述

```
void notifyListeners(EventType event, String nodeKey);
```

在传感器状态变化时，Agent框架会调用第三方bundle的该接口，比如在新增传感器添加完成后需要实现额外的业务处理，需要在新增bundle中实现此接口中的notifyListeners方法，并通过OSGI声明式服务将其接口注册。

参数说明

字段	类型	描述
event	EventType	表示发生的事件： ADDING //传感器添加时 ADDED //传感器添加完成后 REMOVED //传感器删除后 UPDATESTATUS //传感器状态更新时
nodeKey	String	发送状态变化的传感器标识

返回结果

无

注意事项

无

示例

新增bundle中只需实现SensorChangeListener接口中的notifyListeners方法，并在该类xml配置文件中声明实现了该接口。

```
public class ThirdCallbackListener implements
      SensorChangeListener{
      public void notifyListeners(EventType event, String nodeKey) {
// TODO Auto-generated method stub
 }
}

<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" activate="activate"
deactivate="deactivate"  immediate="true" name="com.huawei.m2m.gateway.Z-Wave.
legacydoorwindow.ThirdCallbackListener ">
   <implementation class=" com.huawei.m2m.gateway.Z-Wave.legacydoorwindow.ThirdCallbackListener"/>
   <service>
      <provide interface="com.huawei.m2m.gateway.service.Z-Wave.SensorChangeListener "/>
   </service>
</scr:component>
```

3.3 常用数据结构定义

1. ZWaveCommandHandler 接口说明

说明：Agent收到传感器的command后调用该传感器对应的ZwaveCommandHandler的handle函数进行业务处理。

字段/方法名	参数	返回值	描述
handle	ZWaveNode node, ZWaveCommand command	void	node：表示对应的传感器的节点数据 command：传感器上报的command。

2. ZWaveNode 数据类说明

说明：保存Z-Wave传感器的静态数据。

字段/方法名	参数	返回值	描述
getNodeId	Null	int	返回该传感器在Z-Wave网络中的nodeid。
getThingTypeUID	Null	ThingTypeUID	返回传感器的ThingTypeUID
getAllSupportedCommandClasses	Null	Set<Integer>	返回传感器支持的所有command，Integer为Z-Wave协议中定义的commandClassId。
getNodeState	Null	ZWaveNodeState	返回传感器的节点状态，“ALIVE”表示传感器在线，“FAILED”表示传感器离线。

4 Agent API 参考(ZigBee)

4.1 开发者必读

1. 概述

华为Agent作为华为IOT解决方案的一部分，主要实现了各种物联网通信协议接入、各种传感器类型的抽象、与平台之间的登录/认证/数据上报/命令下发、Bundle的部署和管理。

华为Agent会提供一个基础的能力包（中间件），并通过开放相关API，实现合作伙伴（第三方传感器，第三方物联网协议等）通过此Agent中间件快速接入到IOT解决方案中。第三方开发者只需要调用并实现相关接口，即可完成快速的集成和扩展。

华为Agent是基于OSGi框架开发，目前已经支持Z-Wave、ZigBee协议传感器、Onvif摄像头的接入，并提供这些协议的应用层API接口，第三方通过这些API接口可以快速完成传感器的接入。

本文档主要介绍将ZigBee传感器接入到华为IOT解决方案的集成场景：

场景一：该传感器类型IOT Agent已经支持，当前已经支持的传感器类型包括门磁、智能插座等。

接入方法：

1. 该传感器支持上报传感器的业务类型，同时传感器上报业务状态、给传感器下发的控制命令采用ZigBee协议标准消息，此时不需要任何开发和配置，可直接接入该传感器。
2. 该传感器不支持上报传感器的业务类型或传感器上报业务状态和下发控制指令在协议中无明确定义（如使用非标准消息或使用扩展字段等），则需要通过XML文件配置相应的转换规则，具体配置方法请参考**传感器配置**。

场景二：该传感器类型IOT Agent暂不支持。

接入方法：先按照场景一添加传感器配置，然后实现该传感器的业务消息处理。请参考**ZigBee协议传感器业务处理接口**。

2. 接口全景图

集成开发者通过全景图了解到：华为Agent提供了哪些功能？每个功能包含哪些接口？接口之间的逻辑关系如何？从而更快速的找到正确的接口来实现具体业务。

接口类别	功能	接口	说明
软件扩展接口	传感器配置	ThingType.xml	传感器型号配置
		StateReport.xml	传感器数据上报规则配置
		Command.xml	传感器控制命令规则配置
	ZigBeee协议传感器业务处理接	ZigBeeeCommandHandlerFactory	ZigBeeCommandHandler的工厂接口，根据特定传感器返回对应的命令处理类生
		getNodeByNodeKey	获取传感器的设备数据的接口，传感器的静态数据主要包含在ZigBee网络中的nodeid、ThingType等信息
		getNodeDataByNodeKey	获取传感器状态数据的接口，状态数据是指传感器提供的各功能的state
		sendCommand	向传感器发送命令的接口
		reportData	将传感器数据上报到平台的接口
		SensorChangeListener	支持监听传感器业务状态改变化，实现业务定制和扩展，当前支持添加传感器时、添加传感器后、传感器状态变化时、传感器删除后4个业务触发点的回调

4.2 接口列表

本章节描述传感器的接入、数据上报、控制命令下发接口的形式、各字段的作用和使用方法。在“传感器配置”一节中主要描述通过配置XML方式实现对IoT平台已经支持的类型实现不同厂家或批次传感器的接入。在“Z-Wave协议传感器业务处理接口”一节中描述如何通过使用Java开发语言实现数据上报、控制命令下发的实现，从而实现IoT平台尚未支持的传感器类型的接入。↵

4.2.1 传感器配置

4.2.1.1 传感器型号配置

接口功能

根据传感器的厂商ID、型号等识别传感器类型；当传感器不支持上报传感器的业务类型时，系统无法自动识别传感器类型，需要在该配置文件增加传感器的设备信息与传感器类型的对应关系。

接口描述

```
ThingType.xml
```

配置需求：根据传感器的厂商，型号属性信息识别传感器类型时配置此xml文件。

配置方式：根据传感器的设备信息在ThingType.xml配置新增传感器的标示，然后在CreateDevice.xml中定义传感器的类型；将xml放在新增bundle的src/mian/resources/TRANSFORMER-INF的目录下，在bundle启动的时候，华为Agent会自动去扫描此目录下的配置文件并将其加载。

参数说明

字段	是否可选	描述	举例
specificName	否	协议	固定为zigbee
manufacturerId	否	传感器厂商ID	ZigBee协议中：profileId为0x0104，clusterId为0x0000，attributeId为0x0004的值。如1037。
mode id	否	传感器产品类型	ZigBee协议中：profileId为0x0104，clusterId为0x0000，attributeId为0x0005的值。如Door_Sensor。

接口返回值

无

返回结果

无

注意事项

无

示例

1. 在ThingType.xml中根据传感器的厂商ID等设备信息，定义传感器的标示

```
<?xml version="1.0" encoding="UTF-8"?>

<thingTypes>
         <nexturn.doorsensor> zigbee:1037-Door_Sensor</nexturn.doorsensor>
</thingTypes>
```

   示例说明：

   比如门磁需要通过basic通道上报数据，则需要首先在ThingType.xml中配置该门磁的型号。

   其中：

   zigbee表示协议名；

   1037表示设备的厂商ID。

   Door_Sensor表示设备的传感器类型ID。

   后续通过${nexturn.doorsensor}来表示该传感器；

2. 然后在CreateDevice.xml中将该传感器配置为ContactSensor门磁

```
<deviceType>
       <thingType>${nexturn.doorsensor}</thingType>
       <deviceType>ContactSensor</deviceType>
</deviceType>
```

4.2.1.2 传感器数据上报规则配置

接口功能

配置传感器上报数据的转化规则；当传感器上报状态采用非标准消息或存在私有定义时，需要配置转换规则；比如某恒温器通过协议中标准的ThermostatSetpointReport命令的上报水温，此时不需要增加转换规则；如果某传感器通过basicset上报烟感时，由于basicset在协议中不代表特性的业务状态，一般作为通用状态上报的命令，所以需要根据传感器的设备信息新增如下转换规则：

```
<stateReport channelId="basicset">
    <thingType>${aeon.smoke.eu}</thingType> //该传感器的设备信息，在ThingType.xml中定义
    <serviceId>Smoke</serviceId>       //serviceid为该传感器在平台profile中定义的服务id
    <dataType>integer</dataType>
<transform>
<Smoke if="value !=0 ">"DETECTED"</Smoke>  //转换规则1，Smoke为平台profile中对应服务类型下的属性名称，"DETECTED"和"UNDETECTED"为Smoke属性的枚举值
<Smoke if="value ==0 ">"UNDETECTED"</Smoke> //转换规则2
</transform>
</stateReport>
```

接口描述

```
StateReport.xml
```

配置需求：新增传感器的数据上报状态采用非标准消息或存在私有定义时，需要在StateReport.xml中增加规则。

将此xml放在新增bundle的src/mian/resources/TRANSFORMER-INF的目录下。bundle启动的时候，华为Agent会自动去扫描此目录下的配置文件，并将其加载。

目前已经支持的数据上报通道如下图。

channel	描述
battery-level	电量
temperature	温度
humidity	湿度
ultraviolet	紫外线
motion	动感
doorwindow	门窗开关
smoke	烟
water	水
basicset	通用业务状态，需要根据传感器类型来识别对应的业务类型。
color	颜色，支持RGB和HSB格式
doorlock	门锁
garagedoor	车库卷闸门
curtain	窗帘
switch	开关
brightness	亮度

参数说明

字段	是否可选	描述	举例
channelId	否	数据上报通道，即传感器的一种功能通道。	如上报温度时，可将其配置为temperature

字段	是否可选	描述	举例
thingType	是	传感器型号，此参数可选。 1. 如果某个channelId的数据上报中不配置thingType，则表明对所有传感器有效。 2. 如果配置，则此channelId只针对当前配置的thingType有效	如某个channel（功能）只对某个型号的门磁有效，则需要配置<thingType>${aeon.doorwindow.eu}</thingType>
serviceId	否	和平台协商定义的service名称	如门磁的serviceid为DoorWindow
dataType	否	上报的数据类型	如门磁上报的basic数据为数字，因此需要配置dataType为integer类型
transform	否	转换规则	<transform> <status if="value !=0 ">"OPEN"</status> <status if="value ==0 ">"CLOSED"</status> </transform> 注：比如上报的0转换为”CLOSED”，非0转换为”OPEN”
pattern	是	数据转换中的模式	<pattern>%.1f</pattern> 注：比如讲字符串转换为int时保留一位小数

返回结果

无

注意事项

无

示例

```
<stateReport channelId="basic">
    <thingType>${aeon.doorwindow.eu}</thingType>
    <thingType>${aeon.doorwindow.eu2}</thingType>
    <serviceId>DoorWindow</serviceId>
    <dataType>integer</dataType>
    <transform>
    <status if="value !=0 ">"OPEN"</status>
    <status if="value ==0 ">"CLOSED"</status>
    </transform>
```

```
    <mute>true</mute>
</stateReport>
```

示例说明：

比如aeon.doorwindow.eu，aeon.doorwindow.eu2这两款门磁门磁需要通过basic通道上报数据，同时需要根据上报的basic数据，将数据转换为平台的数据。

如：传感器通过basic上报0，在平台上将被转换为”CLOSED”;传感器通过basic上报非0，则在平台上将被转化为为”OPEN”。

4.2.2 ZigBee 协议传感器业务处理接口

4.2.2.1 传感器命令处理工厂接口

接口功能

命令处理工厂接口创建并返回指定传感器的命令处理类；根据ThingTypeUID生成相应command的业务处理类（ZigbeeCommandHandler）。

Agent支持对所有ZigBee command的消息解析，但Agent只实现了部分ZigBee command的业务处理逻辑，当收到传感器的command属于Agent未实现的ZigBee command时才会调用该接口，分发到第三方Bundle来处理。

接口描述

```
ZigBeeCommandHandlerFactory
```

当新增传感器的command的业务处理当前Agent不支持时需要实现该接口，并通过osgi声明式服务的方式注册Agent框架，注册时声明该接口对应的传感器类型，通过thingTypeUID来标识，如下所示：

```
<property name="thingTypeUID" type="String" value="zigbee:1037-Door_Sensor"/>
```

参数说明

字段/方法名	参数	返回值	描述
createCommandHandler	ThingTypeUID thingtypeUID	ZigBeeCommandHandler	根据thingtypeUID创建ZigBeeCommandHandler子类的实例。此接口在Agent收到串口command后处理具体业务逻辑时被调用。

返回结果

无

注意事项

无

示例

新增bundle中新增一个ZigBeeCommandHandlerFactory接口的实现类：

```
public class LegacyDoorWindowCommandFactory implements ZigBeeCommandHandlerFactory
{
  ...
}
```

然后通过下面XML将该类作为服务声明，在bundle启动时框架会根据该XML的配置来自动加载该服务，该XML使用标准的OSGi声明式服务，详细的字段描述请参考OSGi规范：

```
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" activate="activate"
deactivate="deactivate"  immediate="true"
name="com.huawei.m2m.gateway.ZigBee.legacydoorwindow.LegacyDoorWindowCommandFactory">
<implementation
class="com.huawei.m2m.gateway.zigbee.legacydoorwindow.LegacyDoorWindowCommandFactory"/>
<service>
<provide interface=com.huawei.m2m.gateway.zigbee.service.ZigBeeCommandHandlerFactory"/>
</service>
<property name="thingTypeUID" type="String" value="zigbee:1037-Door_Sensor"/>
</scr:component>
```

4.2.2.2 获取传感器的设备数据

接口功能

提供给第三方传感器Bundle获取传感器的设备数据的接口，传感器的静态数据主要包含endpoint、nwkAddress、ieeeAddress以及在ZigBee网络中的nodeid、ThingType等信息。

接口描述

```
public ZigBeeNode getNodeByNodeKey(String nodeKey);
```

属于ZigBeeControllerService接口类提供的函数，使用时需要先获得ZigBeeControllerService的实例。

参数说明

字段	类型	描述
nodeKey	String	传感器的唯一标识

返回结果

ZigBeeNode

ZigBee传感器的数据类，包含该传感器的endpoint、nwkAddress、ieeeAddress以及在ZigBee网络中的nodeid、ThingType等信息，详情参考数据结构定义。

注意事项

无

示例

```
ZigBeeNode node = ZigBeeController.getNodeByNodeKey(nodeKey);
ThingTypeUID thingTypeUID = node.getThingTypeUID();
```

4.2.2.3 获取传感器的状态数据

接口功能

提供给第三方传感器Bundle获取传感器状态数据的接口，状态数据是指传感器提供的各功能的state，如门磁传感器的状态数据包含门的开关状态和电池的电量。

接口描述

```
public Map<String, ZigBeeUpdateValueType> getNodeDataByNodeKey(String nodeKey);
```

属于ZigBeeControllerService接口类提供的函数，使用时需要先获得ZigBeeControllerService的实例。

参数说明

字段	类型	描述
nodeKey	String	传感器的唯一标识

返回结果

Map<String, ZigBeeUpdateValueType>

状态数据的MAP表，key为传感器功能名，等于channelId；value为对应功能的state数据，支持数值类型数据和二进制类型数据。

注意事项

无

示例

```
// 获取传感器的电量数据，并上报。
Map<String, ZigBeeUpdateValueType> dataMap = ZigBeeController.getNodeDataByNodeKey (nodeKey);

ZigBeeUpdateValueType data = dataMap.get(ChannelId.BATTERY_LEVEL);

//调用reportData数据将数据上报
ZigBeeController.reportData(nodeKey, ChannelId.BATTERY_LEVEL, data);
```

4.2.2.4 发送指令

接口功能

提供给第三方传感器Bundle向传感器发送命令的接口。

接口描述

```
public void sendCommand(String nodeKey, ZigBeeCommand command);
```

当需要改变传感器的状态或者查询传感器状态时使用；

属于ZigBeeControllerService接口类提供的函数，使用时需要先获得ZigBeeControllerService的实例。

参数说明

字段	类型	描述
nodeKey	String	传感器的唯一标识
command	ZigBeeCommand	ZigBee Command实体类

返回结果

无

注意事项

无

示例

```
// DoorLockOperationSet为门锁支持的一个command，它继承于ZigBeeCommand
public class DoorLockOperationSet extends ZigBeeCommand
{
  ...
}
//创建一个要发送的command的对象
DoorLockOperationSet doorLockOperationSet = new DoorLockOperationSet();
//调用接口，将此command发送给相应的传感器
ZigBeeController.sendCommand(nodeKey, doorLockOperationSet);
```

4.2.2.5 上报数据

接口功能

将传感器的状态上报到IOT平台。

接口描述

```
public void reportData(String nodeKey, String channelId, State data);
```

传感器状态变化时调用该接口将状态上报到IOT平台；

属于ZigBeeControllerService接口类提供的函数，使用时需要先获得ZigBeeControllerService的实例

参数说明

字段	类型	描述
nodeKey	String	传感器的唯一标识
channelId	String	上报数据通道
data	State	上报的数据

返回结果

无

注意事项

无

示例

```
//比如要在COMMAND_DOOR_LOCK_CONFIGURATION_REPORT命令的handle（业务处理）代码中上报数据。

//收到COMMAND_DOOR_LOCK_CONFIGURATION_REPORT命令后，将此report命令中携带的信息通过
ChannelId.DOORLOCK通道上报上去。

public void handle(String nodeKey, ZigBeeCommand command)
    {
        DoorLockOperationReport report = (DoorLockOperationReport)command;
        //获取ZigBeeController实例

        //获取report命令中携带的需要上报到的数据
        String doorCondition = getDoorCondition(report.getDoorCondition());
        //调用reportData数据将数据上报
        ZigBeeController.reportData(nodeKey, ChannelId.DOORLOCK, new StringType(doorCondition));

    }
```

4.2.2.6 传感器业务触发点

接口功能

SensorChangeListener在传感器状态变化时，通知第三方bundle，以便进行业务扩展，采用注册通知的方法。

接口描述

```
void notifyListeners(EventType event, String nodeKey);
```

在传感器状态变化时，Agent框架会调用第三方bundle的该接口，比如在新增传感器添加完成后需要实现额外的业务处理，需要在新增bundle中实现此接口中的notifyListeners方法，并通过OSGI声明式服务将其接口注册。

参数说明

字段	类型	描述
event	EventType	表示发生的事件： ADDING //传感器添加时 ADDED //传感器添加完成后 REMOVED //传感器删除后 UPDATESTATUS //传感器状态更新时
nodeKey	String	发送状态变化的传感器标识

返回结果

无

注意事项

无

示例

新增bundle中只需实现SensorChangeListener接口中的notifyListeners方法即可。并在实现类的OSGI的声明xml中将此接口提供出去。

```
public class ThirdCallbackListener implements
SensorChangeListener{
public void notifyListeners(EventType event, String nodeKey) {
// TODO Auto-generated method stub
}
}
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" activate="activate"
deactivate="deactivate"  immediate="true" name="com.huawei.m2m.gateway.ZigBee.
legacydoorwindow.ThirdCallbackListener ">
   <implementation class=" com.huawei.m2m.gateway.ZigBee.legacydoorwindow.ThirdCallbackListener"/>
   <service>
      <provide interface="com.huawei.m2m.gateway.service.sensor.  SensorChangeListener "/>
   </service>
</scr:component>
```

4.3 常用数据结构定义

1. ZigBeeCommandHandler 接口说明

说明：ZigBee中所有Command的业务处理接口，具体的command的逻辑处理需要实现接口中的handle方法。

字段/方法名	参数	返回值	描述
handle	ZigBeeNode node ZigBeeCommand command	void	参数command对应的业务处理方法

2. ZigBeeNode 数据类说明

说明：保存ZigBee传感器的静态数据。

字段/方法名	参数	返回值	描述
getNodeKey	Null	String	返回该传感器在ZigBee网络中的nodeKey。
getThingTypeUID	Null	ThingTypeUID	返回传感器的ThingTypeUID
getNwkAddress	Null	Set<Integer>	返回该传感器在ZigBee网络中的NwkAddress。
getNodeState	Null	ZigBeeNodeState	返回传感器的节点状态，“ALIVE”表示传感器在线，“FAILED”表示传感器离线。

5 Agent Lite API 参考(Android)

5.1 开发者必读

1. 概述

华为IoT Agent Lite在智慧家庭、工业物联网、车联网等领域为智能设备提供了标准接入华为IoT联接管理平台的能力。主要面向IPC、轻量级网关、工业网关、车机等计算能力较强的终端/网关设备。

2. 接口全景图

集成开发者通过全景图了解到：Agent Lite提供了哪些功能？每个功能包含哪些接口？接口之间的逻辑关系如何？从而更快速的找到正确的接口来实现具体业务。

功能	接口	说明
直连设备接入	BaseService.init	初始化Agent Lite模块资源
	BaseService.destroy	释放Agent Lite模块资源
	BindService.bind	设备绑定
	BindConfig.setConfig	绑定配置
	LoginService.login	设备登录
	LoginConfig.setConfig	登录配置
	LoginService.logout	设备登出
	HubService.deviceStatusUpdate	设备状态更新

功能	接口	说明
	HubService.TOPIC_UNBINDDEVICE	设备解绑定
网关管理非直连设备	HubService.addDevice	设备添加
	HubService.rmvDevice	设备删除
设备服务数据上报	DataTransService.reportData	设备服务数据上报
设备命令接收	DataTransService.TOPIC_COMMAND_RECEIVE	设备服务命令接收

5.2 接口列表

Agent Lite对外提供的接口主要包括广播机制、直连设备的接入、非直连设备的添加和删除、设备数据的上报以及设备命令接收接口。

5.2.1 广播机制

Agent Lite广播采用android系统的本地广播，用来接收Agent Lite上报的消息。

由于AgentLite广播采用本地广播，所以注册广播时只能采用动态注册方式。

5.2.2 直连设备接入

第三方开发者获得Agent Lite后首先需要将设备接入到IoT连接管理平台中。

直连设备：通过设备绑定、登录流程直接接入IoT平台的设备。

非直连设备：通过网关设备接入IoT平台的设备。

5.2.2.1 初始化 Agent Lite 资源

接口功能

初始化Agent Lite资源。

接口描述

```
public static boolean init(String workPath, String logPath, Context context)
```

接口所属类

BaseService

参数说明

字段	必选/可选	类型	描述
workPath	必选	String	Agent Lite工作路径，用于存放Agent Lite的配置文件与生产的临时文件。
logPath	可选	String	日志路径（若日志路径为空则日志写在工作路径中）
context	必选	Context	Android应用程序上下文

接口返回值

返回值	描述
true	成功
false	失败

示例

```
// 开发者调用该接口初始化Agent Lite资源
BaseService.init("/sdcard/helloWorld", null, context);
```

5.2.2.2 释放 Agent Lite 资源

接口功能

调用此函数，Agent Lite会释放申请的所有动态资源（内存、线程等等）。

接口描述

```
public static void destroy();
```

接口所属类

BaseService

接口返回值

返回值	描述
true	成功
false	失败

示例

```
// 开发者调用该接口销毁Agent Lite资源
BaseService.destroy();
```

5.2.2.3 绑定配置

接口功能

在设备绑定前配置平台的地址和端口

接口描述

```
public static boolean setConfig(int key, String value)
```

接口所属类

BindConfig

参数说明

字段	必选/可选	类型	描述
key	必选	int	设备绑定的配置项 平台IP：BindConfig.BIND_CONFIG_ADDR 平台端口：BindConfig.BIND_CONFIG_PORT
value	必选	String	设置的值 平台IP：Agent Lite对接平台地址 平台端口：8943

接口返回值

返回值	描述
true	成功
false	失败

返回结果

无

示例

```
BindConfig.setConfig(BindConfig.BIND_CONFIG_ADDR, “127.0.0.1”);
BindConfig.setConfig(BindConfig.BIND_CONFIG_PORT, “8943”);
```

5.2.2.4 设备绑定

接口功能

设备第一次接入IoT联接管理平台时需要进行绑定操作，上层应用通过调用该接口传入设备序列号或者MAC地址以及设备信息来绑定到IoT联接管理平台。在绑定前需要调用

BindConfig.setConfig接口设置绑定服务器IP与端口（IoCM服务器地址与端口，Agent Lite会配置默认端口8943）。

说明

设备绑定是指设备第一次接入IoT平台的过程，需要开发者先在IoT平台注册直连设备，之后在设备上发起绑定操作，将设备绑定到IoT平台上。如果未在IoT平台注册该设备，则绑定操作会失败，Agent Lite将会等待一段时间继续尝试。

接口描述

```
public static boolean bind(String verifyCode, IotaDeviceInfo deviceInfo);
```

接口所属类

BindService

参数说明

字段	必选/可选	类型	描述
verifyCode	必选	String	设备绑定验证码
deviceInfo	必选	**IotaDeviceInfo** 类	设备信息

接口返回值

返回值	描述
true	成功
false	失败

注意

- 此返回值是调用接口的同步返回结果，返回0只是说明接口调用成功，并不说明绑定成功，绑定成功需要收到BindService.TOPIC_BINDDEVICE_RSP广播。
- 该设备未绑定时，该接口会自动尝试重新绑定，重试间隔采用退避算法，**开发者无需进行重试**，如果需要停止重试，调用Agent Lite销毁接口即可。同时请勿短时间内多次调用该接口。
- 当前绑定流程的重试策略为，如果绑定失败，则30秒后继续进行重试，如果重试超过5次（总计尝试超过6次），则返回失败，不再进行重试。如果想要重新发起绑定，建议让用户重启设备。

返回结果

广播名称	广播参数	成员	描述
TOPIC_BINDDEVICE_RSP	IotaMessage对象（使用intent.getSerializableExtra(BindService.BIND_BROADCAST_MSG_IE_IOTAMSG)方法获取）	BIND_IE_RESULT	绑定结果
		BIND_IE_DEVICEID	平台分配的逻辑设备ID
		BIND_IE_DEVICESECRET	设备接入的鉴权秘钥
		BIND_IE_APPID	开发者应用ID
		BIND_IE_HA_ADDR	HA服务器地址
		BIND_IE_LVS_ADDR	LVS服务器地址

注意事项

如果接入设备是网关设备，则DeviceType填写为“Gateway”,如果为传感器设备直连，则填写为对应的设备类型。

示例

```
//调用设备绑定接口
BindService.bind(new IotaDeviceInfo("nodeId", "manufacturerId", " Gateway", "model",
"protocolType"));

//接收设备绑定响应消息
//当设备成功绑定之后，Agent Lite会返回给UI如下几个参数，需要UI进行持久化存储，设备登录前需要提前进行配置
BroadcastReceiver mBindRsp;
mBindRsp = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
    //Do Something
    IotaMessage iotaMsg =
(IotaMessage)intent.getSerializableExtra(BindService.BIND_BROADCAST_MSG_IE_IOTAMSG);
    int result = iotaMsg.getUint(BindService.BIND_IE_RESULT, 0);
    String deviceId = iotaMsg.getString(BindService.BIND_IE_DEVICEID);
    String Secret = iotaMsg.getString(BindService.BIND_IE_DEVICESECRET);
    String Appid = iotaMsg.getString(BindService.BIND_IE_APPID);
    String haAddr = iotaMsg.getString(BindService.BIND_IE_HA_ADDR);
    String lvsAddr = iotaMsg.getString(BindService.BIND_IE_LVS_ADDR);
    return;
    }
};
mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);
IntentFilter filterBind = new IntentFilter(BindService.TOPIC_BINDDEVICE_RSP);
mLocalBroadcastManager.registerReceiver(mBindRsp, filterBind);
```

5.2.2.5 设备解绑定

接口功能

注册设备解绑定接收广播来接收处理平台下发的直连设备解绑定命令，开发者收到该广播后需要删除直连设备的配置信息并且释放所有资源，下一次重启后需要重新进行绑定。

接口描述

```
HubService.TOPIC_UNBINDDEVICE
```

参数说明

无

返回结果

无

示例

```
BroadcastReceiver mUnbindRsp;
mUnbindRsp = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        //Delete config file, free resource
        return;
    }
};
mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);
IntentFilter filterUnbind = new IntentFilter(HubService.TOPIC_UNBINDDEVICE);
mLocalBroadcastManager.registerReceiver(mUnbindRsp, filterUnbind);
```

5.2.2.6 登录配置

接口功能

在登录前配置登录所需要的参数

接口描述

```
public static boolean setConfig(int key, String value)
```

接口所属类

LoginConfig

参数说明

字段	必选/可选	类型	描述
key	必选	int	设备登录的配置项 设备ID：LoginConfig.LOGIN_CONFIG_DEVICEID AppId：LoginConfig.LOGIN_CONFIG_APPID 密码：LoginConfig.LOGIN_CONFIG_SECRET HTTP地址：LoginConfig.LOGIN_CONFIG_IOCM_ADDR HTTP端口：LoginConfig.LOGIN_CONFIG_IOCM_PORT MQTT地址：LoginConfig.LOGIN_CONFIG_MQTT_ADDR MQTT端口：LoginConfig.LOGIN_CONFIG_MQTT_PORT
value	必选	String	设置的值 设备ID：从绑定成功的广播中获取 AppId：从绑定成功的广播中获取 密码：从绑定成功的广播中获取 HTTP地址：Agent Lite对接平台地址 HTTP端口：8943 MQTT地址：Agent Lite对接平台地址 MQTT端口：8883

接口返回值

返回值	描述
true	成功
false	失败

返回结果

无

示例

```
//保存绑定响应消息携带的参数
private void saveBindPara(IotaMessage iotaMsg) {
    LogUtil.i(this, TAG, "saveBindParaAndGotoLogin");
    String appId = iotaMsg.getString(BindService.BIND_IE_APPID);
    String deviceId = iotaMsg.getString(BindService.BIND_IE_DEVICEID);
    String secret = iotaMsg.getString(BindService.BIND_IE_DEVICESECRET);
    String haAddress = AgentLiteUtil.get(ConfigName.platformIP);

    saveGatewayInfo(appId, deviceId, secret, haAddress, null);
}

//设置登录配置
private void configLoginPara() {
    LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_DEVICEID, GatewayInfo.getDeviceID());
    LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_APPID, GatewayInfo.getAppID());
    LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_SECRET, GatewayInfo.getSecret());
    LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_IOCM_ADDR, GatewayInfo.getHaAddress());
    LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_IOCM_PORT, "8943");
    LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_MQTT_ADDR, GatewayInfo.getHaAddress());
    LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_MQTT_PORT, "8883");
}
```

5.2.2.7 设备登录

接口功能

设备在第一次绑定后，或者在设备重启后需要进行登录的流程。

接口描述

```
public static boolean login();
```

接口所属类

LoginService

接口返回值

返回值	描述
true	成功
false	失败

说明

此返回值是调用接口的同步返回结果，返回true只是说明接口调用成功，并不说明登陆成功，登陆成功需要收到LoginService.TOPIC_LOGIN_CONNECTED广播。登录前通过参数配置接口（**LoginConfig.setConfig**）传入所需的登录信息。

返回结果

广播名称	广播参数	成员	描述
TOPIC_LOGIN_CONNECTED	IotaMessage对象（使用intent.getSerializableExtra(LoginService.LOGIN_BROADCAST_MSG_IE_IOTAMSG)方法获取）	无	登录成功或重连成功
TOPIC_LOGIN_DISCONNECT	IotaMessage对象（使用intent.getSerializableExtra(LoginService.LOGIN_BROADCAST_MSG_IE_IOTAMSG)方法获取）	LOGIN_IE_REASON	登录或重连失败原因
		LOGIN_IE_STATUS	登录或重连状态

示例

```
//配置登陆参数
LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_DEVICEID, “deviceId”);
LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_APPID, “appId”);
LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_SECRET, “passWord”);
LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_IOCM_ADDR, “haAddr”);
LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_IOCM_PORT, “8943”);
LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_MQTT_ADDR, “haAddr”);
LoginConfig.setConfig(LoginConfig.LOGIN_CONFIG_MQTT_PORT, “8883”);

//调用登录接口
LoginService.login();
```

然后等待Agent Lite的连接状态广播

建议：

1、 在连接成功的处理函数中进行非直连设备状态上报的处理，并且将缓存的上报数据进行上报。

2、 在连接断开的处理函数中记录设备断开状态，之后如果有非直连设备上报数据，需要进行缓存，等到连接成功后再进行上报。

```
//接收登录成功响应
BroadcastReceiver mReceiverConnect;
mReceiverConnect = new BroadcastReceiver() {
   @Override
   public void onReceive(Context context, Intent intent) {
    //获取IotMessage
IotaMessage iotaMsg = (IotaMessage)intent.getSerializableExtra(LoginService.
LOGIN_BROADCAST_MSG_IE_IOTAMSG);
//从回调返回的消息句柄中获取当前系统状态
int status = uspMsg.getInt(LoginService.LOGIN_IE_STATUS, 0);
//update device states
...
return true;
}
}
```

```
mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);
IntentFilter filterCon= new IntentFilter(LoginService.TOPIC_LOGIN_CONNECTED);

mLocalBroadcastManager.registerReceiver(mReceiverConnect, filterCon);
//接收登录失败响应
BroadcastReceiver mReceiverDisconnect;
mReceiverDisconnect = new BroadcastReceiver() {
   @Override
   public void onReceive(Context context, Intent intent) {
    //获取IotMessage
IotaMessage iotaMsg = (IotaMessage)intent.getSerializableExtra(LoginService.
LOGIN_BROADCAST_MSG_IE_IOTAMSG);
//获取响应的错误码
int reason = iotaMsg.getInt(LoginService.LOGIN_IE_REASON, 0);
//stop reporting data
...
return true;
}
}
mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);
IntentFilter filterDiscon= new IntentFilter(LoginService.TOPIC_LOGIN_DISCONNECTED);

mLocalBroadcastManager.registerReceiver(mReceiverDisconnect, filterDiscon);
```

设备登录后，表示该设备已经成功的连接到IoT联接管理平台。

连接成功后，如果因为网络或服务器原因导致连接断开，Agent Lite会自动尝试重新连接，并将实时状态通过这两个广播上报给第三方应用。

5.2.2.8 设备登出

接口功能

断开与IOT平台的连接

接口描述

```
public static boolean logout()
```

接口所属类

LoginService

参数说明

无

接口返回值

返回值	描述
true	成功
false	失败

返回结果

无

示例

```
LoginService.logout();
```

5.2.3 网关管理非直连设备

当开发设备为网关设备时，设备需要管理所有非直连设备（传感器设备）的接入与删除，并且记录这些设备ID与对应设备的映射关系。

5.2.3.1 设备添加

接口功能

当有新设备接入网关后，通过调用设备添加接口将非直连设备接入IoT联接管理平台，并且获得平台分配的唯一设备逻辑ID。

接口描述

```
public static boolean addDevice(int cookie, IotaDeviceInfo deviceInfo);
```

接口所属类

HubService

参数说明

字段	必选/可选	类型	描述
cookie	可选	int	Cookie有效值为1-65535
deviceInfo	必选	**IotaDeviceInfo**类	设备信息

接口返回值

返回值	描述
true	成功
false	失败

说明

此返回值是调用接口的同步返回结果，返回true只是说明接口调用成功，并不说明添加成功，添加成功需要收到TOPIC_ADDDEV_RSP广播。

返回结果

广播名称	广播参数	成员	描述
TOPIC_ADDDEV_RSP	IotaMessage（使用intent.getSerializableExtra(HubService.HUB_BROADCAST_IE_IOTAMSG)方法获取）	HUB_IE_RESULT	添加设备结果
		HUB_IE_DEVICEID	设备ID，如果添加成功则返回设备ID
		HUB_IE_COOKIE	Cookie有效值为1~65535

示例

```
//开发者调用该接口进行设备添加
HubService.addDevice(29011, new IotaDeviceInfo("nodeId", "manufacturerId", "deviceType", "model",
"protocolType"));
```

结果处理：

```
//java code
// 开发者注册该函数进行设备添加后的处理
BroadcastReceiver mAdddeviceRsp;
mAdddeviceRsp = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        //Do Something
        IotaMessage iotaMsg = (IotaMessage)intent.getSerializableExtra(HubService.
HUB_BROADCAST_IE_IOTAMSG);
        int result = iotaMsg.getUint(HubService.HUB_IE_RESULT, 0);
String deviceId = iotaMsg.getString(HubService.HUB_IE_DEVICEID);
int cookie = iotaMsg.getUint(HubService.HUB_IE_COOKIE, 0);
return;
}
};
mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);
IntentFilter filterAddDev = new IntentFilter(HubService.TOPIC_ADDDEV_RSP);
mLocalBroadcastManager.registerReceiver(mAdddeviceRsp, filterAddDev);
```

5.2.3.2 设备状态更新

接口功能

通过该接口更新设备的状态信息，包括直连设备与所管理的非直连设备。设备离线上线均可通过该接口刷新设备状态信息。

说明

直连设备状态通过设备的登录状态进行管理，当直连设备连接断开则表示设备离线，当直连设备连接或重连成功，则表示设备上线，无需通过该接口进行刷新。**故建议开发者使用该接口刷新非直连设备的状态。**

接口描述

```
public static boolean updateDeviceStatus(int cookie, String deviceId, String status, String
statusDetail)
```

接口所属类

HubService

参数说明

字段	必选/可选	类型	描述	
cookie	可选	int	Cookie有效值为1-65535	
deviceId	必选	String	设备Id	
status	必选	String	设备状态	在线: ONLINE
				离线: OFFLINE
statusDetail	必选	String	设备状态详细信息	无：NONE
				配置等待：CONFIGURATION_PENDING
				通信错误：COMMUNICATION_ERROR
				配置错误：CONFIGURATION_ERROR
				桥接器离线 BRIDGE_OFFLINE
				固件升级 FIRMWARE_UPDATING
				循环任务 DUTY_CYCLE
				未激活 NOT_ACTIVE

接口返回值

返回值	描述
true	成功
false	失败

返回结果

广播名称↵	广播参数↵	成员↵	描述↵
TOPIC_DEVSTATUS_UPDATA_RSP	IotaMessage对象（使用intent.getSerializableExtra(HubService.HUB_BROADCAST_IE_IOTAMSG)方法获取）	HUB_IE_RESULT	设备状态更新结果
		HUB_IE_DEVICEID	设备ID

示例

```
HubService.deviceStatusUpdate(0, deviceId, “ONLINE”, “NONE”);
```

然后等待命令执行结果

```
// 开发者注册该函数进行状态更新后的处理
BroadcastReceiver mReceiverDevStatus;
mReceiverDevStatus = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
    //获取IotMessage
IotaMessage iotaMsg =
(IotaMessage)intent.getSerializableExtra(HubService.HUB_BROADCAST_IE_IOTAMSG);
//获取响应的错误码
String result = iotaMsg.getString(HubService.HUB_IE_RESULT);
String deviceId = iotaMsg.getString(HubService.HUB_IE_DEVICEID);
...
return true;
}
}
mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);
IntentFilter filterDiscon= new IntentFilter(HubService. TOPIC_DEVSTATUS_UPDATA_RSP);

mLocalBroadcastManager.registerReceiver(mReceiverDevStatus, filterDiscon);
```

5.2.3.3 设备删除

接口功能

当有新设备需要从网关移除时，通过调用设备删除接口将非直连设备从IoT联接管理平台删除。

接口描述

```
public static boolean rmvDevice(int cookie, String deviceId)
```

接口所属类

HubService

参数说明

字段	必选/可选	类型	描述
cookie	可选	int	Cookie有效值 1-65535
deviceId	必选	String	设备Id

接口返回值

返回值	描述
true	成功
false	失败

说明

此返回值是调用接口的同步返回结果，返回true只是说明接口调用成功，并不说明删除成功，添加成功需要收到TOPIC_RMVDEV_RSP广播。

返回结果

广播名称	广播参数	成员	描述
TOPIC_ RMVDEV_RSP	IotaMessage （使用intent.getSerializableExtra(HubService.HUB_BROADCAST_IE_IOTAMSG)方法获取）	HUB_IE_RESULT	删除设备结果
		HUB_IE_COOKIE	Cookie有效值为1~65535

示例

```
// 开发者调用该接口进行设备删除
HubService.rmvDevice(122, deviceId);
```

结果处理：

```
//java code
BroadcastReceiver mRmvdeviceRsp;
mRmvdeviceRsp = new BroadcastReceiver() {
@Override
 public void onReceive(Context context, Intent intent) {
    //Do Something
        IotaMessage iotaMsg = (IotaMessage)intent.getSerializableExtra(HubService.
HUB_BROADCAST_IE_IOTAMSG);
        int result = iotaMsg.getUint(HubService.HUB_IE_RESULT, 0);
int cookie = iotaMsg.getUint(HubService.HUB_IE_COOKIE, 0);
return;
}
};
```

```
mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);
IntentFilter filterRmvDev = new IntentFilter(HubService.TOPIC_RMVDEV_RSP);
mLocalBroadcastManager.registerReceiver(mDeldeviceRsp, filterRmvDev);
```

5.2.4 设备服务数据上报

接口功能

当直连设备（网关）有数据需要上报或者非直连设备上报数据到网关时，网关需要调用设备服务数据上报接口将数据上报到IoT联接管理平台。

接口描述

```
public static boolean dataReport(int cookie, String requstId, String deviceId, String serviceId,
String serviceProperties);
```

接口所属类

DataTransService

参数说明

字段	必选/可选	类型	描述
cookie	可选	int	Cookie有效值1-65535
requstId	条件必选	String	请求ID，匹配之前平台下发的服务命令。可以从**设备命令接收**的广播中获取requestId。 **主动数据上报：**requestId为NULL **命令结果上报：**当上报的数据匹配到某一次命令请求时，需要填写此次命令请求的请求ID。
deviceId	必选	String	设备ID
serviceId	必选	String	服务ID
serviceProperties	必选	String	服务属性

接口返回值

返回值	描述
true	成功
false	失败

说明

此返回值是调用接口的同步返回结果，返回true只是说明接口调用成功，并不说明删除成功，添加成功需要收到TOPIC_DATA_REPORT_RSP广播

返回结果

广播名称	广播参数	成员	描述
TOPIC_DATA_REPORT_RSP	IotaMessage（使用intent.getSerializableExtra(DataTransService.DATATRANS_BROADCAST_IE_IOTAMSG)方法获取）	DATATRANS_IE_RESULT	数据上报结果
		DATATRANS_IE_COOKIE	Cookie有效值为1~65535

示例

用户根据Profile格式使用Json组件拼装服务属性的内容(serviceProperties);

```
DataTransService.dataReport(1211, NULL, "xxxx_xxxx_xxxx_xxxx", "DoorWindow", "{\"status\":\
"OPEN\"}");
```

数据上报结果接收

//开发者注册该函数进行设备服务数据上报后的处理

```
BroadcastReceiver mReportDataRsp;
mReportDataRsp = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
//Do Something
IotaMessage iotaMsg = (IotaMessage)intent.getSerializableExtra(DataTransService.
DATATRANS_BROADCAST_IE_IOTAMSG);
int cookie = iotaMsg.getUint(DataTransService.DATATRANS_IE_COOKIE, 0);
int ret = iotaMsg.getUint(DataTransService.DATATRANS_IE_RESULT, 0);
return;
}
};
mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);
IntentFilter filterReportData
= new IntentFilter(DataTransService.TOPIC_DATA_REPORT_RSP);
mLocalBroadcastManager.registerReceiver(mReportDataRsp, filterReportData);
```

5.2.5 设备命令接收

接口功能

注册设备命令接收广播来接收处理平台下发的控制命令。

接口描述

```
DataTransService.TOPIC_COMMAND_RECEIVE
```

参数说明

无

返回结果

广播名称	广播参数	成员	描述
TOPIC_COMMAND_RECEIVE	IotaMessage（使用intent.getSerializableExtra(DataTransService.DATATRANS_BROADCAST_IE_IOTAMSG)方法获取）	DATATRANS_IE_REQUESTID	请求ID
		DATATRANS_IE_DEVICEID	设备逻辑ID，在设备添加时由平台分配
		DATATRANS_IE_SERVICEID	服务ID
		DATATRANS_IE_METHOD	服务方法
		DATATRANS_IE_CMDCONTENT	命令内容

示例

```
// 开发者注册该函数进行命令接收的处理
BroadcastReceiver mReceiveCmd;
mReceiveCmd = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
    //Do Something
        IotaMessage iotaMsg = (IotaMessage)intent.getSerializableExtra(DataTransService.
DATATRANS_BROADCAST_IE_IOTAMSG);
String requstId = iotaMsg.getString(DataTransService.DATATRANS_IE_REQUESTID);
String deviceId = iotaMsg.getString(DataTransService.DATATRANS_IE_DEVICEID);
String serviceId = iotaMsg.getString(DataTransService.DATATRANS_IE_SERVICEID);
String method = iotaMsg.getString(DataTransService.DATATRANS_IE_METHOD);
String cmdContent = iotaMsg.getString(DataTransService.DATATRANS_IE_CMDCONTENT);
if (serviceId.equals(“switch”))
{
     //根据Proflie定义的命令参数，使用Json组件解析cmdContent
     //Send command to Switch
}

return;
}
};
mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);
IntentFilter filterReceiveCmd
 = new IntentFilter(DataTransService.TOPIC_COMMAND_RECEIVE);
mLocalBroadcastManager.registerReceiver(mReceiveCmd, filterReceiveCmd);
```

5.3 常用数据结构定义

5.3.1 IotaDeviceInfo 类说明

字段	必选/可选	类型	描述
nodeId	必选	String	关键参数，对接平台的网关下设备唯一标识，设备填写，平台用于判重

字段	必选/可选	类型	描述
name	可选	String	设备名称
description	可选	String	设备描述
manufacturerId	必选	String	厂商ID
manufacturerName	可选	String	厂商名
mac	可选	String	设备MAC地址
location	可选	String	设备的位置
deviceType	必选	String	设备类型
model	必选	String	型号 直连设备：与Profile中定义的model保持一致即可 Z-Wave设备： ProductType + ProductId，格式（使用16 进制）： XXXX-XXXX 补0对齐，如001A-0A12 其他协议再定
swVersion	可选	String	软件版本， Z-Wave 主次版本号 主版本号.次版本号如：1.1
fwVersion	可选	String	固件版本
hwVersion	可选	String	硬件版本
protocolType	必选	String	协议类型 Z-Wave
bridgeId	可选	String	表示设备通过哪个Bridge接入平台
status	可选	String	表示设备是否在线， ONLINE 在线 OFFLINE 离线

字段	必选/可选	类型	描述
statusDetail	可选	String	状态详情，如果pcStatus不为空，则该参数必选。 参数值： 无：NONE 配置等待：CONFIGURATION_PENDING 通信错误：COMMUNICATION_ERROR 配置错误：CONFIGURATION_ERROR 桥接器离线：BRIDGE_OFFLINE 固件升级：FIRMWARE_UPDATING 循环任务：DUTY_CYCLE 未激活：NOT_ACTIVE
mute	可选	String	表示设备是否被屏蔽： TRUE FALSE

5.3.2 BIND_IE_RESULT 信元参数说明

枚举项	枚举值	类型	描述
BIND_RESULT_SUCCESS	0	NA	绑定成功
BIND_RESULT_DEV_NOT_BIND	1	NA	未扫码
BIND_RESULT_VERIFYCODE_EXPIRED	2	NA	验证码过期
BIND_RESULT_FAILED	255	NA	其余失败

5.3.3 LOGIN_IE_REASON 信元参数说明

枚举项	枚举值	类型	描述
LOGIN_REASON_NULL	0	NA	无原因
LGN_REASON_CONNCET_ERR	1	NA	连接失败

枚举项	枚举值	类型	描述
LOGIN_REASON_SERVER_BUSY	2	NA	服务器忙
LOGIN_REASON_AUTH_FAILED	3	NA	鉴权失败、开发者需要停止重新尝试登陆
LOGIN_REASON_NET_UNAVAILABLE	4	NA	网络不可用
LOGIN_REASON_DEVICE_NOEXIST	5	NA	设备不存在、开发者需要停止重新尝试登陆
LOGIN_REASON_DEVICE_RMVED	6	**NA**	设备已删除、开发者需要停止重新尝试登陆
LOGIN_REASON_UNKNOWN	7	NA	未知原因

5.3.4 HUB_IE_RESULT 参数枚举说明

枚举项	枚举值	类型	描述
HUB_RESULT_SUCCESS	0	NA	添加/删除执行成功
HUB_RESULT_DEVICE_EXIST	1	NA	设备已存在
HUB_RESULT_DEVICE_NOTEXIST	2	NA	设备不存在
HUB_RESULT_DEVICE_FAILED	255	NA	执行失败

6 Agent Lite API 参考(C)

6.1 开发者必读

1. 概述

华为IoT Agent Lite在智慧家庭、工业物联网、车联网等领域为智能设备提供了标准接入华为IoT联接管理平台的能力。主要面向IPC、轻量级网关、工业网关、车机等计算能力较强的终端/网关设备。

2. 接口全景图

集成开发者通过全景图了解到：Agent Lite提供了哪些功能？每个功能包含哪些接口？接口之间的逻辑关系如何？从而更快速的找到正确的接口来实现具体业务。

功能	接口	说明
直连设备接入	IOTA_Init	初始化Agent Lite模块资源
	IOTA_Destroy	释放Agent Lite模块资源
	IOTA_SetConfig	相关参数配置
	IOTA_Bind	设备绑定
	IOTA_Login	设备登录
	IOTA_DeviceStatusUpdate	设备状态更新
网关管理非直连设备	IOTA_HubDeviceAdd	设备添加
	IOTA_HubDeviceRemove	设备删除

功能	接口	说明
设备服务数据上报	IOTA_ServiceDataReport	设备服务数据上报
设备命令接收	IOTA_TOPIC_SERVICE_COMMAND_ RECEIVE/{deviceId}	设备服务命令接收

6.2 接口列表

Agent Lite对外提供的接口主要包括广播机制、直连设备的接入、非直连设备的添加和删除、设备数据的上报、设备命令接收以及Json接口。

6.2.1 广播机制介绍

Agent Lite提供了一套广播机制给第三方开发者，用来接收Agent Lite上报的消息。

订阅广播：

```
HW_BroadCastReg(HW_CHAR *pcTopic ,PFN_HW_BROADCAST_RECV pfnReceiver)
```

广播接收处理函数原型：

```
(*PFN_HW_BROADCAST_RECV)(HW_UINT uiCookie, HW_MSG *pstMsg);
```

此处uiCookie对应于接口中传入的uiCookie，用来匹配业务的请求与响应；如接口中无uiCookie参数，或传入的是无效值，则广播中该参数无意义。

取消订阅广播：

```
HW_BroadCastUnreg(HW_CHAR *pcTopic, PFN_HW_BROADCAST_RECV pfnReceiver);
```

从 pstMsg 获取数据的函数：

获取字符串数据：

```
HW_MsgGetStr(HW_MSG pstMsg, HW_UINT uiTag)
```

获取无符号整形数据：

```
HW_MsgGetUint(HW_MSG pstMsg, HW_UINT uiTag, HW_UINT uiDefault)
```

获取字节数组数据：

```
HW_MsgGetByteArray(HW_MSG pstMsg, HW_UINT uiTag)
```

6.2.2 直连设备接入

6.2.2.1 初始化 Agent Lite 资源

接口功能

初始化Agent Lite资源。

接口描述

```
HW_INT IOTA_Init(HW_CHAR *pcWorkPath, HW_CHAR *pcLogPath);
```

参数说明

字段	必选/可选	类型	描述
pcWorkPath	必选	String	Agent Lite工作路径，用于存放Agent Lite的配置文件与生产的临时文件。
pcLogPath	可选	String	日志路径（若日志路径为空则日志写在工作路径中）

接口返回值

参见**函数标准返回值**

示例

```
// 开发者调用该接口初始化Agent Lite资源
IOTA_Init(“/usr/data”,HW_NULL)
```

6.2.2.2 释放 Agent Lite 资源

接口功能

调用此函数，Agent Lite会释放申请的所有动态资源（内存、线程等等）。

接口描述

```
IOTA_VOID IOTA_Destroy();
```

接口返回值

参见**函数标准返回值**

示例

```
// 开发者调用该接口销毁Agent Lite资源
IOTA_Destroy();
```

6.2.2.3 参数设置

接口功能

配置Agent Lite相关参数。

接口描述

```
HW_INT IOTA_ConfigSetStr(HW_INT iItem, HW_CHAR *pValue)
HW_INT IOTA_ConfigSetUint(HW_INT iItem, HW_UINT uiValue)
```

参数说明

字段	必选/可选	类型	描述
iItem	必选	枚举	设备登陆所需的配置项（详见EN_IOTA_CFG_TYPE说明）
pValue	必选	String/int	设置的值

接口返回值

参见函数标准返回值

示例

```
// 开发者调用该接口设置参数
IOTA_ConfigSetStr (EN_IOTA_CONFIG_IOCM_ADDR, “10.0.0.1”);
IOTA_ConfigSetUint(EN_IOTA_CFG_IOCM_PORT, 8943);
```

6.2.2.4 设备绑定

接口功能

设备第一次接入IoT联接管理平台时需要进行绑定操作，上层应用通过调用该接口传入设备序列号或者MAC地址以及设备信息来绑定到IoT联接管理平台。

前提条件

在绑定前需要调用**IOTA_ConfigSetXXX接口**设置绑定服务器IP与端口（IoCM服务器地址与端口，Agent Lite会配置默认端口8943），对应参数配置为EN_IOTA_CFG_IOCM_ADDR和EN_IOTA_CFG_IOCM_PORT。

说明

设备绑定是指设备第一次接入IoT平台的过程，需要开发者先在IoT平台注册直连设备，之后在设备上发起绑定操作，将设备绑定到IoT平台上。如果未在IoT平台注册该设备，则绑定操作会失败，Agent Lite将会等待一段时间继续尝试。

接口描述

```
HW_INT IOTA_Bind(HW_CHAR *pcVerifyCode, ST_IOTA_DEVICE_INFO *pstInfo)
```

参数说明

字段	必选/可选	类型	描述
pcVerifyCode	必选	String	设备绑定验证码
pstInfo	必选	ST_IOTA_DEVICE_INFO结构体	设备信息

接口返回值

参见函数标准返回值

说明

此返回值是调用接口的同步返回结果，返回0只是说明接口调用成功，并不说明绑定成功，绑定成功需要收到IOTA_TOPIC_BIND_RSP广播。

注意

该设备未绑定时，该接口会自动尝试重新绑定，如果绑定失败，则30秒后继续进行重试，如果重试超过5次（总计尝试超过6次），则返回失败，不再进行重试。如果想要重新发起绑定，建议让用户重启设备。开发者无需进行重试，如果需要停止重试，调用Agent Lite销毁接口即可。同时请勿短时间内多次调用该接口。

当前绑定流程的重试策略为，如果绑定失败，则30秒后继续进行重试，如果重试超过5次（总计尝试超过6次），则返回失败，不再进行重试。如果想要重新发起绑定，建议让用户重启设备。

返回结果

广播名称	广播参数	成员	描述
IOTA_TOPIC_BIND_RSP	HW_MSG对象	EN_IOTA_BIND_IE_TYPE	绑定返回结果（详见枚举EN_IOTA_BIND_IE_TYPE）

注意事项

如果接入设备是网关设备，则DeviceType填写为“Gateway”,如果为传感器设备直连，则填写为对应的设备类型。

示例

```
//开发者调用该接口进行设备绑定
ST_HW_DEVICE_INFO stDeviceInfo
stDeviceInfo.pcNodeId = “SN Number”;
stDeviceInfo.pcManufacturerId = “Huawei”;
stDeviceInfo.pcDeviceType = “Gateway”;
stDeviceInfo.pcModel = “HW_GW101”;
stDeviceInfo.pcProtocolType = “HuaweiM2M”;

IOTA_Bind(“SN Number”, &stDeviceInfo)
```

当设备成功绑定之后，Agent Lite会返回给UI如下几个参数，需要UI进行持久化存储，设备登录前需要提前进行配置

```
//注册广播接收处理函数
HW_BroadCastReg(“IOTA_TOPIC_BIND_RSP”, Device_RegResultHandler);
```

```
// 开发者注册该函数处理绑定结果
HW_INT Device_RegResultHandler(HW_UINT uiCookie, HW_MSG pstMsg)
{
HW_CHAR *pcDeviceId;
HW_CHAR *pcDeviceSecret;
HW_CHAR *pcAppId;
HW_CHAR *pcIoCMServerAddr;
```

```
HW_UINT uiIoCMServerPort;
HW_CHAR *pcMqttServerAddr;
HW_UINT uiMqttServerPort;

If (HW_SUCCESS != HW_MsgGetUint(pstMsg, EN_IOTA_BIND_IE_RESULT, 0))
{
Return 0;
}

pcDeviceId = HW_MsgGetStr(pstMsg, EN_IOTA_BIND_IE_DEVICEID);
pcDeviceSecret = HW_MsgGetStr(pstMsg, EN_IOTA_BIND_IE_DEVICESECRET);
pcAppId = HW_MsgGetStr(pstMsg, EN_IOTA_ BIND_IE_IOCM_ADDR );
pcIoCMServerAddr = HW_MsgGetStr(pstMsg, EN_IOTA_ BIND_IE_IOCM_ADDR );
uiIoCMServerPort = HW_MsgGetUint(pstMsg, EN_IOTA_BIND_IE_IOCM_PORT, 0);
pcMqttServerAddr = HW_MsgGetStr(pstMsg, EN_IOTA_ BIND_IE_IOCM_ADDR );
uiMqttServerPort = HW_MsgGetUint(pstMsg, EN_IOTA_BIND_IE_IOCM_PORT, 0);

Config_save(“DeviceId”,pcDeviceId);
Config_save(“DeviceSecret”,pcDeviceSecret);
Config_save(“AppId”,pcAppId);
Config_save(“IoCMAddr”,pcIoCMServerAddr);
Config_save(“IoCMPort”,pcIoCMServerPort);
Config_save(“MqttAddr”,pcMqttServerAddr);
Config_save(“MqttPort”,pcMqttServerPort);

return 0;
}
```

6.2.2.5 设备解绑定

接口功能

注册设备解绑定接收广播来接收处理平台下发的直连设备解绑定命令，开发者收到该广播后需要删除直连设备的配置信息并且释放所有资源，下一次重启后需要重新进行绑定。

接口描述

```
IOTA_TOPIC_CMD_UNBIND_RECEIVE
```

参数说明

无

返回结果

无

示例:

```
// 开发者注册该函数进行解绑定直连设备的处理
HW_INT Gateway_UnbindRecvtHandler(HW_UINT uiCookie, HW_MSG pstMsg)
{
// Delete Config file，free resources.
return 0;
}
//初始化时进行注册该函数
HW_BroadCastReg(“IOTA_TOPIC_CMD_UNBIND_RECEIVE”, Gateway_UnbindRecvtHandler);
```

6.2.2.6 设备登录

接口功能

设备在第一次绑定后，或者在设备重启后需要进行登录的流程。

接口描述

```
HW_INT IOTA_Login()
```

接口返回值

参见函数标准返回值

说明

此返回值是调用接口的同步返回结果，返回0只是说明接口调用成功，并不说明登陆成功，登陆成功需要收到IOTA_TOPIC_CONNECTED_NTY广播。登录前通过参数配置接口（IOTA_SetConfig）传入所需的登录信息。

必要参数配置:

EN_IOTA_CFG_DEVICEID

EN_IOTA_CFG_DEVICESECRET

EN_IOTA_CFG_APPID

EN_IOTA_CFG_IOCM_ADDR

EN_IOTA_CFG_IOCM_PORT

EN_IOTA_CFG_MQTT_ADDR

EN_IOTA_CFG_MQTT_PORT

之后调用登录接口函数进行登录：

HW_INT IOTA_Login();

返回结果

广播名称	广播参数	成员	描述
IOTA_TOPIC_CONNECTED_NTY	HW_MSG对象	无	登录成功或重连成功
IOTA_TOPIC_DISCONNECT_NTY	HW_MSG对象	EN_IOTA_LGN_IE_TYPE	登陆失败或连接断开

示例：

```
Config_Get(“DeviceId”,pcDeviceId);
Config_Get(“DeviceSecret”,pcDeviceSecret);
Config_Get(“AppId”,pcAppId);
Config_Get(“HAAddr”,pcHAServerAddr);
Config_Get(“LVSAddr”,pcLVSServerAddr);

IOTA_SetConfig(EN_IOTA_CFG_DEVICEID, pcDeviceId);
IOTA_SetConfig(EN_IOTA_CFG_DEVICESECRET, pcDeviceSecret);
IOTA_SetConfig(EN_IOTA_CFG_APPID, pcAppId);
IOTA_SetConfig(EN_IOTA_CFG_HA_ADDR, pcHAServerAddr);
IOTA_SetConfig(EN_IOTA_CFG_LVS_ADDR, pcLVSServerAddr);

IOTA_Login();
```

然后等待Agent Lite的连接状态广播

需要提前实现连接状态通知广播接收处理函数，建议：

1. 对于网关设备，在连接成功的处理函数中需要进行非直连设备状态上报的处理，并且将缓存的所有上报数据进行上报。
2. 在连接断开的处理函数中记录设备断开状态，之后如果有数据上报，需要进行缓存，等到连接成功后再进行上报。

```
// 开发者注册该函数进行连接成功后的处理
HW_INT Device_ConnectedHandler(HW_UINT uiCookie, HW_MSG pstMsg)
{
//update device states
//send buffer data

return 0;
}
```

```
// 开发者注册该函数进行连接失败后的处理
HW_INT Device_DisconnectHandler(HW_UINT uiCookie, HW_MSG pstMsg)
{
//stop reporting data
return 0;
}
```

```
//绑定广播接收处理函数
HW_BroadCastReg(“IOTA_TOPIC_CONNECTED_NTY”, Device_ConnectedHandler);
HW_BroadCastReg(“IOTA_TOPIC_DISCONNECT_NTY”, Device_DisconnectHandler);
```

设备登录后，表示该设备已经成功的连接到IoT联接管理平台。

连接成功后，如果因为网络或服务器原因导致连接断开，Agent Lite会自动尝试重新连接，并将实时状态通过这两个广播上报给第三方应用。

6.2.3 网关管理非直连设备

当开发设备为网关设备时，设备需要管理所有非直连设备（传感器设备）的接入与删除，并且记录这些设备ID与对应设备的映射关系。

6.2.3.1 设备添加

接口功能

当有新设备接入网关后，通过调用设备添加接口将非直连设备接入IoT联接管理平台，并且获得平台分配的唯一设备逻辑ID。

接口描述

```
HW_INT IOTA_HubDeviceAdd(HW_UINT uiCookie, ST_IOTA_DEVICE_INFO *pstDeviceInfo);
```

参数说明

字段	必选/可选	类型	描述
uiCookie	可选	HW_UINT	Cookie有效值为1-65535
pstDeviceInfo	必选	ST_IOTA_DEVICE_INFO结构体	设备信息

接口返回值

参见**函数标准返回值**

返回结果

广播名称	广播参数	成员	描述
IOTA_TOPIC_HUB_ADDDEV_RSP	HW_MSG对象	EN_IOTA_HUB_IE_TYPE	返回设备添加结果，如果添加成功则返回设备ID

示例

```
// 开发者调用该接口进行设备添加
ST_IOTA_DEVICE_INFO stDeviceInfo
stDeviceInfo.pcNodeId = “SN Number”;
stDeviceInfo.pcManufacturerId = “Huawei”;
stDeviceInfo.pcDeviceType = “Camera”;
stDeviceInfo.pcModel = “HW_CAM101”;
stDeviceInfo.pcProtocolType = “ONVIF”;
```

IOTA_HubDeviceAdd(29011, &stDeviceInfo)

结果处理：

```
// 开发者注册该函数进行设备添加后的处理
HW_INT Device_AddResultHandler(HW_UINT uiCookie, HW_MSG pstMsg)
{
uiResult = HW_MsgGetUint(pstMsg, EN_IOTA_HUB_IE_RESULT);
if (EN_IOTA_HUB_RESULT_SUCCESS != uiResult)
{
// retry with uiCookie
return 0;
}

return 0;
}
```

```
//绑定广播接收处理函数
HW_BroadCastReg(“IOTA_TOPIC_HUB_ADDDEV_RSP”, Device_AddResultHandler);
```

6.2.3.2 设备状态更新

接口功能

通过该接口更新设备的状态信息，包括直连设备与所管理的非直连设备。设备离线上线均可通过该接口刷新设备状态信息。

说明

直连设备状态通过设备的登录状态进行管理，当直连设备连接断开则表示设备离线，当直连设备连接或重连成功，则表示设备上线，无需通过该接口进行刷新。故建议开发者使用该接口刷新非直连设备的状态。

接口描述

```
HW_INT IOTA_DeviceStatusUpdate(HW_UINT uiCookie, HW_CHAR *pcDeviceId, HW_CHAR pcStatus, HW_CHAR pcStatusDetail)
```

参数说明

字段	必选/可选	类型	描述	
uiCookie	可选	HW_UINT	Cookie有效值为1-65535	
pcDeviceId	必选	HW_CHAR	设备Id	
pcStatus	必选	HW_CHAR	设备状态	在线: ONLINE
				离线: OFFLINE
pcStatusDetail	必选	HW_CHAR	设备状态详细信息	无：NONE
				配置等待：CONFIGURATION_PENDING
				通信错误：COMMUNICATION_ERROR
				配置错误：CONFIGURATION_ERROR
				桥接器离线 BRIDGE_OFFLINE
				固件升级 FIRMWARE_UPDATING
				循环任务 DUTY_CYCLE
				未激活 NOT_ACTIVE

接口返回值

参见**函数标准返回值**

返回结果

广播名称	广播参数	成员	描述
IOTA_TOPIC_DEVUPDATE_RSP/{deviceId}	HW_MSG对象	无	设备状态更新结果

示例：

```
HW_CHAR *pcDeviceId = stDevice.pcDeviceId;

IOTA_DeviceStatusUpdate(0, pcDeviceId, “ONLINE” , “NONE”);
```

然后等待命令执行结果

```
// 开发者注册该函数进行状态更新后的处理
HW_INT Device_StatusUpdateHandler(HW_UINT uiCookie, HW_MSG pstMsg)
{
HW_CHAR pcCmdContent;
pcCmdContent = HW_MsgGetStr(pstMsg, EN_IOTA_DEVUPDATE_IE_RESULT);
pcCmdContent = HW_MsgGetStr(pstMsg, EN_IOTA_DEVUPDATE_IE_DEVICEID);

return 0;
}


//绑定广播接收处理函数
HW_BroadCastReg(“IOTA_TOPIC_DEVUPDATE_RSP”, Device_StatusUpdateHandler);
```

6.2.3.3 设备删除

接口功能

当有新设备需要从网关移除时，通过调用设备删除接口将非直连设备从IoT联接管理平台删除。

接口描述

```
HW_INT IOTA_HubDeviceRemove(HW_UINT uiCookie, HW_CHAR *pcDeviceId);
```

参数说明

字段	必选/可选	类型	描述
uiCookie	可选	HW_UINT	Cookie有效值 1-65535
pcDeviceId	必选	String	设备Id

接口返回值

参见函数标准返回值

返回结果

广播名称	广播参数	成员	描述
IOTA_TOPIC_HUB_RMVDEV_RSP	HW_MSG对象	EN_IOTA_HUB_IE_TYPE	删除结果

示例

```
// 开发者调用该接口进行设备删除
HW_CHAR *pcDeviceId = stDevice.pcDeviceId;
```

```
IOTA_HubDeviceRemove(HW_NULL, pcDeviceId);
结果处理：
HW_INT Device_RemoveResultHandler(HW_UINT uiCookie, HW_MSG pstMsg)
{
uiResult = HW_MsgGetUint (pstMsg, EN_IOTA_HUB_IE_RESULT);
if (EN_IOTA_HUB_RESULT_SUCCESS != uiResult)
{
// retry with uiCookie
return 0;
}

return 0;
}

HW_BroadCastReg(“IOTA_TOPIC_HUB_RMVDEV_RSP”, Device_RemovResultHandler);
```

6.2.4 设备服务数据上报

接口功能

当非直连设备上报数据到网关时，网关需要调用设备服务数据上报接口将数据上报到IoT联接管理平台。

接口描述

```
HW_INT IOTA_ServiceDataReport(HW_UINT uiCookie, HW_CHAR *pcRequstId,
HW_CHAR *pcDeviceId, HW_CHAR *pcServiceId, HW_CHAR *pcServiceProperties);
```

参数说明

字段	必选/可选	类型	描述
uiCookie	可选	unsign int	Cookie有效值 1-65535
pcRequstId	条件必选	String	请求ID，用来匹配之前平台下发的服务命令。当该次数据上报为此前某一次命令请求的响应时，需要填写此次命令请求的请求ID。
pcDeviceId	必选	String	设备ID
pcServiceId	必选	String	服务ID
pcServiceProperties	必选	String	服务属性

接口返回值

参见函数标准返回值

返回结果

广播名称	广播参数	成员	描述
IOTA_TOPIC_DATATRANS_REPORT_RSP/{deviceId}	HW_MSG对象	EN_IOTA_DATAREPORT_IE_TYPE	数据上报结果

示例

用户根据Profile格式使用Json组件拼装服务属性的内容(pcServiceProperties)

```
HW_UINT *uiLen;

IOTA_ServiceDataReport(1211, NULL, "xxxx_xxxx_xxxx_xxxx"
, "DoorWindow", "{\"status\":\"OPEN\"}");
```

数据上报结果接收

```
//开发者注册该函数进行设备服务数据上报后的处理
HW_INT Device_DataReportResultHandler(HW_UINT uiCookie, HW_MSG pstMsg)
{
uiResult = HW_MsgGetUint(pstMsg, EN_IOTA_DATATRANS_IE_RESULT);
if (HW_SUCCESS != uiResult)
{
// retry with uiCookie
return 0;
}

return 0;
}
```

```
//在设备添加成功后立即注册服务数据上报结果接收广播
HW_BroadCastReg("IOTA_TOPIC_SERVICE_REPORT_RET/XXXX_XXXX_XXXX_XXXX", Device_AddResultHandler);
```

6.2.5 设备命令接收

接口功能

注册设备命令接收广播来接收处理平台下发的控制命令。

接口描述

```
IOTA_TOPIC_SERVICE_COMMAND_RECEIVE/{deviceId}
```

参数说明

请参考**EN_IOTA_DATATRANS_IE_TYPE 消息信元枚举说明**

返回结果

无

示例

```
// 开发者注册该函数进行命令接收的处理
HW_INT Switch_CommandRecvtHandler(HW_UINT uiCookie, HW_MSG pstMsg)
```

```
{
HW_CHAR *pcMethod, *pcServiceId, *pcCmdContent, *pcDeviceId;

pcDeviceId = HW_MsgGetStr(pstMsg, EN_IOTA_DATATRANS_IE_DEVICEID);
pcServiceId = HW_MsgGetStr(pstMsg, EN_IOTA_DATATRANS_IE_SERVICEID);
pcMethod = HW_MsgGetStr(pstMsg, EN_IOTA_DATATRANS_IE_METHOD);
pcCmdContent = HW_MsgGetStr(pstMsg, EN_IOTA_DATATRANS_IE_CMDCONTENT);

if (strcmp(pcServiceId, “switch”))
{
//根据Proflie定义的命令参数，使用Json组件解析pcCmdContent
//Send command to Switch
}

return 0;
}
```

```
//在设备添加成功后立即注册设备命令接收广播
HW_BroadCastReg(“IOTA_TOPIC_SERVICE_CMD_RECEIVE/XXXX_XXXX_XXXX_XXXX”,
Device_AddResultHandler);
```

开发者需要在设备添加成功后注册该设备的命令接收广播，广播主题为

“ IOTA_TOPIC_SERVICE_CMD_RECEIVE/设备ID”，Agent Lite收到平台发往给设备的命令后会直接广播给该设备注册的广播处理函数。如果开发者不需要按设备进行分发，直接使用主题名即可即可，即” IOTA_TOPIC_SERVICE_CMD_RECEIVE”。

6.2.6 Json 组件使用说明

该组件为Agent Lite提供给开发者的工具组件，如果开发者无法进行Json格式的编码和解码，则可以使用该组件进行编码和解码。主要用于上报数据组装与下发命令解析。

1. Json 编码

使用Json组件进行编码的流程。

创建Json编码对象。

```
HW_JSONOBJ HW_JsonObjCreate()
```

获取Json对象根节点。

```
HW_JSON HW_JsonGetJson(HW_JSONOBJ hjson)
```

往Json对象中添加键值对：

添加pcVal为字符串的Json键值对。

```
HW_INT HW_JsonAddStr(HW_JSON *pstJson, HW_CHAR *pcKey, HW_CHAR *pcVal)
```

添加uiVal为整数的Json键值对。

```
HW_INT HW_JsonAddUint(HW_JSON *pstJson, HW_CHAR *pcKey, HW_UINT uiVal)
```

添加bVal为bool的Json键值对。

```
HW_INT HW_JsonAddBool(HW_JSON *pstJson, HW_CHAR *pcKey, HW_BOOL bVal)
```

添加值为Json的Json键值对，获取到的为子Json对象。

```
HW_JSON HW_JsonAddJson(HW_JSON *pstJson, HW_CHAR *pcKey)
```

添加值为Json数组Json键值对，获取到的为子Json数组对象

```
HW_JSON_ARRAY HW_JsonAddJsonArray(HW_JSON *pstJson, HW_CHAR *pcKey)
```

往Json数组中添加键值对：

添加pcVal为字符串的Json键值对。

```
HW_INT HW_JsonArrayAddStr(HW_JSON_ARRAY *pstArray, HW_CHAR *pcKey, HW_CHAR *pcVal)
```

添加uiVal为整数的Json键值对。

```
HW_INT HW_JsonArrayAddUint(HW_JSON_ARRAY *pstArray, HW_CHAR *pcKey, HW_UINT uiVal)
```

添加bVal为bool的Json键值对。

```
HW_INT HW_JsonArrayAddBool(HW_JSON_ARRAY *pstArray, HW_CHAR *pcKey, HW_BOOL bVal)
```

添加pucValue为Json的Json值，获取到的为子Json对象。

```
HW_JSON HW_JsonArrayAddJson(HW_JSON_ARRAY *pstArray)
```

添加pucValue为Json数组Json键值对，获取到的为子Json数组对象

```
HW_JSON_ARRAY HW_JsonArrayAddJsonArray(HW_JSON_ARRAY *pstArray, HW_CHAR *pcKey)
```

获取Json字符串

```
HW_CHAR HW_JsonEncodeStr(HW_JSONOBJ hJson);
```

删除Json对象

```
HW_VOID HW_JsonObjDelete(HW_JSONOBJ *phJson);
```

Json编码示例：

待解析Json格式：

```
{
"temperature":22
"otherInfo":
{
“batteryLevel”:”low”
}
}
/*变量定义*/
HW_JSONOBJ jsonObj;
HW_JSON rootjson;
HW_JSON json;
HW_CHAR *pcJsonStr;

/*创建Json编码对象*/
hJsonObj = HW_JsonObjCreate();

/*获取跟节点Json对象*/
rootjson = HW_JsonGetJson(hJsonObj);

/*往根节点中添加键值对*/
HW_JsonAddUint(rootjson, "temperature", 22);

/*从根节点中获取子Json对象*/
json = HW_JsonAddJson(rootjson, "otherInfo");
```

```
/*在子Json中添加键值对*/
HW_JsonAddStr(json, " batteryLevel", "low");


/*获取Json字符串*/
pcJsonStr = HW_JsonEncodeStr(hjsonObj);

/*删除之前创建的Json编码对象，释放资源*/
HW_JsonObjDelete(&hJsonObj);
```

2. Json 解码

使用Json组件进行解码的流程

创建Json解析对象。

```
HW_JSONOBJ HW_JsonDecodeCreate(HW_CHAR *pucStr, HW_BOOL bStrCpy)
```

获取Json解析对象中的Json数据部分。

```
HW_JSON HW_JsonGetJson(HW_JSONOBJ hJson)
```

获取Json数据中与pucKey对应的字符串。

```
HW_CHAR *HW_JsonGetStr(HW_JSON pstJson, HW_CHAR *pucKey)
```

获取Json数据中与pucKey对应的无符号整型。

```
HW_UINT HW_JsonGetUint(HW_JSON pstJson, HW_CHAR *pucKey, HW_UINT uiDft)
```

获取Json数据中与pucKey对应的Boolean值。

```
HW_BOOL HW_JsonGetBool(HW_JSON pstJson, HW_CHAR *pucKey, HW_BOOL bDft)
```

获取Json数据中与pucKey对应的数组。

```
HW_UJSON_ARRAY HW_JsonGetArray(HW_JSON pstJson, HW_CHAR *pucKey)
```

获取Json数组的长度。

```
HW_UINT HW_JsonArrayGetCount(HW_UJSON_ARRAY pstArray)
```

获取Json数组中序号为uiIndex项的Json数据。

```
HW_JSON HW_JsonArrayGetJson(HW_UJSON_ARRAY pstArray, HW_UINT uiIndex)
```

获取Json数组中序号为uiIndex项的无符号整形。

```
HW_UINT HW_JsonArrayGetUint(HW_UJSON_ARRAY pstArray, HW_UINT uiIndex, HW_UINT uiDft)
```

获取Json数组中序号为uiIndex项的Boolean值。

```
HW_UINT HW_JsonArrayGetBool(HW_UJSON_ARRAY pstArray, HW_UINT uiIndex, HW_BOOL bDft)
```

获取Json数组中序号为uiIndex项的字符串。

```
HW_CHAR *HW_JsonArrayGetStr(HW_UJSON_ARRAY pstArray, HW_UINT uiIndex)
```

获取Json数组中序号为uiIndex项的子数组。

```
HW_UJSON_ARRAY HW_JsonArrayGetArray(HW_UJSON_ARRAY pstArray, HW_UINT uiIndex)
```

删除之前创建的Json解析对象。

```
HW_VOID HW_JsonObjDelete(HW_JSONOBJ *phJson)
```

Json解析示例：

待解析Json格式：

```
{
"action": "notify",
"type": "userstate",
"userstateinfo":
[
{ "num": "11111 ", "state": "idle"},
{ "num": "11111", "state": "ringing"}
]
}
/*变量定义*/
HW_JSONOBJ jsonObj;
HW_JSON json;
HW_UJSON_ARRAY jsonArray;
HW_CHAR *action;
HW_CHAR *type;
HW_UINT count;
HW_UINT index;

/*创建Json解析对象*/
jsonObj = HW_JsonDecodeCreate(jsonStr, HW_TRUE);

/*获取Json解析对象中的Json数据部分*/
json = HW_JsonGetJson(jsonObj);

/*获取Json数据中与"action"对应的字符串*/
action = HW_JsonGetStr(json, "action");

/*获取Json数据中与"type"对应的字符串*/
type = HW_JsonGetStr(json, "type");

/*获取Json数据中与"userstateinfo"对应的Json数组*/
jsonArray = HW_JsonGetArray(json, "userstateinfo");

/*获取数组jsonArray的长度*/
count = HW_JsonArrayGetCount(jsonArray);
for (index = 0; index < count; index++)
{
/*获取数组jsonArray中序号为index项的Json数据*/
HW_JSON jsonItem = HW_JsonArrayGetJson(jsonArray, index);

/*获取jsonItem中与" num "对应的字符串*/
HW_CHAR *num = HW_JsonGetStr(jsonItem, "num");

/*获取jsonItem中与" state "对应的字符串*/
HW_CHAR *state = HW_JsonGetStr(jsonItem, "state");
......
}

/*删除之前创建的Json解析对象，释放资源*/
HW_JsonObjDelete(jsonObj);
```

6.3 常用数据结构定义

1. ST_IOTA_DEVICE_INFO 结构体说明

字段	必选/可选	类型	描述
pcNodeId	必选	String	关键参数，对接平台的网关下设备唯一标识，设备填写，平台用于判重
pcName	可选	String	设备名称
pcDescription	可选	String	设备描述
pcManufacturerId	必选	String	厂商ID
pcManufacturerName	可选	String	厂商名
pcMac	可选	String	设备MAC地址
pcLocation	可选	String	设备的位置
pcDeviceType	必选	String	设备类型
pcModel	必选	String	型号 z-wave: ProductType + ProductId 16 进制： XXXX-XXXX 补0对齐 如：001A-0A12 其他协议再定
pcSwVersion	可选	String	软件版本， Z-Wave 主次版本号 主版本号.次版本号 如：1.1
pcFwVersion	可选	String	固件版本
pcHwVersion	可选	String	硬件版本
pcProtocolType	必选	String	协议类型 Z-Wave
pcBridgeId	可选	String	表示设备通过哪个Bridge接入平台

字段	必选/可选	类型	描述
pcStatus	可选	String	表示设备是否在线， ONLINE 在线 OFFLINE 离线
statusDetail	可选	String	状态详情，如果pcStatus不为空，则该参数必选。 参数值： 无： NONE 配置等待： CONFIGURATION_PENDING 通信错误： COMMUNICATION_ERROR 配置错误： CONFIGURATION_ERROR 桥接器离线 BRIDGE_OFFLINE 固件升级 FIRMWARE_UPDATING 循环任务 DUTY_CYCLE 未激活 NOT_ACTIVE
pcMute	可选	String	表示设备是否被屏蔽： TRUE FALSE

2. EN_IOTA_BIND_IE_TYPE 消息信元枚举说明

枚举项	枚举值	类型	描述
EN_IOTA_BIND_IE_RESULT	0	EN_IOTA_BIND_RESULT_TYPE	绑定结果

枚举项	枚举值	类型	描述
EN_IOTA_BIND_IE_DEVICEID	1	String	平台分配的逻辑设备ID
EN_IOTA_BIND_IE_DEVICESECRET	2	String	设备接入的鉴权秘钥
EN_IOTA_BIND_IE_APPID	3	String	开发者应用ID
EN_IOTA_BIND_IE_IOCM_ADDR	4	String	IoCM服务器地址
EN_IOTA_BIND_IE_IOCM_PORT	5	unsigned int	IoCM服务器端口
EN_IOTA_BIND_IE_MQTT_ADDR	6	String	MQTT服务器地址
EN_IOTA_BIND_IE_MQTT_PORT	7	unsigned int	MQTT服务器端口
EN_IOTA_BIND_IE_IODM_ADDR	8	String	IoDM服务器地址
EN_IOTA_BIND_IE_IODM_PORT	9	unsigned int	IoDM服务器端口

3. EN_IOTA_BIND_RESULT_TYPE 参数枚举说明

枚举项	枚举值	类型	描述
EN_IOTA_BIND_RESULT_SUCCESS	0	NA	绑定成功
EN_IOTA_BIND_RESULT_DEV_NOT_BIND	1	NA	未扫码
EN_IOTA_BIND_RESULT_VERIFYCODE_EXPIRED	2	NA	验证码过期
EN_IOTA_BIND_RESULT_FAILED	255	NA	其余失败

4. EN_IOTA_LGN_IE_TYPE 消息信元枚举说明

枚举项	枚举值	类型	描述
EN_IOTA_LGN_IE_REASON	0	EN_IOTA_LGN_REASON_TYPE	登录失败原因

5. EN_IOTA_LGN_REASON_TYPE 参数枚举说明

枚举项	枚举值	类型	描述
EN_IOTA_LGN_REASON_NULL	0	NA	无原因
EN_IOTA_LGN_REASON_CONNCET_ERR	1	NA	连接失败
EN_IOTA_LGN_REASON_SERVER_BUSY	2	NA	服务器忙
EN_IOTA_LGN_REASON_AUTH_FAILED	3	NA	鉴权失败、开发者需要停止重新尝试登陆。
EN_IOTA_LGN_REASON_NET_UNAVAILABLE	5	NA	网络不可用
EN_IOTA_LGN_REASON_DEVICE_NOEXIST	12	NA	设备不存在、开发者需要停止重新尝试登陆。
EN_IOTA_LGN_REASON_DEVICE_RMVED	13	NA	设备已删除、开发者需要停止重新尝试登陆。
EN_IOTA_LGN_REASON_UNKNOWN	255	NA	未知原因

6. EN_IOTA_CFG_TYPE 参数枚举说明

枚举项	枚举值	类型	描述
EN_IOTA_CFG_DEVICEID	0	String	平台分配的逻辑设备ID
EN_IOTA_CFG_DEVICESECRET	1	String	设备接入的鉴权秘钥

枚举项	枚举值	类型	描述
EN_IOTA_CFG_APPID	2	String	开发者应用ID
EN_IOTA_CFG_IOCM_ADDR	3	String	IoCM服务器地址
EN_IOTA_CFG_IOCM_PORT	4	unsigned int	IoCM服务器端口
EN_IOTA_CFG_MQTT_ADDR	5	String	MQTT服务器地址
EN_IOTA_CFG_MQTT_PORT	**6**	unsigned int	MQTT服务器端口
EN_IOTA_CFG_IODM_ADDR	**7**	String	IoDM服务器地址
EN_IOTA_CFG_IODM_PORT	**8**	unsigned int	IoDM服务器端口

7. EN_IOTA_HUB_IE_TYPE 消息信元枚举说明

枚举项	枚举值	类型	描述
EN_IOTA_HUB_IE_RESULT	0	EN_IOTA_HUB_RESULT_TYPE	添加/删除执行结果
EN_IOTA_HUB_IE_DEVICEID	1	String	添加成功后分配的设备ID

8. EN_IOTA_HUB_RESULT_TYPE 参数枚举说明

枚举项	枚举值	类型	描述
EN_IOTA_HUB_RESULT_SUCCESS	0	NA	添加/删除执行成功
EN_IOTA_HUB_RESULT_DEVICE_EXIST	1	NA	设备已存在
EN_IOTA_HUB_RESULT_DEVICE_NOTEXIST	2	NA	设备不存在
EN_IOTA_HUB_RESULT_DEVICE_FAILED	255	NA	执行失败

9. EN_IOTA_DATATRANS_IE_TYPE 消息信元枚举说明

枚举项	枚举值	类型	描述	
EN_IOTA_DATATRANS_IE_RESULT	0	unsigned int	命令执行返回结果	成功：0
				失败：1
EN_IOTA_DATATRANS_IE_DEVICEID	1	String	设备ID	
EN_IOTA_DATATRANS_IE_REQUESTID	2	String	请求ID	
EN_IOTA_DATATRANS_IE_SERVICEID	3	String	服务ID	
EN_IOTA_DATATRANS_IE_METHOD	4	String	服务方法	
EN_IOTA_DATATRANS_IE_CMDCONTENT	5	String	命令内容，以Json格式进行拼装的服务命令参数，开发者根据对应服务命令的定义进行Json解析，从而获取命令参数值。	

10. EN_IOTA_DEVUPDATE_IE_TYPE 消息信元枚举说明

枚举项	枚举值	类型	描述	
EN_IOTA_DEVUPDATE_IE_RESULT	0	unsigned int	命令执行返回结果	成功：0
				失败：1
EN_IOTA_DEVUPDATE_IE_DEVICEID	**1**	**String**	设备ID	

6.4 数据类型定义

1. 常用数据类型

类型名称	类型原型
HW_INT	int
HW_UINT	unsigned int

类型名称	类型原型
HW_CHAR	char
HW_UCHAR	unsigned char
HW_BOOL	int
HW_ULONG	unsigned long
HW_USHORT	unsigned short
HW_MSG	void*
HW_VOID	void
HW_NULL	0

2. 函数标准返回值

返回值名称	值	类型原型
HW_OK	0	执行成功
HW_ERR	1	执行错误
HW_ERR_PTR	2	错误的指针
HW_ERR_ID	3	错误的ID
HW_ERR_PARA	4	错误的参数
HW_ERR_KEY	5	错误的KEY
HW_ERR_NOMEM	6	内存不足
HW_ERR_MAGIC	7	保留
HW_ERR_OVERFLOW	8	存在溢出
HW_ERR_GVAR	9	保留
HW_ERR_POOL	10	保留
HW_ERR_NO_MUTEX	11	未加锁
HW_ERR_PID	12	保留
HW_ERR_FILEOPEN	13	文件打开失败
HW_ERR_FD	14	错误的文件描述符
HW_ERR_SOCKET	15	SOCKET异常
HW_ERR_NOTSUPPORT	16	不支持
HW_ERR_NOTLOAD	17	未加载

返回值名称	值	类型原型
HW_ERR_ENCODE	18	编码错误
HW_ERR_DECODE	19	解码错误
HW_ERR_CALLBACK	22	错误的回调函数
HW_ERR_STATE	23	错误的状态
HW_ERR_OVERTIMES	24	重试超过次数
HW_ERR_ENDOVER	25	保留
HW_ERR_ENDLINE	26	保留
HW_ERR_NUMBER	27	错误的数字
HW_ERR_NOMATCH	28	不匹配
HW_ERR_NOSTART	29	未开始
HW_ERR_NOEND	30	未结束
HW_ERR_OVERLAP	31	保留
HW_ERR_DROP	32	丢弃
HW_ERR_NODATA	33	无数据
HW_ERR_CRC_CHK	34	CRC校验失败
HW_ERR_AUTH	35	鉴权失败
HW_ERR_LENGTH	36	长度错误
HW_ERR_NOTALLOW	37	不被允许的操作
HW_ERR_TOKEN	38	凭据错误
HW_ERR_NOTIPV4	39	不支持IPV4
HW_ERR_NOTIPV6	40	不支持IPV6
HW_ERR_IELOST	41	保留
HW_ERR_IELOST1	42	保留
HW_ERR_IELOST2	43	保留
HW_ERR_AUDIO	44	保留
HW_ERR_VIDEO	45	保留
HW_ERR_MD5	46	保留
HW_ERR_MD5_HA1	47	保留
HW_ERR_MD5_RES	48	保留
HW_ERR_DIALOG	49	错误的对话

返回值名称	值	类型原型
HW_ERR_OBJ	50	错误的对象

7 APP SDK API 参考(Android)

7.1 开发者必读

SDK 开放组件介绍

SDK提供如下组件：

说明

所有接口参数的有效性检查由开发者自己保证。

API组件	说明
SysApi	系统初始化，提供整个系统初始化、设置和获取配置信息，相关接口
LogApi	日志打印，提供日志打印和日志上传相关接口
UpgradeApi	升级组件，提供App新版本检测与升级安装相关接口
LoginApi	登录注销，提供登录、注销、激活账号等操作接口
UserApi	用户管理，提供自助开户、忘记密码、修改密码操作相关接口
DeviceApi	设备管理，提供设备相关操作接口，如设备接入、数据上报、控制命令下发和视频相关接口

API组件	说明
EventApi	事件处理，提供新事件上报、事件查询、删除等相关操作接口
PushApi	离线推送，提供离线推送相关接口（仅支持GCM）
RoomApi	房间组件，提供房间创建、更新、删除、添加设备到房间相关操作接口
RuleApi	规则组件，提供规则创建、修改、删除、激活\去激活等相关操作接口
SettingsApi	设置组件，提供事件通知设置查询与修改，云存储自动清理开关配置查询与修改等相关操作接口

修改记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

文档版本	发布日期	修改说明	修改影响
1	2016/06/01	第一次发布	

7.2 SDK 功能介绍

该章节提供的SDK功能包括，系统的初始化配置，日志打印收集功能，登陆及注销功能，用户及用户信息管理，设备相关的操作及信息获取，摄像头的操作，实时视频及历史视频的播放，设备及操作产生的事件呈现，离线消息推送，规则增删改查。该章节旨在详细介绍SDK提供的所有功能和接口，开发者可按照该章节进行相关接口和功能的开发调试。

7.2.1 系统初始化

系统初始化（SysApi）提供整个系统初始化、设置和获取配置信息相关接口。开发者需要先初始化整个系统，然后才能调用其它API。

7.2.1.1 初始化

接口功能：

初始化整个系统。

接口原型：

```
void init(Context context)。
```

参数说明：

参数	必选/可选	类型	描述
context	必选	Context	Android系统上下文

返回结果：

类型	描述
Void	无返回值

注意事项：

在调用其它API前，需要确保该接口已经被调用。建议在程序启动的时候调用该接口。

调用示例：

```
SysApi.init(getApplicationContext());
```

7.2.1.2 设置配置信息

接口功能：

设置配置信息，如服务器IP、端口等。

接口原型：

```
void setConfig(String key, String value)
```

参数说明：

参数	必选/可选	类型	描述
key	必选	String	配置项。 当前支持如下配置项。开发也可以自己定义配置项，设置到SDK存储。 SysApi.SYS_CFG_NA_SERVER_IP配置服务器IP SysApi.SYS_CFG_NA_SERVER_PORT配置服务器端口 SysApi.SYS_CFG_MQTT_SERVER_IP配置推送服务器IP SysApi.SYS_CFG_MQTT_SERVER_PORT配置推送服务器端口 SysApi.SYS_CFG_OPERATION_TIMEOUT_TIME 配置操作超时时间, 单位秒(s)
value	必选	String	配置项的值。

返回结果

类型	描述
Void	无返回值

注意事项

在调用其它业务API前，需要调用该接口设置服务器的地址。

调用示例

```
SysApi.setConfig(SysApi.SYS_CFG_NA_SERVER_IP, TEST_SERVICE_IP);
```

7.2.1.3 获取配置信息

接口功能：

获取配置项的值。

接口原型：

```
String getConfig(String key)。
```

参数说明：

参数	必选/可选	类型	描述
key	必选	String	配置项。 当前支持如下配置项。开发也可以自己定义设置到SDK存储的配置项。 SysApi.SYS_CFG_NA_SERVER_IP配置服务器IP SysApi.SYS_CFG_NA_SERVER_PORT配置服务器端口 SysApi.SYS_CFG_MQTT_SERVER_IP配置推送服务器IP SysApi.SYS_CFG_MQTT_SERVER_PORT配置推送服务器端口 SysApi.SYS_CFG_OPERATION_TIMEOUT_TIME 配置操作超时时间，，单位秒(s)。

返回结果

类型	描述
String	相关配置项的值

注意事项

若该配置项没有，会返回空字符串”“或null

调用示例

```
SysApi.getConfig(SysApi.SYS_CFG_NA_SERVER_IP);
```

7.2.1.4 销毁

接口功能

释放SysApi的资源

接口原型

```
void destroy()
```

参数说明

参数	必选/可选	类型	描述
无参数			

返回结果

类型	描述
Void	无返回值

注意事项

退出APP前需要调用该接口回收资源

调用示例

```
SysApi.destroy();
```

7.2.2 日志打印

日志打印（LogApi）提供日志打印和日志上传相关接口。在调用LogApi提供的相关接口前，必须确保已经调用了SysApi的init接口初始化了整个系统。

7.2.2.1 打印信息级别日志

接口功能

打印Info级别日志

接口原型

```
void i(String tag, String msg)
```

参数说明

参数	必选/可选	类型	描述
tag	必选	String	日志标识，通常是调用者的类名
msg	必选	String	日志内容

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
LogApi.i("Tag", "message");
```

7.2.2.2 打印告警级别日志

接口功能

打印告警级别日志

接口原型

```
void w(String tag, String msg)
```

参数说明

参数	必选/可选	类型	描述
tag	必选	String	日志标识，通常是调用者的类名
msg	必选	String	日志内容

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
LogApi.w("Tag", "message");
```

7.2.2.3 打印调试级别日志

接口功能

打印调试级别日志

接口原型

```
void d(String tag, String msg)
```

参数说明

参数	必选/可选	类型	描述
tag	必选	String	日志标识，通常是调用者的类名
msg	必选	String	日志内容

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
LogApi.d("Tag", "message");
```

7.2.2.4 打印错误级别日志

接口功能

打印错误级别日志

接口原型

```
void e(String tag, String msg)
```

参数说明

参数	必选/可选	类型	描述
tag	必选	String	日志标识，通常是调用者的类名
msg	必选	String	日志内容

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
LogApi.e("Tag", "message");
```

7.2.2.5 日志上传

接口功能

上传日志到云端，用于问题分析。

接口原型

```
void uploadLog(UploadLogListener listener)
```

参数说明

参数	必选/可选	类型	描述
listener	必选	UploadLogListener	日志上传结果回调

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口的前提条件是，已经登录成功。

调用示例

```
LogApi.uploadLog(new LogApi.UploadLogListener() {
    @Override
    public void onProgress(int currentProgress) {
        //show the progress
    }

    @Override
    public void onSuccess() {
        //do something
    }

    @Override
    public void onError(int errorCode) {
        //do something
    }
});
```

7.2.2.6 收到自动收集日志消息广播

接口功能

收到自动收集日志消息广播通知。

接口原型

```
EVENT_RECEIVE_AUTO_COLLECTION_LOG_MESSAGE
```

参数说明

参数	必选/可选	类型	描述
void			

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mAutoCollectionLogReceiver,
new IntentFilter(LogApi.EVENT_RECEIVE_AUTO_COLLECTION_LOG_MESSAGE));
……
private static BroadcastReceiver mAutoCollectionLogReceiver = new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {
  //do something
  }
};
……
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mAutoCollectionLogReceiver);
```

7.2.3 版本升级

App新版本的检测与升级依赖UpgradeApi组件，提供App新版本检测、新版本安装功能

7.2.3.1 检测新版本

接口功能

检测当前是否有新版本

接口原型

```
void checkNewVersion(ResultsListener<UpgradeInfoEntity> listener)
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener<UpgradeInfoEntity>	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
UpgradeApi.checkNewVersion(new ResultsListener<UpgradeInfoEntity>() {
    @Override
    public void onProgress(int currentProgress) {
        //show the progress
    }

    @Override
    public void onSuccess(UpgradeInfoEntity info) {
        //do something
    }

    @Override
    public void onError(int errorCode) {
        //do something
    }
});
```

7.2.3.2 下载新版本

接口功能

从指定URL下载版本到指定路径

接口原型

```
void downloadNewVersion(String srcUrl, String destPath, FileDownloadListener listener)
```

参数说明

参数	必选/可选	类型	描述
srcUrl	必选	String	新版本地址URL
destPath	必选	String	下载保存的目的地址
listener	必选	FileDownloadListener	下载结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

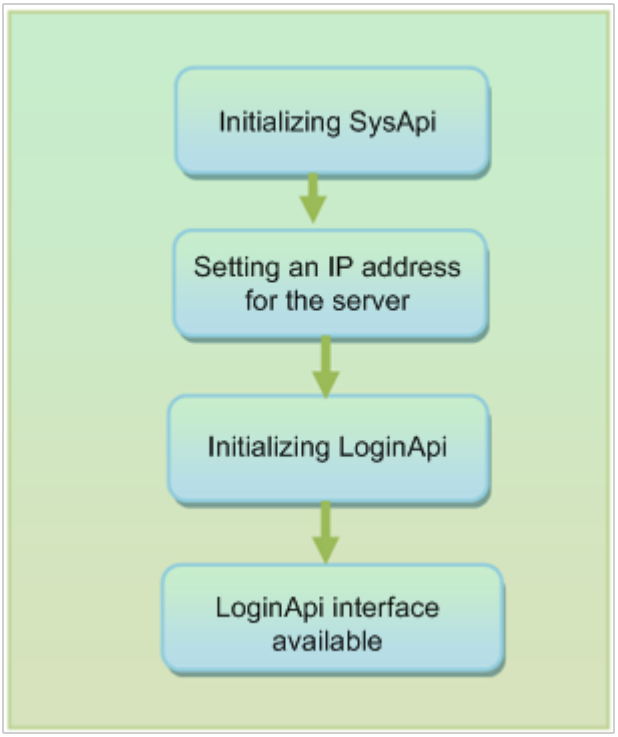
```
private void downloadNewVersionClick(){
    String srcUrl = "https://171.0.0.0:8080/test";
    String destPath = "/sdcard/test";
    UpgradeApi.downloadNewVersion(srcUrl ,destPath ,new ResultsListener() {
        @Override
        public void onProgress(int currentProgress) {
            //show the progress
        }

        @Override
        public void onSuccess() {
            //do something
        }

        @Override
        public void onError(int errorCode) {
            //do something
        }
    });
}
```

7.2.4 登录注销

提供登录、注销、等操作接口。要实现登录相关业务，需要确保调用SysAp.init(Context context)接口初始化了系统，调用SysApi.setConfig(String key, String value)设置了服务器的地址。



7.2.4.1 初始化

接口功能

初始化LoginApi

接口原型

```
void init(Context context)
```

参数说明

参数	必选/可选	类型	描述
context	必选	Context	Android 系统上下文

返回结果

类型	描述
Void	无返回值

注意事项

调用前需要确保已经调用SysApi.init(Context context)初始化了SysApi。调用登录组件的初始化接口后，登录组件的其它接口才可用。

调用示例

```
LoginApi.init(getApplicationContext());
```

7.2.4.2 登录

接口功能

用户登录

接口原型

```
void login(String userId, String password, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户ID，如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
password	必选	String	密码
listener	必选	ResultsListener<Class<Void>>	操作结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口能够登录成功的前提条件是，SysApi已经初始化、LoginApi已经初始化、服务器的地址已经设置。

调用示例

```
LoginApi.login("a user ID", "password", new ResultsListener<Class<Void>>() {
    @Override
    public void onSuccess(Class<Void> results) {
        //do something
    }

    @Override
    public void onError(int errorCode) {
        //do something
    }

    @Override
    public void onTimeout() {
        //do something
    }
});
```

7.2.4.3 带扩展参数登录

接口功能

用户登录，可以携带除用户ID和密码外的扩展参数，如验证码

接口原型

```
void login(String userId, String password, Intent intent, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户ID，如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
password	必选	String	密码
intent	必选	Intent	Android系统Intent，用来携带扩展参数，当前支持： LOGIN_EXTENDING_PARAM_VERIFY_CODE 携带登录验证码（长度是4位）：多次鉴权失败，再次登录需要携带验证码。如何获取验证码，请参考获取登录验证码章节。 LOGIN_EXTENDING_PARAM_TRUST_DEVICE_VERIFY_CODE携带新设备授权验证码：新设备首次登录，需要携带授权码，平台会将授权码以邮件的方式放到该账户注册的邮箱。
listener	必选	ResultsListener<Class<Void>>	操作结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口能够登录成功的前提条件是，SysApi已经初始化、LoginApi已经初始化、服务器的地址已经设置。

调用示例

```
Intent intent = new Intent();
intent.putExtra(LoginApi.LOGIN_EXTENDING_PARAM_VERIFY_CODE, "a code");
intent.putExtra(LoginApi. LOGIN_EXTENDING_PARAM_TRUST_DEVICE_VERIFY_CODE, "a code");
```

```
LoginApi.login("a user ID", "password", intent, new ResultsListener<Class<Void>>() {
    @Override
    public void onSuccess(Class<Void> results) {
        //do something
    }

    @Override
    public void onError(int errorCode) {
        //do something
    }

    @Override
    public void onTimeout() {
        //do something
    }
});
@Override
public void onTimeout() {
//do something
}
});
```

7.2.4.4 获取登录验证码

接口功能

多次登录鉴权失败，用户需要输入验证码登录。该接口用于从云端获取验证码。

接口原型

```
void getVerifyCodeForLogin(ResultsListener<Bitmap> listener)
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener<Bitmap>	操作结果监听器，携带验证码图片

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口的前提条件是，SysApi已经初始化、LoginApi已经初始化、服务器的地址已经设置。

调用示例

```
LoginApi.getVerifyCodeForLogin(new ResultsListener<Bitmap>() {
@Override
public void onSuccess(Bitmap results) {
//do something
}

@Override
```

```
public void onError(int errorCode) {
//do something
}
```

7.2.4.5 登出

接口功能

登出，logout

接口原型

```
void logout(ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener	操作结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口的前提条件是，SysApi已经初始化、LoginApi已经初始化、服务器的地址已经设置，并且已经成功登录过。

调用示例

```
LoginApi.logout(new ResultsListener<Class<Void>>() {
@Override
public void onSuccess(Class<Void> results) {
//do something
}

@Override
public void onError(int errorCode) {
//do something
}

@Override
public void onTimeout() {
//do something
}
});
```

7.2.4.6 激活账号

接口功能

用户开户成功后，激活开户时填写的账号。

接口原型

```
void activateAccount(String userId, String activationCode,  ResultsListener<String> listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户Id（手机或邮箱），当前为用户手机
activationCode	激活码	String	广播参数，若当前的登录状态为LOGIN_STATUS_DISCONNECTED 断开连接，会携带断开连接的原因值： ERROR_CODE_LOGIN_AUTH_FAILED 鉴权失败
listener	可选	ResultsListener<String>	操作结果监听器，携带token

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void activateAccountButtonClick(){
String userId = “0086123456789”;
String activationCode = “36954”;
LoginApi.activateAccount(userId, activationCode, new ResultsListener<String>() {
@Override
public void onSuccess(String results) {
//do something
}
@Override
public void onError(int errorCode) {
//do something
switch (errorCode) {
case LoginApi. ERROR_CODE_ACTIVATE_ACCOUNT_ACTIVATION_CODE_EXPIRED:
//do something
break;
caseLoginApi. ERROR_CODE_ACTIVATE_ACCOUNT_ATTEMPTS_TIMES_REACH_MAX:
//do something
break;
case LoginApi. ERROR_CODE_ACTIVATE_ACCOUNT_ACTIVATION_CODE_EXISTED:
//do something
Break;
default:
break;
```

```
}
}
@Override
public void onTimeout() {
 //do something
}
});
}
```

7.2.4.7 获取激活码

接口功能

获取激活码，服务器发送邮件到用户开户邮箱中

接口原型

```
void requestActivationCode(String userId, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户Id（手机或邮箱），当前为用户手机
listener	可选	ResultsListener<String>	操作结果监听器，携带token

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void requestActivationCodeButtonClick(){
    String userId = “0086123456789”;
    LoginApi. requestActivationCode(userId, new ResultsListener () {
    @Override
    public void onSuccess() {
     //do something
    }
    @Override
    public void onError(int errorCode) {
        switch (errorCode) {
        case LoginApi.ERROR_CODE_REQUEST_ACTIVATION_CODE_FAILED:
        //do something
        break;
        case LoginApi.ERROR_CODE_REQUEST_ACTIVATION_CODE_ATTEMPTS_TIMES_REACH_MAX:
        //do something
        break;
```

```
        default:
        break;
        }
    }
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.4.8 登录状态改变广播

接口功能

当登录状态改变时，上报最新的登录状态。

接口原型

```
String EVENT_LOGIN_STATUS_CHANGED
```

参数说明

参数	必选/可选	类型	描述
PARAM_LOGIN_STATUS	必选	String	广播参数，携带当前登录状态： LOGIN_STATUS_CONNECTING连接中 LOGIN_STATUS_CONNECTED 已经连接 LOGIN_STATUS_DISCONNECTED 断开连接
PARAM_LOGIN_REASON	可选	String	广播参数，若当前的登录状态为LOGIN_STATUS_DISCONNECTED 断开连接，会携带断开连接的原因值： ERROR_CODE_LOGIN_AUTH_FAILED 鉴权失败 ERROR_CODE_LOGIN_ACCOUNT_INACTIVE 账号未激活 ERROR_CODE_LOGIN_ACCOUNT_LOCKED 账号因多次鉴权失败被锁定 ERROR_CODE_LOGIN_INVALID_VERIFY_CODE 验证码错误 ERROR_CODE_LOGIN_FORCE_LOGOUT 账号被踢 ERROR_CODE_SYS_NETWORK_UNAVAILABLE 网络不可用

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mLoginStatusChangedReceiver,
        new IntentFilter(LoginApi.EVENT_LOGIN_STATUS_CHANGED));

.........

private static BroadcastReceiver mLoginStatusChangedReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        int status = intent.getIntExtra(LoginApi.PARAM_LOGIN_STATUS, LOGIN_STATUS_DISCONNECTED);
        int errorCode = intent.getIntExtra(LoginApi.PARAM_LOGIN_REASON,
LoginApi.ERROR_CODE_SYS_NULL);

        switch (status) {
            case LoginApi.LOGIN_STATUS_CONNECTING:
                //do something
                break;

            case LoginApi.LOGIN_STATUS_CONNECTED:
                //do something
                break;
            case LoginApi.LOGIN_STATUS_DISCONNECTED:
                //do something
                break;
            default:
                break;
        }
    }
};
```

7.2.4.9 销毁

接口功能

销毁LoginApi

接口原型

```
void destroy()
```

参数说明

参数	必选/可选	类型	描述
无参数			

返回结果

类型	描述
Void	无返回值

注意事项

应用退出前需要调用该接口释放LoginApi的资源；调用该接口后，再调用LoginApi其它接口前需要调用LoginApi.init(Context context)初始化LoginApi。

调用示例

```
LoginApi.destroy();
```

7.2.5 用户管理

用户管理（UserApi），提供自助开户、忘记密码、修改密码相关接口。调用用户组件相关接口，需要确保调用SysAp.init(Context context)接口初始化了系统，调用SysApi.setConfig(String key, String value)设置了服务器的地址。

7.2.5.1 初始化

接口功能

初始化UserApi

接口原型

```
void init(Context context)
```

参数说明

参数	必选/可选	类型	描述
context	必选	Context	Android 系统上下文

返回结果

类型	描述
Void	无返回值

注意事项

调用前需要确保已经调用SysApi.init(Context context)初始化了SysApi。调用用户组件初始化接口后，用户组件的其它接口才可用。

调用示例

```
UserApi.init(getApplicationContext());
```

7.2.5.2 用户自助开户

接口功能

自助开户，注册账号

接口原型

```
void registerUser(String userId, String userName, String password, Intent intent, ResultsListener
listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户ID，如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
userName	必选	String	用户名
intent	可选	Intent	Android系统Intent，用来携带扩展参数，当前支持： USER_EXTENDING_PARAM_EMAIL携带邮箱信息
listener	必选	ResultsListener	操作结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口的前提条件是，SysApi已经初始化、UserAPi已经初始化、服务器的地址已经设置。

调用示例

```
private void registerUserClick(){
    String userId = "+8612345678910";
    String username = "user name";
    String password = “adc123”;
    Intent intent = new Intent();
    intent.putExtra(UserApi.USER_EXTENDING_PARAM_EMAIL, "emai@example.com");
    UserApi.registerUser(userId , username , intent, password , new ResultsListener<Class<Void>>()
{
    @Override
    public void onSuccess(Class<Void> results) {
    //do something
    }

    @Override
    public void onError(int errorCode) {
    //do something
    }

    @Override
    public void onTimeout() {
    //do something
   }
```

```
    });
}
```

7.2.5.3 忘记密码获取验证码

接口功能

忘记密码，找回密码，需要先从平台获取验证码，该验证码会以短信方式发到用户注册的手机号码上。该接口用于获取该验证码。

接口原型

```
void getVerifyCodeForRecoveryPassword(String userId, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户ID，如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
listener	必选	ResultsListener	操作结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口的前提条件是，SysApi已经初始化、UserAPi已经初始化、服务器的地址已经设置。

调用示例

```
private void getVerifyCodeForRecoveryPasswordClick(){
    String userId = “+8612345678910”;
    UserApi. getVerifyCodeForRecoveryPassword(userId , new ResultsListener<Class<Void>>() {
    @Override
    public void onSuccess(Class<Void> results) {
    //do something
    }

    @Override
    public void onError(int errorCode) {
    //do something
    }

    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.5.4 忘记密码验证验证码

接口功能

忘记密码，找回密码，需要先从平台获取验证码，该验证码会以短信方式发到用户注册的手机号码上，然后用户输入验证码，进行验证。该接口用于验证验证码。

接口原型

```
void validateRecoveryPasswordVerifyCode(String userId, String verifyCode, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户ID，如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
verifyCode	必选	String	验证码
listener	必选	ResultsListener	操作结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口的前提条件是，SysApi已经初始化、UserApi已经初始化、服务器的地址已经设置，已经获取过忘记密码的验证码。

调用示例

```
private void validateRecoveryPasswordVerifyCodeClick(){
    String userId = “+8612345678910”;
    String verifyCode = “12345”
    UserApi.validateRecoveryPasswordVerifyCode(userId , verifyCode , new
ResultsListener<Class<Void>>() {
    @Override
    public void onSuccess(Class<Void> results) {
    //do something
    }

    @Override
    public void onError(int errorCode) {
    //do something
    }

     @Override
     public void onTimeout() {
     //do something
     }
```

```
    });
}
```

7.2.5.5 忘记密码重置密码

接口功能

忘记密码，找回密码，需要先从平台获取验证码，该验证码会以短信方式发到用户注册的手机号码上，然后用户输入验证码，进行验证，验证成功后，用户可以重置密码。该接口用于重设密码。

接口原型

```
void resetPassword(String userId, String newPassword, String verifyCode, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户ID，如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
newPassword	必选	String	新密码
verifyCode	必选	String	验证码
listener	必选	ResultsListener	操作结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口的前提条件是，SysApi已经初始化、UserApi已经初始化、服务器的地址已经设置，已经获取了忘记密码验证码，并且验证该验证码成功。

调用示例

```
private void resetPasswordClick(){
    String userId = “+8612345678910”;
    String password = “abc123”
    String verifyCode = “12345”
    UserApi.resetPassword(userId , password , verifyCode , new ResultsListener<Class<Void>>() {
    @Override
    public void onSuccess(Class<Void> results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
    //do something
    }
```

```
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.5.6 修改密码

接口功能

修改密码

接口原型

```
void modifyPassword(String userId, String oldPassword, String newPassword, ResultsListener
listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户ID，如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
oldPassword	必选	String	旧密码
newPassword	必选	String	新密码
listener	必选	ResultsListener	操作结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口的前提条件是，已经登录成功。

调用示例

```
private void modifyPasswordClick(){
    String userId = “+8612345678910”;
    String oldPassword = “abc123”
    String newPassword = “def456”;
    UserApi.modifyPassword(userId , oldPassword , newPassword , new ResultsListener<Class<Void>>()
{
    @Override
    public void onSuccess(Class<Void> results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
```

```
    //do something
    }
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.5.7 查询个人信息

接口功能

查询个人信息

接口原型

```
void queryMyInfo(final String userId, final ResultsListener<HashMap<String, String>> listener)
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户ID，如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
listener	必选	ResultsListener	操作结果监听器

返回结果

类型	描述
HashMap <String, String>	扩展参数，携带要设置的个人信息。当前支持： USER_EXTENDING_PARAM_EMAIL 设置个人信息扩展参数：邮箱 USER_EXTENDING_PARAM_USER_NAME 设置个信息扩展参数：用户名

注意事项

调用该接口的前提条件是，已经登录成功。

调用示例

```
private void queryMyInfoClick(){
    String userId = “+8612345678910”;
    UserApi.queryMyInfo(userId , new ResultsListener<HashMap<String, String>>() {
    @Override
    public void onSuccess(HashMap<String, String> results) {
        String userName = results.get(UserApi.USER_EXTENDING_PARAM_USER_NAME);
        String email = results.get(UserApi.USER_EXTENDING_PARAM_EMAIL);
        // do something
    }
    @Override
    public void onError(int errorCode) {
        // do something
    }
```

```
    @Override
    public void onTimeout() {
        // do something
    }
    });
}
```

7.2.5.8 设置个人信息

接口功能

设置个人信息，当前支持设置邮箱和用户名信息

接口原型

```
void setMyInfo(HashMap params, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
params	必选	HashMap	扩展参数，携带要设置的个人信息。当前支持： USER_EXTENDING_PARAM_EMAIL设置个人信息扩展参数：邮箱 USER_EXTENDING_PARAM_USER_NAME设置个信息扩展参数：用户名
listener	必选	ResultsListener	操作结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

调用该接口的前提条件是，已经登录成功。

调用示例

```
private void setMyInfoClick(){
    HashMap params = new HashMap();
    UserApi.setMyInfo(params, new ResultsListener<Class<Void>>() {
    @Override
    public void onSuccess(Class<Void> results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
```

```
    //do something
    }
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.5.9 查询所有用户

接口功能

查询当前登录用户家庭中的所有用户。

接口原型

```
void queryAllUsers(ResultsListener<List<User>> listener)
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener<List<User>>	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
UserApi.queryAllUsers(new ResultsListener<List<User>>() {
    @Override
    public void onSuccess(<List<User>> results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
    //do something
    }
    @Override
    public void onTimeout() {
    //do something
    }
});
```

7.2.5.10 邀请子用户

接口功能

邀请一个子用户加入到当前登录用户的家庭中。

接口原型

```
void inviteUser(String userId, String userName, ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	被邀请用户Id（一般为手机号）
userName	必选	String	被邀请用户的名字
listener	必选	ResultsListener<List<User>>	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

调用该接口的前提条件是，已经登录成功。

调用示例

```
private void inviteUserButtonClick(){
     String userId = “0086123456789”;
     String username = ”LiSi”;
     UserApi.inviteUser(userId, userName, new ResultsListener () {
     @Override
     public void onSuccess() {
     //do something   }
     @Override
     public void onError(int errorCode) {
         //do something
         switch (errorCode) {
               case UserApi.ERROR_CODE_INVITE_USER_EXISTED_IN_OTHER_HOME:
                //do something
                 break;
               case UserApi.ERROR_CODE_INVITE_USER_CAN_NOT_BE_INVITED:
                //do something
                  break;
               case UserApi.ERROR_CODE_INVITE_USER_NUMBER_REACH_MAX:
                //do something
                  break;
               case UserApi.ERROR_CODE_INVITE_USER_CAN_NOT_BE_SELF:
               //do something
                  break;
           default:
                break;
          }
     }
    @Override
    public void onTimeout() {
    //do something
     }
```

```
    });
}
```

7.2.5.11 子用户接收邀请

接口功能

用户收到其它用户发来的邀请加入家庭后，接受这个邀请，加入邀请用户的家庭中。

接口原型

```
void acceptInvitation(ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener	操作结果通知

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void acceptInvitationButtonClick(){
      UserApi.acceptInvitation(new ResultsListener () {
      @Override
      public void onSuccess() {
       //do something
       }
      @Override
      public void onError(int errorCode) {
      //do something
      }
      @Override
      public void onTimeout() {
      //do something
      }
      });
}
```

7.2.5.12 子用户拒绝邀请

接口功能

用户接收到其它用户发来的邀请加入家庭后，拒绝这个邀请，不加入邀请用户的家庭中。

接口原型

```
void rejectInvitation(ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener	操作结果通知

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void rejectInvitationButtonClick(){
     UserApi.rejectInvitation(new ResultsListener () {
     @Override
     public void onSuccess() {
     //do something
     }
     @Override
     public void onError(int errorCode) {
     //do something
     }
     @Override
     public void onTimeout() {
     //do something
     }
     });
}
```

7.2.5.13 查询套餐信息

接口功能

查询用户开户时，选择的套餐信息。

接口原型

```
void queryServicePackageInfo(ResultsListener<List<ServicePackageInfo>> listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener<List<ServicePackageInfor>>	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void queryServicePackageInfoButtonClick(){
     UserApi.queryServicePackageInfo(new ResultsListener<List<ServicePackageInfo>>() {
     @Override
     public void onSuccess(List<ServicePackageInfo> list) {
     //do something
     }
     @Override
     public void onError(int errorCode) {
     //do something
     }
     @Override
     public void onTimeout() {
     //do something
     }
     });
}
```

7.2.5.14 套餐信息变更广播通知

接口功能

上报套餐信息变更通知。

接口原型

```
String PARAM_PACKAGE_INFO
```

参数说明

参数	必选/可选	类型	描述
PARAM_PACKAGE_INFO	必选	String	套餐信息

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mPackageChangeReceiver, new
IntentFilter(UserApi.PARAM_PACKAGE_INFO));
.........
private static BroadcastReceiver mPackageChangeReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
//do something
}
};
.........
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mPackageChangeReceiver);
```

7.2.5.15 用户销户广播通知

接口功能

上报用户被销户通知

接口原型

```
String EVENT_USER_DELETED
```

参数说明

参数	必选/可选	类型	描述
无参数			

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mUserDeletedReceiver, new
IntentFilter(UserApi.EVENT_USER_DELETED));
.........
private static BroadcastReceiver mUserDeletedReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
//do something
}
};
.........
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mUserDeletedReceiver);
```

7.2.5.16 用户去激活广播通知

接口功能

上报用户去激活通知

接口原型

```
String EVENT_USER_INACTIVE
```

参数说明

参数	必选/可选	类型	描述
无参数			

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mUserInactiveReceiver, new
IntentFilterUserApi.EVENT_USER_INACTIVE));
.....
private static BroadcastReceiver mUserInactiveReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
//do something
}
};
.....
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mUserDeletedReceiver);
```

7.2.5.17 销毁

接口功能

销毁UserApi

接口原型

```
void destroy()
```

参数说明

参数	必选/可选	类型	描述
无参数			

返回结果

类型	描述
Void	无返回值

注意事项

应用退出前需要调用该接口释放UserApi的资源；调用该接口后，再调用UserApi其它接口前需要调用UserApi.init(Context context)初始化UserApi。

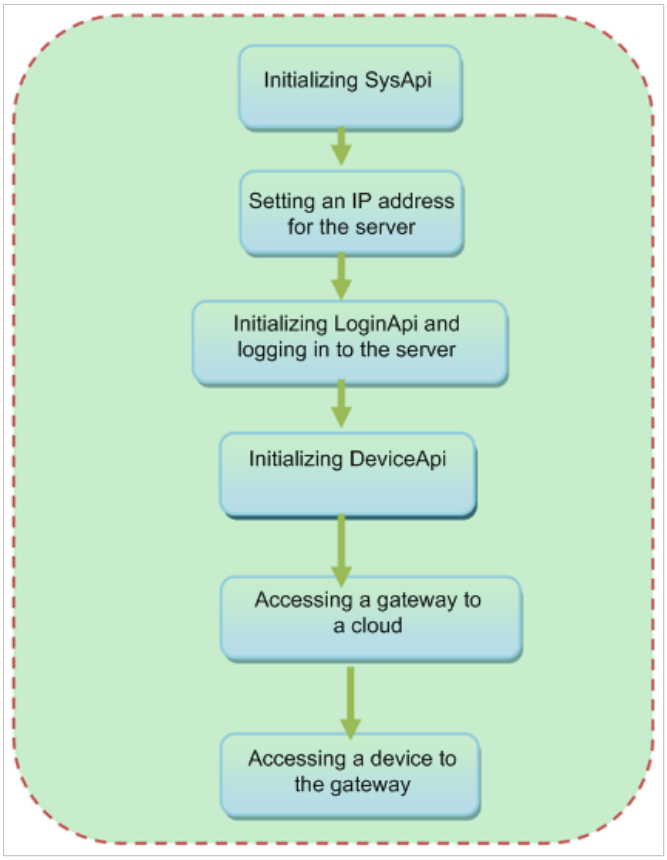
调用示例

```
UserApi.destroy();
```

int ERROR_CODE_USER_RESET_PASSWORD_PASSWORD_MATCHES_OLD	修改密码、重设密码错误码：重设密码新老密码一样

7.2.6 设备管理

设备管理提供了设备接入云、设备接入网关、设备连接状态检查、设备查询、设备信息修改、设备删除、设备控制命令下发、实时视频、视频录像、抓图、视频下载等一系列接口。调用设备组件相关接口前，需要确保调用SysAp.init(Context context)接口初始化了系统，调用SysApi.setConfig(String key, String value)设置了服务器的地址，调用登录接口登录成功。接入设备到网关之前，需要把网关（直连设备）接入到云。



7.2.6.1 初始化

接口功能

初始化DeviceApi

接口原型

```
void init(Context context)
```

参数说明

参数	必选/可选	类型	描述
context	必选	Context	Android 系统上下文

返回结果

类型	描述
Void	无返回值

注意事项

调用前需要确保已经调用DeviceApi.init(Context context)初始化了DeviceApi。调用该接口后，登录组件的其它接口才可用。

调用示例

```
DeviceApi.init(getApplicationContext());
```

7.2.6.2 销毁

接口功能

销毁DeviceApi

接口原型

```
void destroy()
```

参数	必选/可选	类型	描述
无参数			

参数说明

返回结果

类型	描述
Void	无返回值

注意事项

应用退出前需要调用该接口释放DeviceApi的资源；调用该接口后，再调用DeviceApi其它接口前需要调用Device.init(Context context)初始化DeviceApi。

调用示例

```
DeviceApi.destroy();
```

7.2.6.3 注册设备到云

接口功能

应用程序注册直连设备到云上，注册成功后，会返回该设备的信息。

接口原型

```
void registerDeviceToCloud(String pin, ResultsListener<Device> listener)
```

或

```
void registerDeviceToCloud(String pin, int timeoutTime, ResultsListener<Device> listener)
```

参数说明

参数	必选/可选	类型	描述
pin	必选	String	物理设备唯一标识符，视具体厂商而定。例如：设备Mac地址、设备UUID等
timeoutTime	可选	int	超时时长，单位是秒，默认为90s（不能小于30s）
listener	必选	ResultsListener<Device>	接口结果监听器

返回结果

类型	描述
void	无返回值

注意事项

注册到云上的设备必须是直连类设备（Gateway），且此接口必须在登录成功之后调用。

调用示例

```
private void registerDeviceToCloudButtonClick(){
    String pin = “5d:6t:2c:3s”;
    Int timeoutTime = 90;
    DeviceApi.registerDeviceToCloud(pin , timeoutTime , new ResultsListener<Device>(){
    @Override
    public void onSuccess(Device results) {
    //do something
    }

    @Override
    public void onError(int errorCode) {
    //do something
    }

    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.6.4 注册设备到网关

接口功能

应用程序注册设备到网关，注册成功后，会返回该设备的信息。

接口原型

```
void startDeviceDiscovery(String protocolType, ResultsListener<Device> listener)
```

或

```
void startDeviceDiscovery(String protocolType, int timeoutTime, ResultsListener<Device> listener)
```

参数说明

参数	必选/可选	类型	描述
protocolType	必选	String	设备接入时使用的协议类型，例如：z-wave、zigbee、wps等
timeoutTime	可选	int	设备注册超时时长，单位为秒，默认为60s（不小于30s）
listener	必选	ResultsListener<Device>	接口结果监听器

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
private void startDeviceDiscoveryClick(){
    String protocolType = "Z-Wave";
    int timeoutTime = 60;
    DeviceApi.startDeviceDiscovery(protocolType , timeoutTime , new ResultsListener<Device>(){
    @Override
    public void onSuccess(Device results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
    //do something
    }
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.6.5 查询设备连接状态

接口功能

查询某个设备的连接状态，即设备在线还是离线。

接口原型

```
void checkDeviceConnection(String deviceId, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备标识
listener	必选	ResultsListener	接口结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
private void checkDeviceConnectionClick(){
    String deviceId = “fasf46541231asf”;
    DeviceApi. checkDeviceConnection(deviceId , new ResultsListener(){
    @Override
    public void onSuccess(Object results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
    //do something
    }
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.6.6 查询设备列表

接口功能

查询设备列表，返回设备的详细信息。

接口原型

```
void queryAllDevices(ResultsListener<List<Device>> listener)
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener<List<Device>>	接口结果监听器

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
private void queryAllDevicesClick(){
    DeviceApi.queryAllDevices(new ResultsListener<List<Device>>(){
    @Override
    public void onSuccess(<List<Device>> results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
```

```
    //do something
    }
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.6.7 查询指定设备信息

接口功能

查询某个指定设备的详细信息。

接口原型

```
void queryDevice(String deviceId, ResultsListener<Device> listener)
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备标识
listener	必选	ResultsListener<Device>	接口结果监听器

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
private void queryDeviceClick(){
    String deviceId = “fasf46541231asf”;
    DeviceApi.queryDevice(deviceId , new ResultsListener<Device>(){
    @Override
    public void onSuccess(Device results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
    //do something
    }
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.6.8 删除设备

接口功能

删除设备。

接口原型

```
void unregisterDevice(String deviceId, ResultsListener listener)
```

或

```
void unregisterDevice(String deviceId, int timeoutTime, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备标识
timeoutTime	可选	int	超时时长，单位为秒，默认为60s（不小于30s）
listener	必选	ResultsListener	接口结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
private void unregisterDeviceClick(){
    String deviceId = “fasf46541231asf”;
    int timeoutTime = 60;
    DeviceApi.unregisterDevice(deviceId , timeoutTime , new ResultsListener(){
    @Override
    public void onSuccess(Object results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
    //do something
    }
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.6.9 设备重命名

接口功能

对某个设备重命名。

接口原型

```
void renameDevice(String deviceId, String deviceName, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备标识
deviceName	必选	String	新的设备名称
listener	必选	ResultsListener	接口结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
private void renameDeviceClick(){
    String deviceId = “fasf46541231asf”;
    int deviceName = “newName”;
    DeviceApi.renameDevice(deviceId ,deviceName , new ResultsListener(){
    @Override
    public void onSuccess(Object results) {
    //do something
    }
    @Override
    public void onError(int errorCode) {
    //do something
    }
    @Override
    public void onTimeout() {
    //do something
    }
    });
}
```

7.2.6.10 下发设备控制命令

接口功能

对某个设备下发一条控制命令。

接口原型

```
void sendDeviceControlCommand(String deviceId, ServiceCommand command, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备标识
command	必选	ServiceCommand类	设备服务命令类，类的属性必须根据平台定义的Service profile赋值
listener	必选	ResultsListener	接口结果监听器

返回结果

类型	描述
Void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
ServiceCommand command = new ServiceCommand();
command.setServiceId("serviceId from profile");
command.setCommandName("Light");
Map<String, Object> params = new HashMap<String, Object>();
params.put("status", "ON");
params.put("time", 30);
command.setCommandParameters(params);

DeviceApi.sendDeviceControlCommand("deviceId", command, new ResultsListener(){
@Override
public void onSuccess(Object results) {

}
@Override
public void onError(int errorCode) {

}
@Override
public void onTimeout() {
}
});
```

7.2.6.11 设备升级状态变化广播

接口功能

直连设备接入云后，会检测设备是否需要升级，若设备在升级，则会发送该广播；同时升级完成时，也会发送该广播。

接口原型

```
EVENT_DEVICE_UPGRADE_STATUS_CHANGED
```

参数说明

字段	必选/可选	类型	描述
PARAM_DEVICE	必选	String	广播参数，携带设备信息（Device对象）
PARAM_UPGRADE_STATUS	必选	String	设备升级状态： DEVICE_STATUS_DETAIL_NONE：升级完成 DEVICE_STATUS_DETAIL_UPDATING：正在升级

返回结果

类型	描述
Void	无返回值

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mDeviceUpgradeStatusReceiver,
new IntentFilter(DeviceApi. EVENT_DEVICE_UPGRADE_STATUS_CHANGED));
……
private BroadcastReceiver mDeviceUpgradeStatusReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
Device device = intent. getSerializableExtra(DeviceApi.PARAM_DEVICE);
String upgradeStatus = intent.getStringExtra(DeviceApi.
PARAM_UPGRADE_STATUS);
//do something
}
}
};
……
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mDeviceUpgradeStatusReceiver);
```

7.2.6.12 新设备添加广播

接口功能

应用程序添加新设备，添加成功后，SDK会上报一个新设备添加的广播。

接口原型

```
EVENT_DEVICE_ADDED
```

参数说明

字段	必选/可选	类型	描述
PARAM_DEVICE	必选	String	广播参数，携带已添加的设备的信息（Device对象）

返回结果

类型	描述
Void	无返回值

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mDeviceAddReceiver,
new IntentFilter(DeviceApi. EVENT_DEVICE_ADDED));
.........
private BroadcastReceiver mDeviceAddReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
Device device = intent. getSerializableExtra(DeviceApi.PARAM_DEVICE);
//do something
}
}
};
.........
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mDeviceAddReceiver);
```

7.2.6.13 设备删除广播

接口功能

应用程序删除设备，删除成功后，会收到一个设备删除的广播。

接口原型

```
EVENT_DEVICE_DELETED
```

参数说明

字段	必选/可选	类型	描述
PARAM_DEVICE_ID	必选	String	广播Extra，携带已删除的设备的信息

返回结果

类型	描述
Void	无返回值

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mDeviceDeleteReceiver,
new IntentFilter(DeviceApi. EVENT_DEVICE_DELETED));
……
private BroadcastReceiver mDeviceDeleteReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
String deviceId = intent. getSerializableExtra(DeviceApi.PARAM_DEVICE_ID);
//do something
}
}
};
……
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mDeviceDeleteReceiver);
```

7.2.6.14 设备信息变化广播

接口功能

应用程序操作设备（例如重命名），操作成功后，会收到一个设备信息变化的广播。

接口原型

```
EVENT_DEVICE_INFO_CHANGED
```

参数说明

字段	必选/可选	类型	描述
PARAM_DEVICE	必选	String	广播Extra，携带已有信息变化的设备的信息

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mDeviceInfoChangedReceiver,
new IntentFilter(DeviceApi.EVENT_DEVICE_INFO_CHANGED));
……
private BroadcastReceiver mDeviceInfoChangedReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
```

```
Device device = intent. getSerializableExtra(DeviceApi.PARAM_DEVICE);
//do something
}
}
};
.....
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mDeviceInfoChangedReceiver);
```

7.2.7 摄像头操作

摄像头操作依赖DeviceApi组件，获取SDK媒体播放器、支持开始录像、结束录像、下载录像、抓取录像首帧图等功能。

7.2.7.1 开始视频录像

接口功能

下发控制命令到IP摄像头，开始录制视频

接口原型

```
void startVideoRecord(String deviceId, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备标识
listener	必选	ResultsListener	结果回调

返回结果

类型	描述
void	无返回值

注意事项

设备必须在线。开始录像后，若不调用停止录像接口，系统默认的最长录像时间是两分钟，即录到两分钟后，系统会自动结束本次录像。

调用示例

```
private void startVideoRecordClick(){
    String deviceId = “fasf46541231asf”;
    DeviceApi.startVideoRecord(deviceId, new ResultsListener() {
    @Override
    public void onSuccess(Object results) {
    }
    @Override
    public void onError(int errorCode) {
    }
    @Override
    public void onTimeout() {
    }
```

```
    });
}
```

7.2.7.2 结束视频录像

接口功能

结束摄像头的视频录制

接口原型

```
void stopVideoRecord(String deviceId, ResultsListener listener)
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备标识
listener	必选	ResultsListener	结果回调

返回结果

类型	描述
void	无返回值

注意事项

在调用开始录制视频接口后才能调用

调用示例

```
private void stopVideoRecordClick(){
    String deviceId = “fasf46541231asf”;
    DeviceApi.stopVideoRecord(deviceId, new ResultsListener() {
    @Override
    public void onSuccess(Object results) {
    }
    @Override
    public void onError(int errorCode) {
    }
    @Override
    public void onTimeout() {
    }
    });
}
```

7.2.7.3 下载视频录像

接口功能

从云存储服务器下载视频录像，将下载后的录像保存到SD卡中。

接口原型

```
void downloadRecordVideo(String srcUrl, String desFile, ResultsListener, listenet);
```

参数说明

参数	必选/可选	类型	描述
srcUrl	必选	String	视频下载地址（参考视频录制结果广播通知和常用数据结构章节，获取视频地址）。
desFile	必选	String	保存目录
listenet	可选	Interface	结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void downloadRecordVideoClick(){
    String srcUrl = “http://www.xxx.com/aa/bb”;
    String desFile = “//sdcard/test”;
    DeviceApi.downloadRecordVideo(srcUrl, desFile, new FileDownloadListener() {
    @Override
    public void onProgress ( int downloadedSize, int totalSize ) {
    }
    @Override
    public void onError(int errorCode) {
    }
    Override
    public void onComplete () {
    }
    });
}
```

7.2.7.4 取消正在下载的适配录像

接口功能

取消正在下载的视频录像，并且把已下载的部分数据清除。

接口原型

```
void cancelDownloadingVideoRecord();
```

参数说明

参数	必选/可选	类型	描述
Void			

返回结果

类型	描述
void	无返回值

7.2.7.5 抓取摄像头当前图片

接口功能

抓取摄像头当前图像，成功时返回图像URL。

接口原型

```
void captureImage(DeviceId deviceId, ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备标识
listener	必选	ResultsListener	结果回调

返回结果

类型	描述
String	IP摄像头所抓拍的图片URL

注意事项

本接口必须在登录成功之后调用。

调用示例

```
private void captureImageClick(){
    String deviceId = “achs12fcsf1234”;
    DeviceApi.captureImage(deviceId, new ResultsListener<String>() {
    @Override
    public void onSuccess(String results) {
    }
    @Override
    public void onError(int errorCode) {
    }
    @Override
    public void onTimeout() {
```

```
    }
    });
}
```

7.2.7.6 获取 SDK 媒体播放器

接口功能

本接口用于获取SDK媒体播放器，使用SDK媒体播放器与Andriod系统类似。

接口原型

```
SdkMediaPlayer getMediaPlayer(int type);
```

参数说明

参数	必选/可选	类型	描述
PARAM_VIDEO_RECORDING_INFO	必选	int	定要获取的媒体播放器的类型。如果输入MEDIA_PLAYER_TYPE_LIVE，得到实时视频播放器;如果输入MEDIA_PLAYER_TYPE_VOD，得到视频播放器。

返回结果

类型	描述
Void	无返回值

注意事项

本接口只有在登录成功后才能被调用。

调用示例

```
SdkMediaPlayer mediaPlayer = DeviceApi.getMediaPlayer(DeviceApi. MEDIA_PLAYER_TYPE_LIVE);
```

7.2.7.7 视频录像完成广播

接口功能

上报视屏录制完成后结果通知

接口原型

```
String EVENT_VIDEO_RECORDING_DONE
```

参数说明

参数	必选/可选	类型	描述
PARAM_VIDEO_RECORDING_INFO	必选	Class	广播Extra，携带录制的视频信息

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mRecordVideoReceiver,
new IntentFilter(DeviceApi.EVENT_VIDEO_RECORDING_DONE));
.........
private static BroadcastReceiver mRecordVideoReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
VideoRecordingInfo videoRecordingInfo = intent.getIntExtra(DeviceApi. PARAM_VIDEO_RECORDING_INFO);
//do something
}
}
};
.........
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mRecordVideoReceiver);
```

7.2.7.8 查询云存储信息

接口功能

查询云存储空间信息

接口原型

```
void queryCloudStorageInfo(ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	interface	结果回调

返回结果

类型	描述
void	无返回值

注意事项

无。

调用示例

```
private void queryCloudStorageInfoClick(){
    DeviceApi.queryCloudStorageInfo(new ResultsListener<CloudStorageSpace>() {
    @Override
    public void onSuccess(CloudStorageSpace results) {
    }
    @Override
    public void onError(int errorCode) {
    }
    Override
    public void onTimeout() {
    }
    });
}
```

7.2.7.9 查询视频录制最大时长

接口功能

查询视频录制时，允许录制的最大时长。

接口原型

```
void queryMaxRecordTime(ResultsListener<int> listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener<int>	操作结果回调

返回结果

类型	描述
void	无返回值

7.2.7.10 云存储状态变化广播

接口功能

云存储空间快要满时，会收到这个广播。

接口原型

```
EVENT_CLOUD_STORAGE_STATUS_CHANGED
```

参数说明

字段	必选/可选	类型	描述
PARAM_CLOUD_STORAGE_STATUS	必选	String	携带云存储状态信息

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mCloudStorageChangedReceiver, new
IntentFilter(DeviceApi. EVENT_CLOUD_STORAGE_STATUS_CHANGED));
……
private BroadcastReceiver mCloudStorageChangedReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
Device device = intent. getSerializableExtra(DeviceApi.PARAM_DEVICE);
//do something
}
}
};
……
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mCloudStorageChangedReceiver);
```

7.2.8 视频播放

7.2.8.1 获取播放实时视频对象

接口功能

获取此对象可以查看实时视频，其基本使用方法与系统播放器使用方法相同。

接口原型

```
SdkMediaPlayer getMediaPlayer (int type);
```

参数说明

参数	必选/可选	类型	描述
type	必选	int	参数设置为DeviceApi.MEDIA_PLAYER_TYPE_LIVE,可以获取播放实时视频对象

返回结果

类型	描述
SdkMediaPlayer	实时视频播放对象

注意事项

无

调用示例

```
SdkMediaPlayer mediaPlayer = DeviceApi.getMediaPlayer(DeviceApi. MEDIA_PLAYER_TYPE_LIVE);
```

7.2.8.2 获取播放历史视频对象

接口功能

播放摄像头已录制的视频

接口原型

```
SdkMediaPlayer getMediaPlayer (int type);
```

参数说明

参数	必选/可选	类型	描述
type	必选	int	参数设置为DeviceApi.MEDIA_PLAYER_TYPE_VOD,可以获取播放实时视频对象

返回结果

类型	描述
SdkMediaPlayer	历史视频播放对象

注意事项

无

调用示例

```
SdkMediaPlayer mediaPlayer = DeviceApi.getMediaPlayer(DeviceApi. MEDIA_PLAYER_TYPE_VOD);
```

7.2.8.3 创建音频通道

接口功能

创建音频通道时实现进行语音对讲的条件

接口原型

```
void setEnableAudio(boolean bool);
```

参数说明

参数	必选/可选	类型	描述
bool	必选	boolean	true：创建音频通道 false：关闭音频通道

返回结果

类型	描述
void	无返回值

注意事项

必须在打开实时视频的时候调用该接口，打开音频通道。

调用示例

```
mediaPlayer.setEnableAudio (true);
```

7.2.8.4 实时视频播放数据源

接口功能

查看实时视频时设置的播放数据源

接口原型

```
void setDataSource(deviceID, cameraID);
```

参数	必选/可选	类型	描述
deviceID	必选	String	设备唯一标识
cameraID	必选	String	摄像头的唯一标识，与deviceId一样

返回结果

类型	描述
void	无返回值

注意事项

必须在播放视频前调用

调用示例

```
mediaPlayer.setDataSource(deviceID, cameraID);
```

7.2.8.5 设置播放界面

接口功能

播放视频时的界面

接口原型

```
void setSurface(SurfaceView surfaceView);
```

参数	必选/可选	类型	描述
surfaceView	必选	SurfaceView	播放视频界面

返回结果

类型	描述
void	无返回值

注意事项

必须在播放视频前调用

调用示例

```
mediaPlayer.setSurface(mSurfaceView);
```

7.2.8.6 设置视频播放错误监听器

接口功能

设置播放视频时错误监器

接口原型

```
void setOnErrorListener(SdkMediaPlayer.OnErrorListener listener)
```

参数	必选/可选	类型	描述
listener	必选	SdkMediaPlayer.OnErrorListener	播放视频错误回调

返回结果

类型	描述
void	无返回值

注意事项

必须在播放视频前调用

调用示例

```
mMediaPlayer.setOnErrorListener(new SdkMediaPlayer.OnErrorListener() {
@Override
public boolean onError(SdkMediaPlayer mp, int what, Object extra) {
switch (what) {
case SdkMediaDefinedType.ERROR_VIDEO_IS_BUSY:
//camera busy
break;
case SdkMediaDefinedType.ERROR_RPC_CALL_FAILED:
//start vedio failed
break;
default:
// do something
break;
}
return false;
}
});
```

7.2.8.7 设置视频播放状态信息监听器

接口功能

设置视频播放时，视频播放状态信息监听器

接口原型

```
void setOnInfoListener(SdkMediaPlayer. OnInfoListener listener)
```

参数	必选/可选	类型	描述
listener	必选	SdkMediaPlayer. OnInfoListener	视频信息回调

返回结果

类型	描述
void	无返回值

注意事项

必须在播放视频前调用

调用示例

```
mediaPlayer.setOnInfoListener(new SdkMediaPlayer.OnInfoListener() {
@Override
public boolean onInfo(SdkMediaPlayer mp, int what, Object extra) {
switch (what) {
case SdkMediaDefinedType.INFO_OPENING:
//正在连接视频
break;
case SdkMediaDefinedType.INFO_PREPARED:
//已经连接好视频
break;
case SdkMediaDefinedType.INFO_QOS_DATA:
//QOS信息：画面丢包率，视频质量信息，分辨率等
break;
case SdkMediaDefinedType.INFO_VIDEO_BAD:
// 连续两次视频丢包率超过10%
break;
case SdkMediaDefinedType.INFO_STOPPED
//视频结束
break;
default:
break;
}
return false;
}
});
```

7.2.8.8 设置麦克风采集到的音频音量监听器

接口功能

设置麦克风采集到音频音量大小监听器

接口原型

```
void setOnAudioRecordVolume(SdkMediaPlayer. OnAudioRecordVolume listener)
```

参数	必选/可选	类型	描述
listener	必选	SdkMediaPlayer.OnAudioRecordVolume	音频声波大小回调

返回结果

类型	描述
void	无返回值

注意事项

必须在播放视频前调用

调用示例

```
mediaPlayer.setOnAudioRecordVolume(new SdkMediaPlayer.OnAudioRecordVolume() {
@Override
public boolean onAudioVolume(SdkMediaPlayer mp, int volume) {
return false;
}
});
```

7.2.8.9 视频准备开始

接口功能

可以准备开始播放视频，播放视频时需先调用该接口

接口原型

```
void prepareAsync();
```

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
mediaPlayer. prepareAsync();
```

7.2.8.10 播放视频

接口功能

播放视频

接口原型

```
void start();
```

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
mediaPlayer. start();
```

7.2.8.11 结束视频播放

接口功能

结束视频播放

接口原型

```
void stop();
```

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
mediaPlayer. stop();
```

7.2.8.12 暂停视频播放

接口功能

暂停视频播放

接口原型

```
void pause();
```

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
mediaPlayer. pause();
```

7.2.8.13 启动视频对讲

接口功能

可以在客户端与摄像头进行语音对讲

接口原型

```
void startAudioRecord();
```

参数	必选/可选	类型	描述
无	无	无	无

返回结果

类型	描述
void	无返回值

注意事项

必须在创建了音频通道后才有效

调用示例

```
mediaPlayer. startAudioRecord();
```

7.2.8.14 结束视频对讲

接口功能

结束客户端与摄像头进行语音对讲

接口原型

```
void stopAudioRecord();
```

参数	必选/可选	类型	描述
无	无	无	无

返回结果

类型	描述
void	无返回值

注意事项

在启动与摄像头对讲一段语音后，调用；

调用示例

```
mediaPlayer. stopAudioRecord();
```

7.2.8.15 获取历史视频本地播放路径

接口功能

历史视频下载到系统相册后，生成的一个FileUrl路径，如果历史视频未下载，则返回空。

接口原型

```
String getLocalHistoryVideoPath(String videoUrl, String desFile);
```

参数	必选/可选	类型	描述
videoUrl	必选	String	历史视频Url
desFile	必选	String	视频存储的文件夹

返回结果

类型	描述
String	本地历史视频播放路径

注意事项

无

调用示例

```
private void getLocalHistoryVideoPathClick(){
    String videoUrl = “http://www.xxx.com/aa/bb”;
    String desFile = “//sdcard/test”;
    return DeviceApi.getLocalHistoryVideoPath(videoUrl,desFile);
}
```

7.2.9 事件处理

包含事件查询、删除接口的API

7.2.9.1 初始化

接口功能

该接口建议在APP登陆成功后进行调用，提供了事件实时上报的广播推送。

接口原型

```
void init(Context context);
```

参数说明

参数	必选/可选	类型	描述
context	必选	Context	Android系统上下文

返回结果

类型	描述
void	无返回值

注意事项

在调用事件API接口之前，必须先进行初始化，即可调用本接口。

调用示例

```
EventApi.init(getApplicationContext());
```

7.2.9.2 销毁

接口功能

事件API销毁，释放内存

接口原型

```
void destroy();
```

参数说明

参数	必选/可选	类型	描述
无			

返回结果

类型	描述
void	无返回值

注意事项

应用程序退出前，必须销毁事件API。

调用示例

```
EventApi.destroy();
```

7.2.9.3 查询所有事件

接口功能

按时间倒序分页查询所有事件的功能。

接口原型

```
void queryEverythingEvents(int pageNumber, int pageSize, ResultListener<List<Event> listener);
```

参数说明

参数	必选/可选	类型	描述
pageNumber	必选	Int	页码
pageSize	必选	Int	每页容纳事件数量

参数	必选/可选	类型	描述
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
EventApi.queryEverythingEvents(0, 40, new ResultsListener<List<Event>>() {
@Override
public void onSuccess(List<Event> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.9.4 查询设备事件

接口功能

按时间倒序分页查询设备事件的功能。

接口原型

```
void queryDeviceEvents(String deviceId, int pageNumber, int pageSize, ResultListener<List<Event>
listener);
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备Id
pageNumber	必选	Int	页码
pageSize	必选	Int	每页容纳事件数量
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
String deviceId = “ccc677dc-7495-40f4-b7b1-08711fb3470d”;
EventApi.queryDeviceEvents(String deviceId, 0, 40, new ResultsListener<List<Event>>() {
@Override
public void onSuccess(List<Event> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.9.5 查询录像事件

接口功能

提供了事件按近远顺序分页查询录像事件的功能。

接口原型

```
void queryRecordingEvents(int pageNumber, int pageSize, ResultListener<List<Event> listener);
```

参数说明

参数	必选/可选	类型	描述
pageNumber	必选	Int	页码
pageSize	必选	Int	每页容纳事件数量
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
EventApi.queryRecordingEvents(0, 40, new ResultsListener<List<Event>>() {
@Override
public void onSuccess(List<Event> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.9.6 查询规则事件

接口功能

提供了事件按近远顺序分页查询规则事件的功能。

接口原型

```
void queryRuleEvents(int pageNumber, int pageSize, ResultListener<List<Event> listener);
```

参数说明

参数	必选/可选	类型	描述
pageNumber	必选	Int	页码
pageSize	必选	Int	每页容纳事件数量
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
EventApi.queryRuleEvents(0, 40, new ResultsListener<List<Event>>() {
@Override
public void onSuccess(List<Event> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.9.7 删除所有事件

接口功能

提供了一次性删除所有事件的功能。

接口原型

```
void deleteAllEvents(ResultListener<Class<Void>> listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultListener	设置回调,对回调结果做处理

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用,该接口执行成功后，包括存储的图像和录像数据都会被删除且无法恢复。

调用示例

```
EventApi.deleteAllEvents(new ResultsListener<List<Void>>() {
@Override
public void onSuccess(List< Void > results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.9.8 删除指定事件

接口功能

提供了删除部分事件功能。

接口原型

```
void deleteEvents(List<String> eventIds, ResultListener <Class<Void>> listener);
```

参数说明

参数	必选/可选	类型	描述
eventIds	必选	List	待删除事件Id的集合
listener	必选	ResultListener	设置回调,对回调结果做处理

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用,该接口执行成功后，包括存储的图像和录像数据都会被删除且无法恢复。

调用示例

```
List<String> eventIds = new ArrayList<>();
eventIds.add(“f988f52a-63c8-48cd-a5b3-0a4306533afd”);
EventApi.deleteEvents(eventIds, new ResultsListener<Class<Void>>() {
@Override
public void onSuccess(Class< Void > results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.9.9 新事件消息广播

接口功能

当有设备状态变更和认为操作产生历史事件记录时会生成一条广播。

接口原型

```
String EVENT_NEW_EVENT_INCOMING
String PARAM_EVENT
```

参数说明

字段	必选/可选	类型	描述
PARAM_EVENT	必选	String	取出事件实例的key值

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mNewEventReceiver,
EventApi.EVENT_NEW_EVENT_INCOMING);
private BroadcastReceiver mNewEventReceiver = new BroadcastReceiver () {
@Override
public void on Receiver (Context context, Intent intent) {
Event newEvent = intent.getSerializableExtra(EventApi.PARAM_EVENT);
}
}
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mNewEventReceiver);
```

7.2.10 规则处理

规则处理依赖RuleApi组件，包括规则创建、删除、编辑、激活去激活、手动触发、广播上报等功能。

7.2.10.1 初始化

接口功能

该接口建议在APP登陆成功后进行调用，提供了规则实时上报的广播推送。

接口原型

```
void init(Context context);
```

参数说明

参数	必选/可选	类型	描述
context	必选	Context	Android系统上下文

返回结果

类型	描述
void	无返回值

注意事项

在调用规则API接口之前，必须先进行初始化，即可调用本接口。

调用示例

```
RuleApi.init(getApplicationContext());
```

7.2.10.2 销毁

接口功能

规则API销毁，释放内存

接口原型

```
void destroy();
```

参数说明

参数	必选/可选	类型	描述
无			

返回结果

类型	描述
void	无返回值

注意事项

应用程序退出前，必须销毁规则API。

调用示例

```
RuleApi.destroy();
```

7.2.10.3 创建规则

接口功能

创建规则

接口原型

```
void createRule(Rule rule, ResultListener<Rule> listener);
```

参数说明

参数	必选/可选	类型	描述
rule	必选	Rule	需要创建的规则
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用

调用示例

```
Rule rule = new Rule();
…
RuleApi. createRule(rule, new ResultsListener<Rule>() {
@Override
public void onSuccess(List<Rule> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.10.4 更新规则

接口功能

更新规则

接口原型

```
void updateRule(Rule rule, ResultListener<Rule> listener);
```

参数说明

参数	必选/可选	类型	描述
rule	必选	Rule	需要更新的规则
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用

调用示例

```
Rule rule = new Rule ();
…
RuleApi. updateRule (rule, new ResultsListener<Rule>() {
@Override
public void onSuccess(List<Rule> results) {
}
@Override
public void onError(int errorCode) {
```

```
}
@Override
public void onTimeout() {
}
});
```

7.2.10.5 查询所有规则

接口功能

查询所有规则

接口原型

```
void queryAllRule(ResultListener<List<Rule>> listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用

调用示例

```
RuleApi. queryAllRule( new ResultsListener<List<Rule>>() {
@Override
public void onSuccess(List<List<Rule>> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.10.6 查询指定规则

接口功能

查询指定规则

接口原型

```
void queryRuleById(int ruleId, ResultListener<Rule> listener);
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	int	规则id
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用

调用示例

```
int ruleId = “f988f52a-63c8-48cd-a5b3-0a4306533afd”;
RuleApi. queryRuleById (ruleId,  new ResultsListener<Rule>() {
@Override
public void onSuccess(List<Rule> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.10.7 激活规则

接口功能

激活规则，规则可以触发。

接口原型

```
void activeRule(int ruleId, ResultListener<Class<Void>> listener);
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	int	规则id
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
RuleApi. activeRule(ruleId,  new ResultsListener<Class<Void>>() {
@Override
public void onSuccess(Class<Void> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.10.8 去激活规则

接口功能

去激活规则，规则不可以触发。

接口原型

```
void inactiveRule(int ruleId, ResultListener<Class<Void>> listener);
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	int	规则id
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
RuleApi. inactiveRule(ruleId,  new ResultsListener<Class<Void>>() {
@Override
```

```
public void onSuccess(Class<Void> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.10.9 手动触发规则

接口功能

创建的手动场景（规则）可以调用该接口直接触发。

接口原型

```
void runRule(int ruleId, ResultListener<Class<Void>> listener);
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	int	规则id
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
RuleApi. runRule(ruleId,  new ResultsListener<Class<Void>>() {
@Override
public void onSuccess(Class<Void> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.10.10 批量激活规则

接口功能

批量激活多个规则，多个规则可以触发。

接口原型

```
void batchActiveRule(List<Integer> ruleIds, ResultListener<Class<Void>> listener);
```

参数说明

参数	必选/可选	类型	描述
ruleIds	必选	List	规则id集合
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
List<Integer> ruleIds = new ArrayList<>();
…
RuleApi. batchActiveRule(ruleIds,  new ResultsListener<Class<Void>>() {
@Override
public void onSuccess(Class<Void> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.10.11 批量去激活规则

接口功能

批量去激活规则，多个规则不可以触发。

接口原型

```
void batchInactiveRule(List<Integer> ruleIds, ResultListener<Class<Void>> listener);
```

参数说明

参数	必选/可选	类型	描述
ruleIds	必选	List	规则id集合

参数	必选/可选	类型	描述
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
List<Integer> ruleIds = new ArrayList<>();
…
RuleApi. batchInactiveRule(ruleIds,  new ResultsListener<Class<Void>>() {
@Override
public void onSuccess(Class<Void> results) {
}
@Override
public void onError(int errorCode) {
}
@Override
public void onTimeout() {
}
});
```

7.2.10.12 删除规则

接口功能

删除指定规则

接口原型

```
void deleteRule(ruleId, ResultListener<Class<Void>> listener);
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	int	规则id
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
RuleApi. deleteRule(ruleId,  new ResultsListener<Class<Void>>() {
@Override
public void onSuccess(Class<Void> results) {

}

@Override
public void onError(int errorCode) {

}

@Override
public void onTimeout() {

}
});
```

7.2.10.13 新规则被创建消息广播

接口功能

在同一个homeId下，当A用户创建了新规则后，除了A用户以外，所有用户都会收到新规则被创建的消息广播。

接口原型

```
String EVENT_RULE_CREATED
String PARAM_RULE
```

参数说明

字段	必选/可选	类型	描述
PARAM_RULE	必选	String	取出新规则实例的key值

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mNewRuleCreatedReceiver, RuleApi.
EVENT_RULE_CREATED);
```

```
private BroadcastReceiver mNewRuleCreatedReceiver = new BroadcastReceiver () {
@Override
public void on Receiver (Context context, Intent intent) {
Rule rule = intent.getSerializableExtra(RuleApi.PARAM_RULE);
}
}
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mNewRuleCreatedReceiver);
```

7.2.10.14 规则更新消息广播

接口功能

在同一个homeId下，当A用户编辑某个规则后，除了A用户以外，所有用户都会收到规则更新的消息广播。

接口原型

```
String EVENT_RULE_INFO_CHANGED
String PARAM_RULE
```

参数说明

字段	必选/可选	类型	描述
PARAM_RULE	必选	String	取出更新过规则实例的key值

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mRuleUpdateReceiver, RuleApi.
EVENT_RULE_INFO_CHANGED);
private BroadcastReceiver mRuleUpdateReceiver = new BroadcastReceiver () {
@Override
public void on Receiver (Context context, Intent intent) {
Rule rule = intent.getSerializableExtra(RuleApi.PARAM_RULE);
}
}
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mRuleUpdateReceiver);
```

7.2.10.15 删除规则消息广播

接口功能

在同一个homeId下，当A用户删除某个规则后，除了A用户以外，所有用户都会收到规则被删除的消息广播。

接口原型

```
String EVENT_RULE_DELETED
String PARAM_RULE
```

参数说明

字段	必选/可选	类型	描述
PARAM_RULE	必选	String	取出被删除规则id实例的key值

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mRuleDeletedReceiver, RuleApi.
EVENT_RULE_DELETED);
private BroadcastReceiver mRuleDeletedReceiver = new BroadcastReceiver () {
@Override
public void on Receiver (Context context, Intent intent) {
int ruleId = intent.getSerializableExtra(RuleApi.PARAM_RULE);
}
}
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mRuleDeletedReceiver);
```

7.2.10.16 查询规则录制时长

接口功能

查询规则录制时长

接口原型

```
void queryRuleRecordTime(ResultsListener<String> listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultsListener<String>	操作结果回调

返回结果

类型	描述
void	无返回值

7.2.11 离线推送

离线推送(PushApi)，提供离线推送相关接口，当前支持的是Google的GCM。要激活离线推送功能后，需要将APP在Google服务上注册，并获取Api Key和senderId。

1. 激活离线推送功能

接口功能

激活离线推送功能，能接收离线消息

接口原型

```
void enablePush(String apiKey, String senderId)
```

参数说明

参数	必选/可选	类型	描述
apiKey	必选	String	在Google上注册离线推送参数Api key
senderId	必选	String	在Google上注册离线推送参数senderId

返回结果

类型	描述
广播	通过EVENT_ENABLE_PUSH_RESULT广播通知

注意事项

必须在登录成功后调用，该接口的执行结果通过广播EVENT_ENABLE_PUSH_RESULT上报，调用该接口前需要注册监听这个广播。

调用示例

```
PushApi.enablePush(apiKey, senderId);
```

2. 离线推送激活结果广播

接口功能

离线推送激活结果通知

接口原型

```
String EVENT_ENABLE_PUSH_RESULT
```

参数说明

参数	必选/可选	类型	描述
PARAM_ENABLE_PUSH_RESULT_FAILED_REASON	必选	Int	广播Extra，激活失败错误码
PARAM_ENABLE_PUSH_RESULT	必选	String	广播Extra 激活结果

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mEnablePushReceiver,
new IntentFilter(PushApi.EVENT_ENABLE_PUSH_RESULT);

.....

private static BroadcastReceiver mEnablePushReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
int userId =
intent.getIntExtra(PushAPi. PARAM_ENABLE_PUSH_RESULT_FAILED_REASON);
String userId = intent.getIntExtra(PushApi. PARAM_ENABLE_PUSH_RESULT);
//do something
}
}
};

.....

LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mEnablePushReceiver);
```

3. 离线推送消息广播

接口功能

离线推送激活结果通知

接口原型

```
String EVENT_RECEIVE_OFFLINE_PUSH
```

参数说明

参数	必选/可选	类型	描述
PARAM_OFFLINE_PUSH_MESSAGE	必选	Int	广播Extra，离线消息内容
PARAM_OFFLINE_PUSH_EVENT_ID	必选	String	广播Extra 事件Id

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mOfflineMessageReceiver,
new IntentFilter(PushApi.EVENT_RECEIVE_OFFLINE_PUSH);

.........

private static BroadcastReceiver mOfflineMessageReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
StringuserId =
intent.getIntExtra(PushAPi. PARAMOFFLINE_PUSH_MESSAGE);
StringuserId = intent.getIntExtra(PushApi. PARAM_OFFLINE_PUSH_EVENT_ID);
//do something
}
}
};

.........

LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mOfflineMessageReceiver);
```

7.2.11.1 新节点 1

7.2.11.2 新节点 2

7.2.11.3 新节点 3

7.2.12 房间管理

房间管理(RoomApi)，提供房间管理的相关接口，当前支持的是房间的增删改查，以及广播相关的通知。

7.2.12.1 初始化

接口功能

初始化RoomApi

接口原型

```
void init(Context context)
```

参数说明

参数	必选/可选	类型	描述
context	必选	Context	Android 系统上下文

返回结果

类型	描述
void	无返回值

注意事项

调用前需要确保已经调用RoomApi.init(Context context)初始化了RoomApi。调用房间管理的初始化接口后，房间管理组件的其它接口才可用。

调用示例

```
RoomApi.init(getApplicationContext());
```

7.2.12.2 销毁

接口功能

规则API销毁，释放内存

接口原型

```
void destroy();
```

参数说明

参数	必选/可选	类型	描述
无			

返回结果

类型	描述
void	无返回值

注意事项

应用程序退出前，必须销毁规则API。

调用示例

```
RuleApi.destroy();
```

7.2.12.3 创建房间

接口功能

创建房间

接口原型

```
void createRoom(Room room, ResultsListener<Room> listener);
```

参数说明

参数	必选/可选	类型	描述
room	必选	Room	需要创建的规则
listener	必选	ResultsListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void createRoomButtonClick(){
    Room room = new Room ();
    RoomApi. createRoom(room, new ResultsListener<Rule>() {
    @Override
    public void onSuccess(List<Rule> results) {
    }
    @Override
    public void onError(int errorCode) {
        switch (errorCode) {
            case RoomApi.ERROR_CODE_ROOM_ALREADY_EXISTED:
              //do something
```

```
                break;
            case RoomApi.ERROR_CODE_ROOM_NOT_FOUND:
             //do something
                break;
            case RoomApi.ERROR_CODE_ROOM_DEFAULT_FORBIDEN:
             //do something
                break;
             default:
            break;
        }
     }
     @Override
     public void onTimeout() {
    }
    });
}
```

7.2.12.4 删除房间

接口功能

删除指定房间

接口原型

```
void deleteRoom(int roomId, ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
roomId	必选	int	房间ID
listener	必选	ResultsListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void createRoomButtonClick(){
    String roomId = “695324”;
    RoomApi. deleteRoom (roomId,  new ResultsListener<Class<Void>>() {
    @Override
    public void onSuccess(Class<Void> results) {
    }
    @Override
    public void onError(int errorCode) {
       switch (errorCode) {
            case RoomApi.ERROR_CODE_ROOM_ALREADY_EXISTED:
               //do something
```

```
                break;
            case RoomApi.ERROR_CODE_ROOM_NOT_FOUND:
                //do something
                break;
            case RoomApi.ERROR_CODE_ROOM_DEFAULT_FORBIDEN:.
                //do something
                 break;
            default:
                break;
          }
    }
    @Override
    public void onTimeout() {
     }
    });
}
```

7.2.12.5 查询房间列表

接口功能

查询所有房间

接口原型

```
void queryAllRooms(ResultsListener<List<Room>> listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultListener	设置回调，回调中的参数为查询结果

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void queryRoomButtonClick(){
    RoomApi. queryAllRooms( new ResultsListener<List< Room >>() {
    @Override
    public void onSuccess(List<List< Room >> results) {
    }
    @Override
    public void onError(int errorCode) {
        switch (errorCode) {
              case RoomApi.ERROR_CODE_ROOM_ALREADY_EXISTED:
               //do something
               break;
              case RoomApi.ERROR_CODE_ROOM_NOT_FOUND:
               //do something
               break;
```

```
                case RoomApi.ERROR_CODE_ROOM_DEFAULT_FORBIDEN:
                //do something
                 break;
                default:
                 break;
            }
        }
        @Override
        public void onTimeout() {
        }
        });
}
```

7.2.12.6 更新房间

接口功能

更新指定房间信息

接口原型

```
void updateRoom(Room room, ResultsListener<Room> listener);
```

参数说明

参数	必选/可选	类型	描述
room	必选	Room	待更新房间实体
listener	必选	ResultsListener	

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void queryAllRoomsClick(){
    Room room = new Room();
    RoomApi. queryAllRooms(room,  new ResultsListener<List< Room >>() {
     @Override
     public void onSuccess(List<List< Room >> results) {
     }
     @Override
     public void onError(int errorCode) {
         switch (errorCode) {
               case RoomApi.ERROR_CODE_ROOM_ALREADY_EXISTED:
                //do something
                break;
               case RoomApi.ERROR_CODE_ROOM_NOT_FOUND:
                //do something
                break;
               case RoomApi.ERROR_CODE_ROOM_DEFAULT_FORBIDEN:
```

```
                //do something
                 break;
                default:
                 break;
            }
        }
        @Override
        public void onTimeout() {
        }
        });
}
```

7.2.12.7 添加设备到房间

接口功能

添加设备到房间，可以把多个设备添加到同一房间，也可以把多个设备各自添加到不同的房间。

接口原型

```
void addDevicesToRoom(Map<deviceId, roomId> map, ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
map	必选	Map<deviceId, roomId>	设备ID与房间ID映射关系map
listener	必选	ResultsListener	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void updateRoomClick(){
     Map<Integer, Integer > map = new Hash Map< Integer, Integer>()
     RoomApi. updateRoom(map, new ResultsListener<List< Room >>() {
     @Override
     public void onSuccess(List<List< Room >> results) {
     }
     @Override
     public void onError(int errorCode) {
         switch (errorCode) {
               case RoomApi.ERROR_CODE_ROOM_ALREADY_EXISTED:
                //do something
                break;
               case RoomApi.ERROR_CODE_ROOM_NOT_FOUND:
                //do something
                break;
```

```
                case RoomApi.ERROR_CODE_ROOM_DEFAULT_FORBIDEN:
                //do something
                 break;
                default:
                 break;
            }
        }
        @Override
        public void onTimeout() {
        }
        });

}
```

7.2.12.8 创建房间广播通知

接口功能

在同一个homeId下，当A用户创建了新房间后，除了A用户以外，所有用户都会收到新房间被创建的消息广播。

接口原型

```
String EVENT_ROOM_CREATED
String PARAM_ROOM
```

参数说明

字段	必选/可选	类型	描述
PARAM_ROOM	必选	String	创建的房间实体

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mNewRoomCreatedReceiver,
RoomApi.EVENT_ROOM_CREATED);
private BroadcastReceiver mNewRoomCreatedReceiver = new BroadcastReceiver () {
@Override
public void on Receiver (Context context, Intent intent) {
Room room = intent.getSerializableExtra(RoomApi.PARAM_ROOM);
}
}
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mNewRoomCreatedReceiver);
```

7.2.12.9 删除房间广播通知

接口功能

在同一个homeId下，当A用户删除某个房间后，除了A用户以外，所有用户都会收到房间被删除的消息广播。

接口原型

```
String EVENT_ROOM_DELETED  String PARAM_ROOM_ID
```

参数说明

字段	必选/可选	类型	描述
PARAM_ROOM_ID	必选	String	取出被删除房间id实例的key值

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mRoomDeletedReceiver,
RoomApi.EVENT_ROOM_DELETED);
private BroadcastReceiver mRoomDeletedReceiver = new BroadcastReceiver () {
@Override
public void on Receiver (Context context, Intent intent) {
int roomId = intent.getSerializableExtra(RoomApi.PARAM_ROOM_ID);
}
}
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mRoomDeletedReceiver);
```

7.2.12.10 更新房间广播通知

接口功能

在同一个homeId下，当A用户更新某个房间后，除了A用户以外，所有用户都会收到房间被更新的消息广播。

接口原型

```
String EVENT_ROOM_UPDATED  String PARAM_ROOM
```

参数说明

字段	必选/可选	类型	描述
PARAM_ROOM	必选	String	更新的房间实体

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mRoomUpDatedReceiver,
RoomApi.EVENT_ROOM_UPDATED);
```

```
private BroadcastReceiver mRoomUpDatedReceiver = new BroadcastReceiver () {
@Override
public void on Receiver (Context context, Intent intent) {
Room room = intent.getSerializableExtra(RoomApi.PARAM_ROOM);
}
}
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mRoomUpDatedReceiver);
```

7.2.13 用户设置

用户设置(SettingsApi)，包括查询事件通知设置、开启\关闭事件通知设置、查询与修改云存储空间满时自动清理开关配置、设置自动收集日志配置等功能。

7.2.13.1 初始化

接口功能

初始化SettingsApi

接口原型

```
void init(Context context)
```

参数说明

参数	必选/可选	类型	描述
context	必选	Context	Android 系统上下文

返回结果

类型	描述
Void	无返回值

注意事项

调用前需要确保已经调用SettingsApi.init(Context context)初始化了SettingsApi。调用设置的初始化接口后，房间管理组件的其它接口才可用。

调用示例

```
SettingsApi.init(getApplicationContext());
```

7.2.13.2 销毁

接口功能

规则API销毁，释放内存

接口原型

```
void destroy();
```

参数说明

参数	必选/可选	类型	描述
无			

返回结果

类型	描述
void	无返回值

注意事项

应用程序退出前，必须销毁设置API。

调用示例

```
SettingsApi.destroy();
```

7.2.13.3 设置查询事件的通知

接口功能

设置查询事件的通知

接口原型

```
void queryEventNotificationSetting(ResultsListener<Map<String, String>> listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultListener	设置回调，回调中的参数为查询结果(Map的Key为事件的类型，即eventType；Map的Value为事件的通知设置激活状态，activated激活，inactivated未激活)

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void queryEventNotificationSettingClick(){
     SettingsApi.queryEventNotificationSetting ( new ResultsListener<Map<String, String>>() {
     @Override
     public void onSuccess(List<Map<String, String>> results) {
     }
     @Override
     public void onError(int errorCode) {
     }
     @Override
     public void onTimeout() {
     }
     });
}
```

7.2.13.4 设置激活事件的通知

接口功能

设置激活事件的通知

接口原型

```
void enableEventNotificationSetting(List<String> eventTypes, ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
eventTypes	必选	List<String>	事件的类型
listener	必选	ResultListener	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void enableEventNotificationSettingClick(){
    List eventTypes = new ArrayList();
    SettingsApi.enableEventNotificationSetting (eventTypes,  new ResultsListener() {
    @Override
    public void onSuccess() {
    }
    @Override
    public void onError(int errorCode) {
    }
    @Override
    public void onTimeout() {
    }
    });
}
```

7.2.13.5 设置去激活事件的通知

接口功能

设置去激活事件的通知

接口原型

```
void disableEventNotificationSetting(List<String> eventTypes, ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
eventTypes	必选	List<String>	事件的类型
listener	必选	ResultListener	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void disableEventNotificationSettingClick(){
    List eventTypes = new ArrayList();
    SettingsApi.disableEventNotificationSetting(eventTypes,  new ResultsListener() {
    @Override
    public void onSuccess() {
    }
    @Override
    public void onError(int errorCode) {
    }
    @Override
    public void onTimeout() {
    }
    });
}
```

7.2.13.6 设置云存储空间满时自动清理配置通知

接口功能

设置云存储空间满时自动清理配置开关

接口原型

```
void setCloudStorageSpaceAutoClean(boolean autoClean, ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
autoClean	必选	boolean	自动清理开关
listener	必选	ResultListener	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void setCloudStorageSpaceAutoCleanClick(){
    boolean autoClean = true;
    SettingsApi. setCloudStorageSpaceAutoClean (autoClean,  new ResultsListener() {
    @Override
    public void onSuccess() {
    }
    @Override
    public void onError(int errorCode) {
    }
    @Override
    public void onTimeout() {
    }
    });
}
```

7.2.13.7 设置查询云存储空间自动清理开关

接口功能

设置查询云存储空间自动清理开关

接口原型

```
void queryCloudStorageSpaceAutoClean(ResultsListener<boolean> listener);
```

参数说明

参数	必选/可选	类型	描述
listener	必选	ResultListener	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void queryCloudStorageSpaceAutoCleanClick(){
    SettingsApi.queryCloudStorageSpaceAutoClean ( new ResultsListener<boolean> () {
    @Override
    public void onSuccess(boolean isClear) {
    }
    @Override
    public void onError(int errorCode) {
    }
    @Override
    public void onTimeout() {
    }
    });
}
```

7.2.13.8 设置自动收集日志配置开关

接口功能

设置自动收集日志配置开关

接口原型

```
void setAutoUploadLog(boolean autoUpload, ResultsListener listener);
```

参数说明

参数	必选/可选	类型	描述
autoUpload	必选	boolean	自动上传日志开关
listener	必选	ResultListener	操作结果回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
private void setAutoUploadLogClick(){
    boolean autoUpload = true;
```

```
    SettingsApi.setAutoUploadLog (autoUpload,  new ResultsListener() {
    @Override
    public void onSuccess() {
    }
    @Override
    public void onError(int errorCode) {
    }
    @Override
    public void onTimeout() {
    }
    });
}
```

7.2.13.9 云存储空间自动清理开关变化广播

接口功能

云储存空间自动清理开关变化后上报的广播。

接口原型

```
String EVENT_CLOUD_STORAGE_SPACE_AUTO_CLEAN_CHANGED
String PARAM_AUTO_CLEAN
```

参数说明

字段	必选/可选	类型	描述
PARAM_AUTO_CLEAN	必选	String	云储存自动清理开关

返回结果

无

注意事项

无

消息示例

```
LocalBroadcastManager.getInstance(mContext).registerReceiver(mAntoCleanChangeReceiver,
SettingsApi.EVENT_CLOUD_STORAGE_SPACE_AUTO_CLEAN_CHANGED);
private BroadcastReceiver mAntoCleanChangeReceiver = new BroadcastReceiver () {
@Override
public void on Receiver (Context context, Intent intent) {
String autoClean = intent.getSerializableExtra(SettingsApi.PARAM_AUTO_CLEAN);
}
}
LocalBroadcastManager.getInstance(mContext).unregisterReceiver(mAntoCleanChangeReceiver);
```

7.2.14 本地存储

本地存储(LocalStorageApi)，提供本地数据的查询、存储相关接口，当前支持的是房间、事件、规则、快捷方式和设备的查询和存储功能。

7.2.14.1 初始化

接口功能

初始化LocalStorageApi

接口原型

```
void init(Context context)
```

参数说明

参数	必选/可选	类型	描述
context	必选	Context	Android 系统上下文

返回结果

类型	描述
Void	无返回值

注意事项

调用前需要确保已经调用LocalStorageApi.init(Context context)初始化了LocalStorageApi。调用设置的初始化接口后，数据库组件的其它接口才可用。

调用示例

```
LocalStorageApi.init(getApplicationContext());
```

7.2.14.2 销毁

接口功能

规则API销毁，释放内存

接口原型

```
void destroy();
```

参数说明

参数	必选/可选	类型	描述
无			

返回结果

类型	描述
void	无返回值

注意事项

应用程序退出前，必须销毁数据库API。

调用示例

```
LocalStorageApi.destroy();
```

7.2.14.3 获取所有房间列表

接口功能

获取所有房间列表（排好序的）

接口原型

```
List<Room> getAllRooms();
```

参数说明

参数	必选/可选	类型	描述
void			

返回结果

类型	描述
List<Room> 房间列表实体数组	接口内部排序

注意事项

无

调用示例

```
无
```

7.2.14.4 根据房间标识获取单个房间

接口功能

根据roomId查询房间

接口原型

```
Room getRoom(int roomId);
```

参数说明

参数	必选/可选	类型	描述
roomId	必选	int	房间Id

返回结果

类型	描述
Room房间实体	房间实体

注意事项

无

调用示例

```
无
```

7.2.14.5 保存排好序的房间标识数组

接口功能

保存排好序的房间标识数组

接口原型

```
void saveSortedRoomIds(List<int> roomIds);
```

参数说明

参数	必选/可选	类型	描述
roomIds	必选	List<int>	排好序的房间Id数组

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
无
```

7.2.14.6 获取所有设备列表

接口功能

获取所有设备列表

接口原型

```
List<Device> getAllDevices();
```

参数说明

参数	必选/可选	类型	描述
void			

返回结果

类型	描述
List<Device> 设备列表实体数组	接口内部排序

注意事项

无

调用示例

```
无
```

7.2.14.7 根据设备类型获取设备列表

接口功能

根据设备类型来查询某类型的设备列表

接口原型

```
List<Device> getDevicesByDeviceTypes(List<String> deviceTypes);
```

参数说明

参数	必选/可选	类型	描述
deviceTypes	必选	List<String>	设备类型数组

返回结果

类型	描述
List<Device> 设备列表实体数组	

注意事项

无

调用示例

```
无
```

7.2.14.8 根据设备标识获取单个设备

接口功能

根据设备ID查询某个设备

接口原型

```
Device getDevice(String deviceId);
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识Id

返回结果

类型	描述
Device device设备实体类	

注意事项

无

调用示例

```
无
```

7.2.14.9 保存排好序的设备标识数组

接口功能

保存排好序的设备Id数组

接口原型

```
void saveSortedDeviceIds(List<int> deviceIds);
```

参数说明

参数	必选/可选	类型	描述
deviceIds	必选	List<int>	排好序的设备Id数组

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
无
```

7.2.14.10 判断设备是否存在规则中

接口功能

根据设备ID判断设备是否在规则中

接口原型

```
boolean isDeviceExistedInRule(String deviceId);
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识Id

返回结果

类型	描述
boolean	true：设备在规则中，false：反之

注意事项

无

调用示例

```
无
```

7.2.14.11 获取所有规则列表

接口功能

获取所有规则列表

接口原型

```
List<Rule> getAllRules();
```

参数说明

参数	必选/可选	类型	描述
void			

返回结果

类型	描述
List<Rule>规则实体列表数组	接口内部排序

注意事项

无

调用示例

```
无
```

7.2.14.12 根据规则标识获取单条规则

接口功能

根据规则ID查询规则

接口原型

```
Rule getRule(int ruleId);
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	String	规则唯一标识ruleId

返回结果

类型	描述
Rule rule实体类	实体类

注意事项

无

调用示例

```
无
```

7.2.14.13 保存排好序的规则标识数组

接口功能

保存排好序的规则ID数组

接口原型

```
void saveSortedRuleIds(List<int> ruleIds);
```

参数说明

参数	必选/可选	类型	描述
ruleIds	必选	List<int>	排好序的规则Id数组

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
无
```

7.2.14.14 获取所有事件列表

接口功能

查询所有事件列表

接口原型

```
List<Event> getAllEvents();
```

参数说明

参数	必选/可选	类型	描述
void			

返回结果

类型	描述
List<Event> 事件实体类列表数组	

注意事项

无

调用示例

```
无
```

7.2.14.15 根据事件标识获取单条事件

接口功能

根据事件ID查询事件

接口原型

```
Event getEvent(int eventId);
```

参数说明

参数	必选/可选	类型	描述
eventId	必选	String	事件唯一标识

返回结果

类型	描述
Event event实体类	

注意事项

无

调用示例

```
无
```

7.2.14.16 根据设备标识获取设备事件列表

接口功能

根据设备ID查询该设备的事件

接口原型

```
List<Event> getDeviceEvents(String deviceId);
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	String	设备的唯一标识

返回结果

类型	描述
List<Event>	事件实体类列表数组

注意事项

无

调用示例

```
无
```

7.2.14.17 获取所有用户列表

接口功能

从本地存储中获取所有的用户列表。

接口原型

```
List<User> getAllUsers();
```

参数说明

参数	必选/可选	类型	描述
void			

返回结果

类型	描述
List<User>	返回所有用户实体类列表数组

注意事项

无

调用示例

```
无
```

7.2.14.18 根据用户标识获取单个用户

接口功能

根据用户Id查询用户信息

接口原型

```
User getUser(String userId);
```

参数说明

参数	必选/可选	类型	描述
userId	必选	String	用户的唯一标识

返回结果

类型	描述
User use	用户实体类

注意事项

无

调用示例

```
无
```

7.2.14.19 获取所有快捷方式列表

接口功能

获取快捷方式列表

接口原型

```
List<ShortcutEntity> getAllShortcuts();
```

参数说明

参数	必选/可选	类型	描述
void			

返回结果

类型	描述
List<ShortcutEntity>快捷键实体列表	Shortcut实体 shortcutId、shortcutType

注意事项

无

调用示例

```
无
```

7.2.14.20 设置快捷方式列表

接口功能

设置快捷方式列表（包括增删排序）

接口原型

```
void setShortcuts(List<ShortcutEntity> shortcuts);
```

参数说明

参数	必选/可选	类型	描述
shortcuts	必选	List<ShortcutEntity>	快捷方式实体列表

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
无
```

7.2.14.21 根据摄像头标识获取摄像头的缩略图信息

接口功能

根据摄像头id获取摄像头的缩略图信息（Android独有）

接口原型

```
CameraSnapshotInfo getCameraSnapshotInfo(String cameraId);
```

参数说明

参数	必选/可选	类型	描述
cameraId	必选	String	摄像头的设备Id

返回结果

类型	描述
CameraSnapshotInfo	缩略图实体类信息

注意事项

无

调用示例

```
无
```

7.2.14.22 保存摄像头的缩略图信息

接口功能

保存摄像头的缩略图信息（Android独有）

接口原型

```
void setCameraSnapshotInfo(CameraSnapshotInfo info);
```

参数说明

参数	必选/可选	类型	描述
info	必选	CameraSnapshotInfo	摄像头的缩略图信息

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
无
```

7.2.14.23 根据摄像头标识删除摄像头的缩略图信息

接口功能

根据摄像头id删除摄像头的缩略图信息（Android独有）

接口原型

```
void deleteCameraSnapshotInfo(String cameraId);
```

参数说明

参数	必选/可选	类型	描述
cameraId	必选	String	摄像头的设备Id

返回结果

类型	描述
Void	无返回值

注意事项

无

调用示例

```
无
```

7.3 常见数据结构定义

该章节主要将SDK提供功能所涉及的数据模型在此呈现出来，方便开发者了解数据模型中具体的属性所代表的意思，当开发者在开发接口或功能时遇到一些数据模型的操作均可以在此找到相对应的结构供参考。

7.3.1 Device

字段	必选/可选	类型	描述
createTime	必选	String	设备添加时间, 时间格式 yyyyMMdd'T'HHmmss'Z' , 例如, 20151212T121212Z
deviceId	必选	String	设备标识
gatewayId	必选	String	网关标识
nodeType	必选	String	节点
deviceInfo	必选	DeviceInfo	设备详情
services	必选	List<Service>	设备的服务信息

7.3.2 DeviceInfo

字段	必选/可选	类型	描述
name	可选	String	设备名称
deviceDescription	可选	String	设备描述
bridgeId	可选	String	桥接标识
manufacturerId	必选	String	供应商标识

字段	必选/可选	类型	描述
manufacturerName	可选	String	供应商名字
mac	可选	String	物理地址
location	可选	String	设备位置
deviceType	必选	String	设备种类，例如： MultiSensor ContactSensor Camera Gateway
model	必选	String	设备型号. 如果是Z-Wave协议，型号就是ProductType + ProductId（用零填充，如果需要） 格式的十六进制数,例如 001A-0A12. 如果是其他协议，型号需要另一种格式表示
swVersion	可选	String	使用Z-Wave协议时的软件版本号。 其格式是 major version number.minor version number, 例如, 1.1.
fwVersion	可选	String	固件版本号
hwVersion	可选	String	硬件版本号
sigVersion	可选	String	信号版本号
protocolType	必选	String	协议类型 (Z-Wave).
signalStrength	可选	String	信号强度
nodeId	可选	String	节点标识
status	可选	String	设备状态. ONLINE: 在线 OFFLINE: 下线 INBOX: 停用.
statusDetail	可选	String	设备状态的详细情况

字段	必选/可选	类型	描述
mute	可选	String	设备是否进行屏蔽 TRUE: 被屏蔽. FALSE: 没被屏蔽.
isSecurity	可选	boolean	是否安全模式
supportedSecurity	可选	boolean	是否支持安全模式

7.3.3 Service

字段	必选/可选	类型	描述
serviceType	必选	String	服务类型
serviceId	必选	String	服务ID标识
data	必选	String	携带服务数据的JSON字符串
eventTime	必选	String	事件发生时间，时间格式yyyyMMdd'T'HHmmss'Z' , 例如, 20151212T121212Z.
serviceInfo	可选	ServiceInfo	服务详情

7.3.4 ServiceInfo

字段	必选/可选	类型	描述
muteCmds	可选	List<String>	隐私模式屏蔽命令列表
status	可选	String	
timestamp	可选	String	时间格式yyyyMMdd'T'HHmmss'Z' , 例如, 20151212T121212Z.

7.3.5 ServiceCommand

字段	必选/可选	类型	描述
commandName	必选	String	设备服务命令名称
commandParameters	必选	Map<String, Object>	设备服务命令参数，需要根据平台的servcie_profile来设置每个命令的具体参数的键值对
serviceId	必选	String	设备服务标识

7.3.6 CloudStorageSpace

字段	必选/可选	类型	描述
quataByte	必选	Int	总共大小
spaceSizeByte	必选	Int	使用大小

7.3.7 VideoRecordingInfo

字段	必选/可选	类型	描述
result	必选	String	视频录制结果
deviceId	必选	String	设备标识
fiestFrameUrl	必选	String	首帧图片路径
availableSize	必选	int	视频大小
duration	可选	int	时长
frameSize	必选	int	首帧图片大小
videoUrl	必选	String	视频路径
totalSize	必选	int	总的大小

7.3.8 Event（所有的事件子类继承自该父类）

字段	必选/可选	类型	描述
eventId;	必选	String	事件id，唯一标示。

字段	必选/可选	类型	描述
homeId;	必选	int	HomeId
triggerId;	必选	String	触发者：username/sceneId/deviceId
triggerType;	必选	String	触发类型(device、rule、manual recording、rule alteration 、storage、dmupgrade)
startTime;	必选	String	事件的触发时间,时间格式 yyyymmddThhmmssZ , 例如, 20151212T121212Z .

7.3.9 RegularEvent(继承自 Event)

字段	必选/可选	类型	描述
name;	必选	String	场景名称
type;	必选	long	场景类型。取值范围：2为定时场景，3为条件场景
logic	必选	String	多条件关系
actions;	必选	List<JSONObject>	场景动作,动作类型type取值： 1、“DEVICE_CMD”对应RegularActionEvent
conditions	必选	List<JSONObject>	场景条件，条件类型type取值： 1、“DEVICE_DATA”对应RegularConditionEvent

7.3.10 RuleEventAction

字段	必选/可选	类型	描述
eventType	必选	String	执行动作的事件类型：device upgrade、device deleted、device disconnection、low power、recording、other
result	必选	String	动作执行结果
deviceType	必选	String	设备类型
deviceName	必选	String	设备名称
actionId	必选	String	动作Id。用于录像动作
action	必选	Action	设备动作
startTime	必选	String	事件的触发时间
data	必选	JSONObject	设备执行动作后的上报结果。根据eventType而不同

7.3.11 RuleEventCondition

字段	必选/可选	类型	描述
deviceId	必选	String	设备Id
deviceType	必选	String	设备类型
deviceName	必选	String	设备名称
serviceId	必选	String	服务Id
property	必选	String	属性名称
rightValue	必选	String	属性值
operator	必选	String	[“>”,”<”,”=”]
satisfactionTime	必选	String	条件满足时间
duration	必选	int	持续时间

7.3.12 Action

字段	必选/可选	类型	描述
messageType;	必选	String	控制命令
messageBody	必选	String	命令参数

7.3.13 EventNotificationSms

字段	必选/可选	类型	描述
phone	可选	String	发短信信息的手机号码
result	可选	String	发送信息的内容

7.3.14 EventNotificationEmail

字段	必选/可选	类型	描述
email	可选	String	发送email的email地址
result	可选	String	发送email的内容

7.3.15 DeviceEvent（继承自 Event）

字段	必选/可选	类型	描述
deviceId	必选	String	发生状态变化的设备ID
deviceType	必选	String	发生状态变化的设备类型
deviceName	必选	String	发生状态变化的设备名

字段	必选/可选	类型	描述
eventType	必选	String	事件类型（具体有apiEventTypeDeviceUpgrade（设备升级） apiEventTypeDeviceDelete（删除设备） apiEventTypeDeviceConnect（设备上线） apiEventTypeDeviceDisconnect（网关断连） apiEventTypeSensorDisconnect（设备断连） apiEventTypeLowpower（低电量）
data	可选	JSONObject	低电量时存储的当前电量，其他事件(apiEventTypeOther)存储的当前设备的发生变化的属性值，如门磁开关事件 {@”status”:@”OPEN”}
deviceStartTime	必选	String	事件的触发时间

7.3.16 StorageEvent（继承自 Event）

字段	必选/可选	类型	描述
eventType	必选	String	事件 类型。取值范围：UDS space not enough
usedPercent	必选	String	存储空间使用百分比
storageStatus	必选	Int	当前存储状态（1、正常；3、满了）

7.3.17 ManualRecordingEvent（继承自 Event）

字段	必选/可选	类型	描述
deviceId	必选	String	录像的设备ID
deviceType	必选	String	录像的设备类型
deviceName	必选	String	录像的设备名
recordingStartTime	必选	String	录像发生的时间
data	必选	RecordReslut	录像的结果

7.3.18 RecordReslut

字段	必选/可选	类型	描述
result	必选	String	录像结果[SUCCESS_VIDEO,ERROR_VIDEO_RECORDING,ERROR_VIDEO_IPC_ERROR]
duration	必选	String	录像时间
videoUrl	必选	String	录像视频的url
fileSize	必选	String	录像视频的文件大小
firstFrameUrl	必选	String	录像视频的首帧图

7.3.19 DeviceUpgradeEvent（继承自 Event）

字段	必选/可选	类型	描述
deviceId	必选	String	设备Id
deviceType	必选	String	设备类型
deviceName	必选	String	设备名称
upgradeTime	必选	String	设备升级的触发时间
upgradeResult	必选	String	设备升级的操作结果

7.3.20 Rule

字段	必选/可选	类型	描述
id	必选	int	规则id
name	必选	String	场景名字
description	可选	String	场景描述
imageIndex	必选	int	场景图片
type	必选	int	场景类型1：手动场景，2：定时场景，3：条件场景，4：预定义场景
status	可选	String	场景状态，取值为：“active”和“inactive”
actions	可选	List<JSONObject>	场景动作,动作类型type取值： 1、“DEVICE_CMD”对应RuleAction
conditions	可选	List<JSONObject>	场景条件，条件类型type取值： 1、“DEVICE_DATA”对应RuleDeviceDataCondition 2、“DAILY_TIMER”对应RuleDailyTimerCondition 3、“CYCLE_TIMER”对应RuleCycleTimerCondition
timeRange	可选	RuleTimeRange	定时场景的时间段
notifies	可选	List<RuleNotification>	场景通知对象列表
matchNow	可选	String	是否马上触发：取值 yes | no

字段	必选/可选	类型	描述
timeZoneID	可选	String	用户家庭时区ID

7.3.21 RuleAction

字段	必选/可选	类型	描述
type	必选	String	动作类型
id	可选	int	规则id
deviceId	必选	String	设备id
serviceId	必选	String	服务id
serviceType	可选	String	服务类型
messageType	必选	String	控制命令
messageBody	必选	JSONObject	命令参数

7.3.22 RuleDeviceDataCondition

字段	必选/可选	类型	描述
type	条件场景必选	String	条件类型
deviceId	条件场景必选	String	设备id
serviced	条件场景必选	String	服务id
serviceType	可选	String	服务类型
property	条件场景必选	String	传感器属性字段
operator	条件场景必选	String	操作符，“>”、“<”、“=”
rightValue	条件场景必选	String	传感器属性值
duration	可选	int	条件满足后规则触发的等待时长，单位为min

7.3.23 RuleDailyTimerCondition

字段	必选/可选	类型	描述
type	必选	String	条件类型
time	必选	String	固定触发时间，格式为：“xx：xx”，二十四小时制，以UTC+0为基准
daysOfWeek	必选	List<Integer>	星期列表，1代表星期日，2代表星期一，以此类推

7.3.24 RuleCycleTimerCondition

字段	必选/可选	类型	描述
type	必选	String	条件类型
timeRange	必选	RuleTimeRange	条件时间
regularityLen	必选	String	周期

7.3.25 RuleTimeRange

字段	必选/可选	类型	描述
type	必选	String	条件类型
startTime	必选	String	开始时间点，格式为：“xx：xx”，二十四小时制，以UTC+0为基准
endTime	必选	String	结束时间点，格式同上
daysOfWeek		List<Integer>	星期列表，1代表星期日，2代表星期一，以此类推

7.3.26 RuleNotification

字段	必选/可选	类型	描述
accounted	可选	int	用户id，需要push通知时必填
phone	可选	String	Sms通知的手机号码
email	可选	String	邮件通知的邮箱地址
push	可选	String	Y代表需要push通知，N代表不必要
title	可选	String	预留用户昵称

7.3.27 ServicePackageInfo

字段	必选/可选	类型	描述
Type	可选	String	服务类型，如：sms,cloudStorage
Status	可选	String	服务状态，如：activated/ deactivated
quota	可选	int	总共可使用的存储空间
usage	可选	int	已经使用的存储空间

7.3.28 CameraSnapshotInfo

字段	必选/可选	类型	描述
updateTime	可选	long	上一次抓取获取缩略图的时间
path	可选	String	缩略图路径
cameraId	可选	String	摄像头标识

7.4 常量定义

7.4.1 配置项常量

常量名	描述
String SYS_CFG_NA_SERVER_IP	配置项：服务器IP
String SYS_CFG_NA_SERVER_PORT	配置项：服务器端口
String SYS_CFG_MQTT_SERVER_IP	配置项：推送服务器IP
String SYS_CFG_MQTT_SERVER_PORT	配置项：推送服务器端口
String SYS_CFG_OPERATION_TIMEOUT_TIME	配置项：操作超时时间

7.4.2 通用错误码

常量名	描述
int ERROR_CODE_SYS_NULL	通用错误码：无错误
int ERROR_CODE_SYS_NETWORK_UNAVAILABLE	通用错误码：网络不可用
int ERROR_CODE_SYS_DEVICE_OFFLINE	通用错误码：设备离线
int ERROR_CODE_SYS_GATEWAY_OFFLINE	通用错误码：网关离线
int ERROR_CODE_SYS_USER_OFFLINE	通用错误码：用户离线
int ERROR_CODE_SYS_POOR_NETWORK	通用错误码：网络不好
int ERROR_CODE_SYS_INVALID_PARAM	通用错误码：无效参数

7.4.3 登录注销常量

1.参数定义

常量名	描述
String LOGIN_EXTENDING_PARAM_VERIFY_CODE	登录扩展参数：登录验证码

2.错误码定义

常量名	描述
int LOGIN_STATUS_CONNECTING	登录状态：连接中
int LOGIN_STATUS_CONNECTED	登录状态：已经连接
int LOGIN_STATUS_DISCONNECTED	登录状态：断开连接
int ERROR_CODE_LOGIN_AUTH_FAILED	登录错误码：鉴权失败
int ERROR_CODE_LOGIN_ACCOUNT_INACTIVE	登录错误码：账号未激活
int ERROR_CODE_LOGIN_ACCOUNT_LOCKED	登录错误码：账号因多次鉴权失败被锁定
int ERROR_CODE_LOGIN_INVALID_VERIFY_CODE	登录错误码：验证码错误
int ERROR_CODE_LOGIN_FORCE_LOGOUT	登录错误码：账号被踢
int ERROR_CODE_LOGIN_TRUST_DEVICE_UNREGISTER	登录错误码：在未信任的设备上登录
int ERROR_CODE_LOGIN_TRUST_DEVICE_VERIFY_CODE_ERROR	登录错误码：设备验证码错误
int ERROR_CODE_LOGIN_TRUST_DEVICE_VERIFY_CODE_RETRY_LIMIT	登录错误码：设备验证码错误且重试次数已用完
int ERROR_CODE_LOGIN_TRUST_DEVICE_VERIFY_CODE_EXPIRED	登录错误码：设备验证码过期
ERROR_CODE_ACTIVATE_ACCOUNT_ACTIVATION_CODE_EXPIRED	激活码过期
ERROR_CODE_ACTIVATE_ACCOUNT_ATTEMPTS_TIMES_REACH_MAX	使用激活码激活失败超过最大次数限制
ERROR_CODE_ACTIVATE_ACCOUNT_ACTIVATION_CODE_EXISTED	该激活码已被使用过
ERROR_CODE_REQUEST_ACTIVATION_CODE_FAILED	获取激活码失败

常量名	描述
ERROR_CODE_REQUEST_ACTIVATION_CODE_ATTEMPTS_TIMES_REACH_MAX	获取激活码超过最大次数限制

7.4.4 用户管理常量

1.参数定义

常量名	描述
String USER_EXTENDING_PARAM_EMAIL	注册和设置个人信息扩展参数：邮箱
String USER_EXTENDING_PARAM_USER_NAME	设置个信息扩展参数：用户名

2.错误码定义

常量名	描述
int ERROR_CODE_USER_EMAIL_OCCUPIED	注册错误码：邮箱被占用
int ERROR_CODE_USER_ACCOUNT_NAME_OCCUPIED	注册错误码：用户名被占用
int ERROR_CODE_USER_ACCOUNT_NAME_AND_EMAIL_OCCUPIED	注册错误码：用户名和邮箱被占用
int ERROR_CODE_USER_PASSWORD_SAME_AS_ACCOUNT_NAME	注册错误码：密码与账号一样
int ERROR_CODE_USER_PASSWORD_SAME_AS_ACCOUNT_NAME_REVERSE	注册错误码：密码与账号倒序一样
int ERROR_CODE_USER_PASSWORD_SAME_AS_USER_NAME	注册错误码：密码与用户名一样
int ERROR_CODE_USER_PASSWORD_SAME_AS_USER_NAME_REVERSE	注册错误码：密码与用户名倒序一样

常量名	描述
int ERROR_CODE_USER_PASSWORD_SAME_AS_EMAIL	注册错误码：密码与邮箱一样
int ERROR_CODE_USER_PASSWORD_SAME_AS_EMAIL_REVERSE	注册错误码：密码与邮箱倒序一样
int ERROR_CODE_USER_PASSWORD_TOO_SHORT	注册错误码：密码太短。要：8--64位
int ERROR_CODE_USER_PASSWORD_TOO_LONG	注册错误码：密码太长。要：8--64位
int ERROR_CODE_USER_PASSWORD_FORMAT_INVALID	注册错误码：密码格式不对。要求：必须包括大、小写字母和数字
int ERROR_CODE_USER_RESET_PASSWORD_INVALID_VERIFY_CODE	重设密码错误码：重设密码验证码不对
int ERROR_CODE_USER_RESET_PASSWORD_ERROR_VERIFY_CODE_EXPIRED	重设密码错误码：重设密码验证码过期
int ERROR_CODE_USER_RESET_PASSWORD_PASSWORD_MATCHES_OLD	修改密码、重设密码错误码：重设密码新老密码一样
ERROR_CODE_INVITE_USER_EXISTED_IN_OTHER_HOME	被邀请用户已经在其它家庭中
ERROR_CODE_INVITE_USER_CAN_NOT_BE_INVITED	该用户不能被邀请
ERROR_CODE_INVITE_USER_NUMBER_REACH_MAX	被邀请的用户数达到上限
ERROR_CODE_INVITE_USER_CAN_NOT_BE_SELF	不能邀请自己

7.4.5 设备管理常量

1.参数定义

常量名	描述
DEVICE_STATUS_ONLINE	设备在线状态：在线

常量名	描述
DEVICE_STATUS_OFFLINE	设备在线状态：离线
DEVICE_STATUS_DETAIL_NONE	设备升级状态：没有在升级
DEVICE_STATUS_DETAIL_UPDATING	设备升级状态：正在升级
DEVICE_STATUS_DETAIL_NOT_ACTIVE	设备状态：设备未被激活
EVENT_VIDEO_RECORDING_DONE	视频录制完成广播
PARAM_VIDEO_RECORDING_INFO	视频录制完成广播带的参数：VideoRecordingInfo
VIDEO_RECORD_SUCCESS	视频录制成功
VIDEO_RECORD_FAILED	视频录制失败

2.错误码定义

常量名	描述
ERROR_CODE_DEVICE_USER_NOT_FOUND	错误码：查询不到用户
ERROR_CODE_DEVICE_DEVICE_NOT_FOUND	错误码：查询不到设备
ERROR_CODE_DEVICE_NO_DEVICE_ACCESS	错误码：用户对设备无权限
ERROR_CODE_DEVICE_DEVICE_ALREADY_ACTIVED	绑定网关错误码：设备已经被激活
ERROR_CODE_DEVICE_RENAME_NO_ACCESS	设备重命名错误码：用户对设备无权限
ERROR_CODE_DEVICE_RENAME_DEVICE_INACTIVATED	设备重命名错误码：设备未被激活
ERROR_CODE_DEVICE_RENAME_NAME_ALREADY_EXIST	设备重命名错误码：设备名已存在

7.4.6 视频常量

1.参数定义

常量名	描述
EVENT_VIDEO_RECORDING_DONE	视频录制完成广播

常量名	描述
PARAM_VIDEO_RECORDING_INFO	视频录制完成广播带的参数：VideoRecordingInfo
VIDEO_RECORD_SUCCESS	视频录制成功
VIDEO_RECORD_FAILED	视频录制失败
MEDIA_PLAYER_TYPE_LIVE	获取播放实时视频MediaPlayer参数
MEDIA_PLAYER_TYPE_VOD	获取播放历史视频MediaPlayer参数

2.错误码定义

常量名	描述
int ERROR_CODE_DEVICE_VIDEO_RECORD_ALREADY_DOWNLOADED	视频下载错误码：视频已经下载
int ERROR_CODE_DEVICE_VIDEO_RECORD_IS_DOWNLOADING	视频下载错误码：视频正在下载
Int ERROR_CODE_DEVICE_VIDEO_RECORD_DOWNLOAD_ERROR_FROM_SERVER	视频下载错误码：视频下载失败
int ERROR_CODE_DEVICE_VIDEO_RECORD_NOT_EXIST	视频下载错误码：下载的视频不存在

7.4.7 离线推送常量

1.参数定义

常量名	描述
EVENT_RECEIVE_OFFLINE_PUSH	离线消息广播
PARAM_OFFLINE_PUSH_MESSAGE	离线消息Extra: 消息内容
PARAM_OFFLINE_PUSH_EVENT_ID	离线消息Extra: 事件Id
EVENT_ENABLE_PUSH_RESULT	激活离线推送广播
PARAM_ENABLE_PUSH_RESULT_FAILED_REASON	离线推送extra：离线推送激活失败错误码
PARAM_ENABLE_PUSH_RESULT	离线推送extra：离线推送激活结果

常量名	描述
ENABLE_PUSH_RESULT_SUCCESS	离线推送激活成功
ENABLE_PUSH_RESULT_FAILED	离线推送激活失败

2.错误码定义

常量名	描述
Int ERROR_CODE_PUSH_FAIL_TO_REGISTER_FOR_REMOTE_NOTIFICATIONS	激活离线推送错误码：用senderId向谷歌获取registerId失败
int ERROR_CODE_PUSH_FAIL_TO_SEND_TOKEN_TO_SERVER	激活离线推送错误码：发送鉴权到第三方服务器失败
int ERROR_CODE_PUSH_LOGIN_STATUS_NOT_CONNECTED	激活离线推送错误码：用户未登陆成功

7.4.8 房间常量管理

错误码定义

常量名	描述
Int ERROR_CODE_ROOM_ALREADY_EXISTED	房间已存在
int ERROR_CODE_ROOM_NOT_FOUND	找不到房间
int ERROR_CODE_ROOM_DEFAULT_FORBIDEN	操作默认房间或使用默认房间名被拒绝

8 APP SDK API 参考(iOS)

8.1 开发者必读

SDK 开放组件介绍

SDK提供如下组件：

注意

所有接口参数的有效性检查由开发者自己保证。

API组件	说明
SysApi	系统组件，提供整个系统初始化、设置和获取配置信息，相关接口
LogApi	日志组件，提供日志打印和日志上传相关接口
UpgradeApi	升级组件，提供App新版本检测与升级安装相关接口
LoginApi	登录组件，提供登录、注销、激活账号等操作接口
UserApi	用户组件，提供自助开户、忘记密码、修改密码、多用户操作相关接口

DeviceApi	设备组件，提供设备相关操作接口，如设备接入、数据上报、控制命令下发
EventApi	事件组件，提供新事件上报、事件查询、删除等相关操作接口
PushApi	离线推送组件，提供离线推送相关接口（仅支持GCM）
RoomApi	房间组件，提供房间创建、更新、删除和添加设备到房间相关操作接口
RuleApi	规则组件，提供规则创建、修改、删除、激活\去激活等相关操作接口
SettingsApi	设置组件，提供事件push通知设置查询、激活\去激活，云存储自动清理开关配置查询与修改等相关操作接口
LocalStorageApi	本地存储组件，提供各组件数据模型model的存取操作接口，仅开放给VIP用户使用

修改记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

文档版本	发布日期	修改说明	修改影响
1	2016/06/01	第一次发布	
2	2017/11/08	文档更新	

8.2 SDK 功能介绍

该章节提供的SDK功能包括，系统的初始化配置，日志打印收集功能，登陆及注销功能，用户及用户信息管理，设备相关的操作及信息获取，摄像头的操作，实时视频及历史视频的播放，设备及操作产生的事件呈现，离线消息推送，规则增删改查。该章节旨在详细介绍SDK提供的所有功能和接口，开发者可按照该章节进行相关接口和功能的开发调试。

8.2.1 系统初始化

系统初始化依赖SysApi组件，提供整个系统初始化、设置和获取配置信息，相关功能。开发者需要先初始化整个系统，然后才能调用其它API。

8.2.1.1 初始化

接口功能

初始化整个系统 。

接口原型

```
+ (void)init
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

在调用其它API前需要确保该接口已经被调用。建议在程序启动的时候调用该接口。

调用示例

```
- (BOOL)application:(UIApplication *)application didFinishLanchWithOptions(NSDictionnary
*)lanchOptions {
[SysApi init];
……
return YES;
}
```

8.2.1.2 设置配置信息

接口功能

设置配置信息，如服务器IP、端口等 。

接口原型

```
+ (void)setConfig:(NSString *)key value:(NSString *)value
```

参数说明

参数	必选/可选	类型	描述
key	必选	NSString	配置项。（定义详见4.1） 当前支持如下配置项。开发也可以自己定义配置项，设置到SDK存储。 SYS_CFG_NA_SERVER_IP 配置服务器IP SYS_CFG_NA_SERVER_PORT配置服务器端口 SYS_CFG_MQTT_SERVER_IP配置推送服务器IP SYS_CFG_MQTT_SERVER_PORT配置推送服务器端口 SYS_CFG_NA_SERVER_STANDBY_IP 配置服务器备用IP SYS_CFG_COUNTRY_CODE 配置国家码 SYS_CFG_OPERATION_TIMEOUT_TIME 配置操作超时时间, 单位秒(s)
value	必选	NSString	配置项的值。

返回结果

类型	描述
void	无返回值

注意事项

在调用其它业务API前，需要调用该接口设置服务器的地址和端口。

调用示例

```
[SysApi setConfig:SYS_CFG_NA_SERVER_IPvalue:@"127.0.0.1"];
[SysApi setConfig:SYS_CFG_NA_SERVER_PORT value:@(8080)];
```

8.2.1.3 获取配置信息

接口功能

获取配置项的值。

接口原型

```
+ (void) getConfig:(NSString *)key
```

参数说明

参数	必选/可选	类型	描述
key	必选	NSString	配置项。 当前支持如下配置项。开发也可以自己定义设置到SDK存储的配置项。 SYS_CFG_NA_SERVER_IP 配置服务器IP SYS_CFG_NA_SERVER_PORT配置服务器端口 SYS_CFG_MQTT_SERVER_IP配置推送服务器IP SYS_CFG_MQTT_SERVER_PORT配置推送服务器端口 SYS_CFG_NA_SERVER_STANDBY_IP 配置服务器备用IP SYS_CFG_COUNTRY_CODE 配置国家码 SYS_CFG_OPERATION_TIMEOUT_TIME 配置操作超时时间, 单位秒(s)

返回结果

类型	描述
NSString	配置项的值

注意事项

若该配置项没有，会返回nil

调用示例

```
NSString *ip = [SysApi getConfig:SYS_CFG_NA_SERVER_IP];
```

8.2.1.4 应用程序前台处理

接口功能

通知SDK应用程序进入前台。

接口原型

```
+ (void) notifyAppForeground:(UIApplicatio *)application
```

参数说明

参数	必选/可选	类型	描述
application	必选	UIApplication	应用程序当前的application。

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
-(void)applicationDidBecomeActive:(UIApplication *)application {
[SysApi notifyAppForeground:application];
}
```

8.2.1.5 应用程序后台处理

接口功能

通知SDK应用程序进入后台。

接口原型

```
+ (void) notifyAppBackground:(UIApplication *)application
```

参数说明

参数	必选/可选	类型	描述
application	必选	UIApplication	应用程序当前的application。

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
-(void)applicationDidEnterBackground:(UIApplication *)application {
[SysApi notifyAppBackground:application];
}
```

8.2.1.6 销毁

接口功能

释放SysApi的资源。

接口原型

```
+ (void) destroy
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

退出APP前需要调用该接口回收资源

调用示例

```
[SysApi destroy];
```

8.2.2 日志打印

日志打印依赖LogApi组件，提供日志打印和日志上传相关功能。在调用LogApi提供的相关接口前，必须确保已经调用了SysApi的init接口初始化了整个系统。

8.2.2.1 初始化

接口功能

初始化日志API。

接口原型

```
+ (void)init
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[LogApi init];
```

8.2.2.2 销毁

接口功能

日志API销毁，释放内存。

接口原型

```
+ (void)destroy
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

应用程序退出前，必须销毁日志API，回收资源。

调用示例

```
[LogApi destroy];
```

8.2.2.3 打印信息级别日志

接口功能

打印Info级别日志。

接口原型

```
+ (void) i:(NSString *)tag msg:(NSString *)msg
```

参数说明

参数	必选/可选	类型	描述
tag	必选	NSString	日志标识，通常是调用者的类名
msg	必选	NSString	日志内容

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[LogApi i:TAG msg:@“this is a information.”];
```

8.2.2.4 打印警告级别日志

接口功能

打印警告级别日志。

接口原型

```
+ (void) w:(NSString *)tag msg:(NSString *)msg
```

参数说明

参数	必选/可选	类型	描述
tag	必选	NSString	日志标识，通常是调用者的类名
msg	必选	NSString	日志内容

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[LogApi w:TAG msg:@“this is a warning.”];
```

8.2.2.5 打印调试级别日志

接口功能

打印调试级别日志。

接口原型

```
+ (void) d:(NSString *)tag msg:(NSString *)msg
```

参数说明

参数	必选/可选	类型	描述
tag	必选	String	日志标识，通常是调用者的类名
msg	必选	String	日志内容

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[LogApi d:TAG, msg:@“this is a debug.”];
```

8.2.2.6 打印错误级别日志

接口功能

打印错误级别日志。

接口原型

```
+ (void) e:(NSString *)tag msg:(NSString *)msg
```

参数说明

参数	必选/可选	类型	描述
tag	必选	NSString	日志标识，通常是调用者的类名
msg	必选	NSString	日志内容

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[LogApi e:TAG msg:@“this is an error.”];
```

8.2.2.7 日志上传

接口功能

上传日志到云端，用于问题分析。

接口原型

```
+ (void) uploadLog:(void (^)(double progress))progress success:(void (^)(void))success failure:(void (^)(void))failure
```

参数说明

参数	必选/可选	类型	描述
progress	可选	void (^)(double progress)	日志上传进度block回调。
success	可选	void (^)(void)	日志上传成功block回调。
failure	可选	void (^)(void)	日志上传失败block回调。

返回结果

类型	描述
void	无返回值

注意事项

调用该接口的前提条件是，已经登录成功.

调用示例

```
[LogApi uploadLog:^(double progress) {

} success:^{

} failure:^{

}];
```

8.2.2.8 自动收集日志广播通知

接口功能

收到自动收集日志消息时，上报的广播通知。

接口原型

```
EVENT_RECEIVE_AUTO_COLLECTION_LOG_MESSAGE
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(receiveAutoCollectionLogMessage:) name:
EVENT_RECEIVE_AUTO_COLLECTION_LOG_MESSAGE
object:nil];

......

- (void)receiveAutoCollectionLogMessage:(NSNotification *)noti {
    // do something
}
```

8.2.3 App 升级

App新版本的检测与升级依赖UpgradeApi组件，提供App新版本检测、新版本安装功能。

8.2.3.1 检测新版本

接口功能

检测App是否有新版本。

接口原型

```
+ (void) checkNewVersion:(void (^)(NewVersionInfo *info))success failure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void (^)(NewVersionInfo *info)	请求执行成功block回调。返回一个新版本信息对象
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
NewVersionInfo	返回一个新版本信息对象

注意事项

登录成功后才能调用此接口。

调用示例

```
#import "UpgradeApi.h"
[UpgradeApi checkNewVersion:^(NewVersionInfo *info) {
    // do something
} failure:^(long errorCode) {
    // do something
} timeout ^{
    // do something
}];
```

8.2.3.2 安装新版本

接口功能

安装App新版本。

接口原型

```
+ (void) installNewVersion:(NSArray<NSString *> *)urls
```

参数说明

参数	必选/可选	类型	描述
urls	必选	NSArray<NSString *>	新版本安装urls路径数组

返回结果

类型	描述
void	无返回值

注意事项

检测到新版本后才能调用此接口。

调用示例

```
#import "UpgradeApi.h"
NSArray *urls = @[@"https://xxxxxxxx"];
[UpgradeApi installNewVersion:];
```

8.2.4 登录注销

登录注销依赖LoginApi组件，提供登录、注销、忘记密码、修改密码等功能。要实现登录相关业务，需要确保调用SysApi init接口初始化了系统，调用SysApi setConfig设置了服务器的地址和端口。

8.2.4.1 初始化

接口功能

初始化LoginApi。

接口原型

```
+ (void) init
```

参数说明

无。

返回结果

类型	描述
void	无返回值

注意事项

调用前需要确保已经调用[SysApi init]初始化了SysApi。调用该接口后，登录组件的其它接口才可用。

调用示例

```
[LoginApi init];
```

8.2.4.2 登录

接口功能

用户登录。

接口原型

```
+ (void) login:(NSString *)userId password:(NSString *)password success:(void (^)(void))success
failure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID, 如果使用phone作为userId登录，必须在号码前面增加国家码。例如：0086
password	必选	NSString	密码
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.3.2

参数	必选/可选	类型	描述
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

调用该接口能够登录成功的前提条件是，SysApi已经初始化、LoginApi已经初始化、服务器的地址已经设置。

调用示例

```
- (void)loginButtonClicked:(id)sender {
NSString *userId = self.userLB.text;
NSString *pwd = self.pwdLB.text;
[LoginApi login:userId password:pwd success:^{
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
}
```

8.2.4.3 带扩展参数登录

接口功能

用户登录，可以携带除用户ID和密码外的扩展参数，如验证码、新设备授权验证码。

接口原型

```
+ (void) login:(NSString *)userId password:(NSString *)password params:(NSDictionary *)params
success:(void (^)(void))success failure:(void (^)(long errorCode))failure timeout:(void (^)
(void))timeout
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID，如果使用phone作为userId登录，必须在号码前面增加国家码。例如：0086
password	必选	NSString	密码

参数	必选/可选	类型	描述
params	必选	NSDictionary	用来携带扩展参数，当前支持LOGIN_EXTENDING_PARAM_VERIFY_CODE携带登录验证码, 多次鉴权失败，再次登录需要携带验证码。如何获取验证码，请参考获取登录验证码章节。 LOGIN_EXTENDING_PARAM_TRUST_DEVICE_VERIFY_CODE携带新设备授权验证码
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。 errorCode定义详见4.3.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

调用该接口能够登录成功的前提条件是，SysApi已经初始化、LoginApi已经初始化、服务器的地址已经设置。验证码使用的场景是用户账号被锁定，需要调用”获取登录验证码”接口来生成；新设备授权验证码使用场景是在一台新的设备上登录，该验证码会发送到您的注册邮箱。

调用示例

```
- (void)loginButtonClicked:(id)sender {
NSString *userId = self.userLB.text;
NSString *pwd = self.pwdLB.text;
NSString verifyCode = self.verifyCodeLB.text;
NSDictionary *params = [NSDictionary dictionaryWithObjectsAndKeys:verifyCode,
LOGIN_EXTENDING_PARAM_VERIFY_CODE, nil];
[LoginApi login:userId password:pwd params:params success:^{
// do something
```

```
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
}
```

8.2.4.4 获取登录验证码

接口功能

多次登录鉴权失败，用户需要输入验证码登录。该接口用于从云端获取验证码。

接口原型

```
+ (void) getVerifyCodeForLogin:(void (^)(UIImage *)success failure:(void (^)(long
errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void (^)(UIImage *)	请求执行成功block回调。携带验证码图片。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
UIImage	一个image类型验证码图片

注意事项

调用该接口的前提条件是，SysApi已经初始化、LoginApi已经初始化、服务器的地址已经设置。

调用示例

```
[UserApi getVerifyCodeForLogin:^(UIImage *image){
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
```

8.2.4.5 登出

接口功能

用户注销，断开和服务器的连接。

接口原型

```
+ (void) logout:(void (^)(void))success failure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

调用该接口的前提条件是，SysApi已经初始化、LoginApi已经初始化、服务器的地址已经设置，并且已经成功登录过。

调用示例

```
- (void)logoutButtonClicked:(id)sender {
[LoginApi logout:^{
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
}
```

8.2.4.6 激活账号

接口功能

用户开户成功后，激活开户时填写的账号。

接口原型

```
+ (void)activateAccount:(NSString *)userIdactivationCode:(NSString *)activationCode success:(void (^)(void))success failure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID，如果使用phone作为userId登录，必须在号码前面增加国家码。例如：0086
activationCode	必选	NSString	激活码，用户开户成功后，服务器会向用户开户时填写的邮箱发送一封激活邮件，邮件内容里面包含激活码。
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.3.3
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

调用该接口的前提条件是，SysApi已经初始化、服务器的地址和端口已经设置。

调用示例

```
- (void)activationButtonDidClicked:(id)sender {
    NSString *userId = @”008618712345678”;
    NSString *activationCode = @”af36i”;
    [LoginApi activateAccount:userId activationCode:activationCode success:^{
        // do something
    } failure:^(long errorCode) {
        // do something
    } timeout:^{
        // do something
    }];
}
```

8.2.4.7 获取激活码

接口功能

激活码超时或者失效后，重新获取激活码邮件。

接口原型

```
+ (void) requestActivationCode:(NSString *)userId success:(void(^)(void))success failure:(void (^)(long errorCode))failure timeout:(void(^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID，如果使用phone作为userId登录，必须在号码前面增加国家码。例如：0086
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.3.4
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

调用该接口的前提条件是，SysApi已经初始化、服务器的地址和端口已经设置。

调用示例

```
- (void)requestActivationCodeButtonDidClicked:(id)sender {
    NSString *userId = @”008618712345678”;
    [LoginApi requestActivationCode:userId success:^{
        // do something
    } failure:^(long errorCode) {
        // do something
    } timeout:^{
        // do something
    }];
}
```

8.2.4.8 登录状态改变广播通知

接口功能

当登录状态改变时，上报最新的登录状态。

接口原型

```
EVENT_LOGIN_STATUS_CHANGED
```

参数说明

参数	必选/可选	类型	描述
PARAM_LOGIN_STATUS	必选	NSString	广播参数，携带当前登录状态： LOGIN_STATUS_CONNECTING连接中 LOGIN_STATUS_CONNECTED 已经连接 LOGIN_STATUS_DISCONNECTED断开连接
PARAM_LOGIN_REASON	可选	NSString	广播参数，若当前的登录状态为LOGIN_STATUS_DISCONNECTED断开连接，会携带断开连接的原因值： ERROR_CODE_LOGIN_AUTH_FAILED 鉴权失败 ERROR_CODE_LOGIN_ACCOUNT_INACTIVE 账号未激活 ERROR_CODE_LOGIN_ACCOUNT_LOCKED 账号因多次鉴权失败被锁定 ERROR_CODE_LOGIN_INVALID_VERIFY_CODE 验证码错误 ERROR_CODE_LOGIN_FORCE_LOGOUT 账号被踢 ERROR_CODE_SYS_NETWORK_UNAVAILABLE网络不可用

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onUserStatusChanged:)
name:EVENT_LOGIN_STATUS_CHANGED object:nil];

......

- (void)onUserStatusChanged:(NSNotification *)notification {
LOGIN_STATUS newStatus = (LOGIN_STATUS)[notification.userInfo[PARAM_LOGIN_STATUS] integerValue];
LOGIN_ERROR_REASON errorReason = (LOGIN_ERROR_REASON)[notification.userInfo[PARAM_LOGIN_REASON]
integerValue];
if (newStatus == LOGIN_STATUS_DISCONNECTED && errorReason == LOGIN_ERROR_REASON_FORCE_LOGOUT) {
// do something
return;
}

// do other
}
```

8.2.4.9 销毁

接口功能

销毁LoginApi。

接口原型

```
+ (void) destroy
```

参数说明

无。

返回结果

类型	描述
void	无返回值

注意事项

应用退出前需要调用该接口释放LoginApi的资源；调用该接口后，再调用LoginApi其它接口前需要调用[LoginApi init]初始化LoginApi。

调用示例

```
[LoginApi destroy];
```

8.2.5 用户管理

用户管理依赖UserApi组件，提供自助开户、忘记密码、修改密码、多用户相关功能。调用用户组件相关接口，需要确保调用[SysAp init]接口初始化了系统，调用[SysApi setConfig]设置了服务器的地址。

8.2.5.1 初始化

接口功能

初始化UserApi。

接口原型

```
+ (void) init
```

参数说明

无。

返回结果

类型	描述
void	无返回值

注意事项

调用前需要确保已经调用[SysApi init]初始化了SysApi。调用该接口后，用户组件的其它接口才可用。

调用示例

```
[UserApi init];
```

8.2.5.2 用户自助开户

接口功能

自助开户，注册账号。

接口原型

```
+ (void) registerUser:(NSString *)userId userName:(NSString *)userName password:(NSString *)password params:(NSDictionary *)params success:(void (^)(void))success failure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID，如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086

参数	必选/可选	类型	描述
userName	必选	NSString	用户名称
password	必选	NSString	密码
params	必选	NSDictionary	扩展参数，当前支持USER_EXTENDING_PARAM_EMAIL携带邮箱信息
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无。

调用示例

```
NSMutableDictionary *params = [NSMutableDictionary dictionary];
[params setValue:@"email@example.com" forKey:USER_EXTENDING_PARAM_EAMIL];
[UserApi registerUser:@"+8612345678910" userName:@"user name" password:@"123456" params:params
success:^{
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
```

8.2.5.3 忘记密码获取验证码

接口功能

忘记密码，找回密码，需要先从平台获取验证码，该验证码会以短信方式发到用户注册的手机号码上。该接口用于获取该验证码。

接口原型

```
+ (void) getVerifyCodeForRecoveryPassword:(NSString *)userId success:(void (^)(void)success
failure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID。如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无。

调用示例

```
[UserApi getVerifyCodeForRecoveryPassword:@"a user ID" success:^{
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
```

8.2.5.4 忘记密码验证验证码

接口功能

忘记密码，找回密码，需要先从平台获取验证码，该验证码会以短信方式发到用户注册的手机号码上，然后用户输入验证码，进行验证。该接口用于验证验证码。

接口原型

```
+ (void) validateRecoveryPasswordVerifyCode:(NSString *)userId verifyCode:(NSString *)verifyCode
success:(void (^)(void)success failure:(void (^)(long errorCode))failure timeout:(void (^)
(void))timeout
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID。如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
verifyCode	必选	NSString	通过getVerifyCodeForRecoveryPassword获取到的短信验证码。
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无。

调用示例

```
[UserApi validateRecoveryPasswordVerifyCode:@"a user ID" verifyCode:@"verify code"  success:^{
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
```

8.2.5.5 忘记密码重置密码

接口功能

忘记密码，找回密码，需要先从平台获取验证码，该验证码会以短信方式发到用户注册的手机号码上，然后用户输入验证码，进行验证，验证成功后，用户可以重置密码。该接口用于重设密码。

接口原型

```
+ (void) resetPassword:(NSString *)userId newPassword:(NSString *)paasword verifyCode:(NSString
*)verifyCode success:(void (^)(void)success failure:(void (^)(long errorCode))failure timeout:
(void (^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID。如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
password	必选	NSString	设置新的密码。
verifyCode	必选	NSString	通过getVerifyCodeForRecoveryPassword获取到的短信验证码。
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无。

调用示例

```
[UserApi resetPassword:@"a user ID" newPassword:@" password"  verifyCode:@" verify code"
success:^{
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
```

8.2.5.6 修改密码

接口功能

修改密码。

接口原型

```
+ (void) modifyPassword:(NSString *)userId oldPassword:(NSString *)oldPassword newPassword:(NSString *)newPassword success:(void (^)(void)success failure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID。如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
oldPassword	必选	NSString	旧密码
newPassword	必选	NSString	新密码
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无。

调用示例

```
[UserApi modifyPassword:@"a user ID" oldPassword:@" old password"  newPassword:@" new password" success:^{
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
```

8.2.5.7 查询当前用户信息

接口功能

查询个人信息。

接口原型

```
+ (void)queryMyInfoSuccess:(void (^)(NSDictionary *params)success failure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void (^)(NSDictionary *params)	请求执行成功block回调。 携带参数为用户信息: 扩展参数，携带要设置的个人信息。当前支持： USER_EXTENDING_PARAM_USER_ID 用户ID USER_EXTENDING_PARAM_EMAIL 邮箱 USER_EXTENDING_PARAM_USER_NAME用户名 USER_EXTENDING_PARAM_ROLE_TYPE 用户角色类型
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
NSDictionary	携带要设置的个人信息。当前支持： USER_EXTENDING_PARAM_USER_ID用户ID USER_EXTENDING_PARAM_EMAIL邮箱 USER_EXTENDING_PARAM_USER_NAME用户名 USER_EXTENDING_PARAM_ROLE_TYPE 用户角色类型

注意事项

无。

调用示例

```
[UserApi queryMyInfoSuccess^(NSDictionary *params){
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
```

8.2.5.8 设置个人信息

接口功能

设置个人信息，当前支持设置邮箱和用户名信息。

接口原型

```
+ (void) setMyInfo:(NSDictionary *) params success:(void (^)(void)success failure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
params	必选	NSDictionary	扩展参数，携带要设置的个人信息。当前支持： USER_EXTENDING_PARAM_EMAIL设置个人信息扩展参数：邮箱 USER_EXTENDING_PARAM_USER_NAME设置个信息扩展参数：用户名

参数	必选/可选	类型	描述
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无。

调用示例

```
NSMutableDictionary *params = [NSMutableDictionary dictionary];
[params setValue:@"email@example.com" forKey:USER_EXTENDING_PARAM_EAMIL];
[params setValue:@"user name" forKey:USER_EXTENDING_PARAM_USER_NAME];
[UserApi setMyInfo:params success:^{
// do something
} failure:^(long errorCode) {
// do something
} timeout:^{
// do something
}];
```

8.2.5.9 查询所有用户

接口功能

查询当前登录用户家庭中的所有用户。

接口原型

```
+ (void)queryAllUsers:(void (^)(NSArray *users)successfailure:(void (^)(long errorCode))failure
timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void (^)(NSArray *)	请求执行成功block回调。返回家庭中的所有用户。

参数	必选/可选	类型	描述
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[UserApi queryAllUsers:^(NSArray *users) {
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.5.10 邀请子用户

接口功能

邀请一个子用户加入到当前登录用户的家庭中。

接口原型

```
+ (void)inviteUser:(NSString *)userId username:(NSString*)username success:(void (^)(void)success
failure:(void (^)(long errorCode))failuretimeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户ID。如果userId为用户手机号码，必须在号码前面增加国家码。例如：0086
userName	必选	NSString	用户名字
success	可选	void (^)(void)	请求执行成功block回调。

参数	必选/可选	类型	描述
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
NSString *userId = @”008618912345678”;
NSString *userName = @”Test”;
[UserApi inviteUser:userId userName:userName success:^{
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.5.11 子用户接受邀请

接口功能

用户收到其它用户发来的邀请加入家庭后，接受这个邀请，加入邀请用户的家庭中。

接口原型

```
+ (void)acceptInvitation:(void (^)(void)successfailure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[UserApi acceptInvitation:^{
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.5.12 子用户拒绝邀请

接口功能

用户接收到其它用户发来的邀请加入家庭后，拒绝这个邀请，不加入邀请用户的家庭中。

接口原型

```
+ (void)rejectInvitation:(void (^)(void)successfailure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void (^)(void)	请求执行成功block回调。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[UserApi rejectInvitation:^{
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.5.13 查询套餐信息

接口功能

查询用户开户时，选择的套餐信息。

接口原型

```
+ (void)queryServicePackageInfo:(void (^)(ServicePackage*servicePackage)success failure:(void (^)
(long errorCode))failure timeout:(void(^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void (^)(ServicePackage *)	请求执行成功block回调。返回一个ServicePackage实体对象。
failure	可选	void (^)(long errorCode)	请求执行失败block回调。errorCode定义详见4.4.2
timeout	可选	void (^)(void)	请求执行超时block回调。

返回结果

类型	描述
ServicePackage	返回一个ServicePackage实体对象。

注意事项

无

调用示例

```
[UserApi queryServicePackageInfo:^ (ServicePackage
*servicePackage) {
    // do something
} failure:^(long errorCode) {
```

```
    // do something
} timeout:^{
    // do something
}];
```

8.2.5.14 套餐信息变更广播通知

接口功能

上报套餐信息变更通知。

接口原型

```
EVENT_SERVICE_PACKAGE_CHANGED
```

参数说明

参数	必选/可选	类型	描述
PARAM_PACKAGE_INFO	必选	NSString	广播参数，携带变更后的套餐信息（ServicePackage对象）

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(servicePackageInfoDidChanged:) name: EVENT_SERVICE_PACKAGE_CHANGED
object:nil];

......

- (void)servicePackageInfoDidChanged:(NSNotification *)noti
{
    ServicePackage *package = noti.userInfo[PARAM_PACKAGE_INFO];
    // do something
}
```

8.2.5.15 用户销户广播通知

接口功能

上报用户被销户通知。

接口原型

```
EVENT_USER_DELETED
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onUserDeleted:)
name:EVENT_USER_DELETED object:nil];

......

- (void)onUserDeleted:(NSNotification *)notification {

// do something
}
```

8.2.5.16 用户去激活广播通知

接口功能

上报用户去激活通知。

接口原型

```
EVENT_USER_INACTIVE
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onUserInactive:)
name:EVENT_USER_INACTIVE object:nil];

......

- (void)onUserInactive:(NSNotification *)notification {
```

```
// do something
}
```

8.2.5.17 销毁

接口功能

释放UserApi的资源。

接口原型

```
+ (void) destroy
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

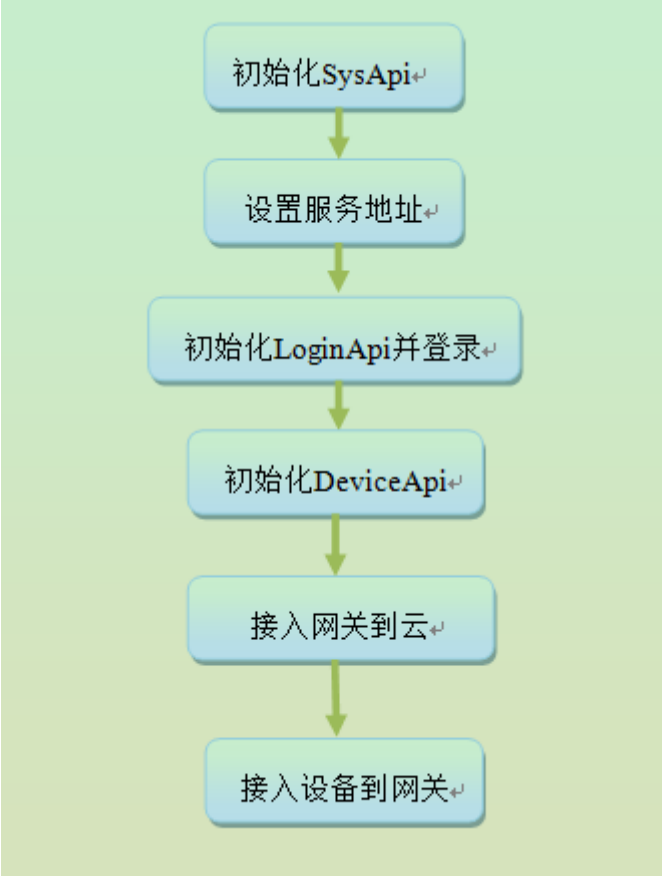
无。

调用示例

```
[UserApi destroy];
```

8.2.6 设备管理

设备管理依赖DeviceApi组件提供了设备接入云、设备接入网关、设备连接状态检查、设备查询、设备信息修改、设备删除、设备控制命令下发、实时视频、视频录像、抓图、视频下载等一系列功能。使用设备管理相关功能前，需要确保调用[SysAp init]接口初始化了系统，调用[SysApi setConfig]设置了服务器的地址，调用登录接口登录成功。接入设备到网关之前，需要把网关（直连设备）接入到云。



8.2.6.1 初始化

接口功能

初始化设备API。

接口原型

```
+ (void)init;
```

参数说明

无。

返回结果

类型	描述
void	无返回值

注意事项

调用设备API接口之前，必须先进行初始化，即调用本接口。

调用示例

```
[DeviceApi init];
```

8.2.6.2 销毁

接口功能

设备API销毁，释放内存。

接口原型

```
+ (void)destroy;
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

应用程序退出前，必须销毁设备API，即调用本接口。

调用示例

```
[DeviceApi destroy];
```

8.2.6.3 注册设备到云

接口功能

应用程序注册直连设备到云上，注册成功后，会返回该设备的信息。

接口原型

```
+ (void)registerDeviceToCloud:(NSString *)pin success:(void(^)(Device *device))success failure:
(void(^)(long errorCode))failure timeout:(void(^)(void))timeout
```

或

```
+ (void)registerDeviceToCloud:(NSString *)pin timeoutTime:(int)timeInterval success:(void (^)
(Device *))success failure:(void (^)(long))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
pin	必选	NSString	物理设备唯一标识符，视具体厂商而定。例如：设备Mac地址、设备UUID等
timeInterval	可选	int	设备注册超时时长，单位秒，默认为90s（不能小于30s）

参数	必选/可选	类型	描述
success	可选	void(^)(Device *device)	接口操作成功回调，携带一个Device对象
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码，errorCode定义参考4.5.3
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
Device	一个Device对象，里面包含设备详细信息

注意事项

注册到云上的设备必须是直连类（Gateway），且此接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”

NSString *mac = @”xxxxxxxxxx”;
[DeviceApi registerDeviceToCloud:mac
timeoutTime:90
success:^(Device *device){
// do something
}                          failure:^(long errorcode){
// do something
}                          timeout:^{
// do something
}];
```

8.2.6.4 注册设备到网关

接口功能

应用程序注册设备到网关，注册成功后，会返回该设备的信息。

接口原型

```
+ (void)startDeviceDiscovery:(NSString *)protocolType success:(void(^)(Device *device))success
failure:(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

或

```
+ (void) startDeviceDiscovery:(NSString *)protocolType timeoutTime:(int)timeInterval success:(void
(^)(Device *))success failure:(void (^)(long))failure timeout:(void (^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
protocolType	必选	NSString	设备接入时使用的协议类型，例如：z-wave、zigbee、wps等
timeInterval	可选	int	设备注册超时时长，单位为秒，默认为60s
success	可选	void(^)(Device *device)	接口操作成功回调，携带一个Device对象
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码 errorCode定义参考4.5.3
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
Device	一个Device对象，里面包含设备详细信息

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”

NSString *protocolType = @”Z-Wave”;
[DeviceApi startDeviceDiscovery:protocolType
timeoutTime:90
success:^(Device *device){
// do something
}                        failure:^(long errorcode){
// do something
}                        timeout:^{
// do something
}];
```

8.2.6.5 查询设备连接状态

接口功能

查询某个设备的连接状态，即设备在线还是离线。

接口原型

```
+ (void)checkDeviceConnection:(NSString *)deviceId success:(void(^)(void))success failure:(void(^)
(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	设备标识
success	可选	void(^)(void)	接口操作成功回调
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码errorCode定义参考4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”

NSString *deviceId = @”xxxxxxxxx”;
[DeviceApi checkDeviceConnection:deviceId
success:^{
// do something
}                          failure:^(long errorcode){
// do something
}                        timeout:^{
// do something
}];
```

8.2.6.6 查询设备列表

接口功能

查询设备列表，返回设备的详细信息。

接口原型

```
+ (void)queryAllDevices:(void(^)(NSArray<Device *> *devices))success failure:(void (^)
(long))failure timeout:(void (^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
success	可选	void(^)(NSArray<Device*> *devices)	接口操作成功回调，返回一个Devices数组
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码errorCode定义参考4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
NSArray	一个设备数组，里面存放Device对象

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”
[DeviceApi queryAllDevices:^(NSArray<Device *> *devices){
// do something
}                    failure:^(long errorcode){
// do something
}                    timeout:^{
// do something
}];
```

8.2.6.7 查询单个设备信息

接口功能

查询某个设备的详细信息。

接口原型

```
+ (void)queryDevice:(NSString *)deviceId success:(void(^)(Device *device))success failure:(void(^)
(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	设备标识
success	可选	void(^)(Device *device)	接口操作成功回调，返回一个Device对象

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码errorCode定义参考4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
Device	一个Device对象，里面包含设备详细信息

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”

NSString *deviceId = @”xxxxxxxx”;
[DeviceApi queryDevice:deviceId
success:^(Device *device){
// do something
}         failure:^(long errorcode){
// do something
}             timeout:^{
// do something
}];
```

8.2.6.8 删除设备

接口功能

删除一个设备。

接口原型

```
+ (void)unregisterDevice:(NSString *)deviceId success:(void(^)(void))success failure:(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

或

```
+ (void)unregisterDevice:(NSString *)deviceId timeoutTime:(int)timeInterval success:(void(^)(void))success failure:(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	设备标识

参数	必选/可选	类型	描述
timeInterval	可选	int	设备注册超时时长，单位为秒，默认为60s（不能小于30s）
success	可选	void(^)(void)	接口操作成功回调
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码errorCode定义参考4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”

NSString *deviceId = @”xxxxxxxxx”;
[DeviceApi unregisterDevice:deviceId
success:^{
// do something
}                    failure:^(long errorcode){
// do something
}                    timeout:^{
// do something
}];
```

8.2.6.9 设备重命名

接口功能

对某个设备重命名。

接口原型

```
+ (void)renameDevice:(NSString *)deviceId deviceName:(NSString *)deviceName success:(void(^)
(void))success failure:(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	设备标识
deviceName	必选	NSString	新的设备名
success	可选	void(^)(void)	接口操作成功回调
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码 errorCode定义参考4.5.3
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”

NSString *deviceId = @”xxxxxxxxx”;
NSString *deviceName = @”new name”;
[DeviceApi renameDevice:deviceId
deviceName:deviceName
success:^{
// do something
}                failure:^(long errorcode){
// do something
}                timeout:^{
// do something
}];
```

8.2.6.10 下发设备控制命令

接口功能

对某个设备下发一条控制命令。

接口原型

```
+ (void)sendDeviceControlCommand:(NSString *)deviceId command:(ServiceCommand *)command success:
(void(^)(void))success failure:(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	设备标识
command	必选	ServiceCommand	设备服务命令类，类的属性必须根据平台定义的Service profile赋值
success	可选	void(^)(void)	接口操作成功回调
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码errorCode定义参考4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”

ServiceCommand *command = [[ServiceCommand alloc] init];
command.commandName = @”Test”;
command.commandParameters = @{@”Status”:@”ON”, @”timeout”:@(30)};
command.serviceId = @”TestId”;
[DeviceApi sendDeviceControlCommand:@”deviceId”
command:command
success:^{
// do something
}               failure:^(long errorcode){
// do something
}               timeout:^{
// do something
}];
```

8.2.6.11 设备升级状态变化广播

接口功能

直连设备接入云后，会检测设备是否需要升级，若设备在升级，则会发送该广播；同时升级完成时，也会发送该广播。

接口原型

```
EVENT_DEVICE_UPGRADE_STATUS_CHANGED
```

参数说明

字段	必选/可选	类型	描述
PARAM_UPGRADE_STATUS	必选	NSString	设备升级状态： DEVICE_STATUS_DETAIL_NONE：升级完成 DEVICE_STATUS_DETAIL_UPDATING：正在升级
PARAM_DEVICE	必选	NSString	广播参数，携带设备信息（Device对象）

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
#import “DeviceApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(deviceUpgradeStatusChanged:) name:EVENT_DEVICE_UPGRADE_STATUS_CHANGED
object:nil];
```

8.2.6.12 新设备添加广播

接口功能

应用程序添加新设备，添加成功后，SDK会上报一个新设备添加的广播。

接口原型

```
EVENT_DEVICE_ADDED
```

参数说明

字段	必选/可选	类型	描述
PARAM_DEVICE	必选	NSString	广播参数，携带已添加的设备的信息（Device对象）

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
#import “DeviceApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(deviceDidAdded:)
name:EVENT_DEVICE_ADDED object:nil];
```

8.2.6.13 设备删除广播

接口功能

应用程序删除设备，删除成功后，会收到一个设备删除的广播。

接口原型

```
EVENT_DEVICE_DELETED
```

参数说明

字段	必选/可选	类型	描述
PARAM_DEVICE_ID	必选	NSString	Value是为设备标识deviceId

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
#import “DeviceApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(deviceDidDeleted:)
name:EVENT_DEVICE_DELETED object:nil];
```

8.2.6.14 设备信息变化广播

接口功能

应用程序操作设备（例如重命名），操作成功后，会收到一个设备信息变化的广播。

接口原型

```
EVENT_DEVICE_INFO_CHANGED
```

参数说明

字段	必选/可选	类型	描述
PARAM_DEVICE	必选	NSString	一个Device类型对象参数

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
#import “DeviceApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(deviceInfoDidChanged:)
name:EVENT_DEVICE_INFO_CHANGED object:nil];
```

8.2.6.15 设备服务信息变化广播通知

接口功能

设备服务的信息发生变化时，上报的广播通知，这里主要是指摄像头进入\退出隐私模式。

接口原型

```
EVENT_SERVICE_INFO_CHANGED
```

参数说明

字段	必选/可选	类型	描述
PARAM_DEVICE_ID	必选	NSString	广播参数，携带设备Id
PARAM_SERVICE_INFO	必选	ServiceInfo	广播参数，携带变更后的设备信息（即ServiceInfo实体）

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
#import “DeviceApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(serviceInfoDidChanged:) name: EVENT_SERVICE_INFO_CHANGED object:nil];

……

- (void)serviceInfoDidChanged:(NSNotification *)noti {
    NSString *deviceId = noti.userInfo[PARAM_DEVICE_ID];
    ServiceInfo *serviceInfo = noti.userInfo[PARAM_SERVICE_INFO];
    // do something
}
```

8.2.6.16 设备房间信息变化广播通知

接口功能

设备所属的房间信息变化时，上报的广播通知。

接口原型

```
EVENT_DEVICE_ROOM_INFO_CHANGED
```

参数说明

字段	必选/可选	类型	描述
PARAM_DEVICE_ROOM_INFO	必选	NSDictionary	广播参数，携带变更后的房间Id与设备Id的映射字典，即@{deviceId : roomId}

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
#import “DeviceApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(deviceRoomInfoDidChanged:)
name: EVENT_DEVICE_ROOM_INFO_CHANGED object:nil];

......

- (void)deviceRoomInfoDidChanged:(NSNotification *)noti {
    // key: deviceId value: roomId
    NSDictionary *infoDic = noti.userInfo[PARAM_DEVICE_ROOM_INFO];
    // do something
}
```

8.2.7 摄像头操作

摄像头操作依赖DeviceApi组件，支持开始录像、结束录像、下载录像、抓取录像首帧图等功能。

8.2.7.1 开始视频录像

接口功能

下发控制命令到IP摄像头，开始录制视频。

接口原型

```
+ (void)startVideoRecord:(NSString *)deviceId success:(void(^)(void))success failure:(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	摄像头设备标识
success	可选	void(^)(void)	接口操作成功回调
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后调用。

视频录像有最大录像时长，默认最大时长可以在DM服务器上配置，当前默认最大时长为2分钟，录像完成后会收到EVENT_VIDEO_RECORDING_DONE广播。

调用示例

```
#import “DeviceApi.h”
[DeviceApi startVideoRecord:@”deviceId”
success:^{
// do something
}                    failure:^(long errorcode){
// do something
}                    timeout:^{
// do something
}];
```

8.2.7.2 停止视频录像

接口功能

摄像头设备停止视频录像。

接口原型

```
+ (void)stopVideoRecord:(NSString *)deviceId success:(void(^)(void))success failure:(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	摄像头设备标识
success	可选	void(^)(void)	接口操作成功回调
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后且视频录像正在进行中调用。

调用示例

```
#import “DeviceApi.h”
[DeviceApi stopVideoRecord:@”deviceId”
success:^{
```

```
// do something
}                    failure:^(long errorcode){
// do something
}                    timeout:^{
// do something
}];
```

8.2.7.3 下载视频录像

接口功能

从云存储服务器下载视频录像，将下载后的录像保存到iOS系统相册中。

接口原型

```
+ (void)downloadVideoRecord:(NSString *)srcUrl success:(void(^)(void))success failure:(void(^)
(long errorCode))failure;
```

参数说明

参数	必选/可选	类型	描述
srcUrl	必选	NSString	视频下载地址（参考视频录制结果广播通知和常用数据结构章节，获取视频地址）。
success	可选	void(^)(void)	接口操作成功回调
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.5.3

返回结果

类型	描述
void	无返回值

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”
[DeviceApi downloadVideoRecord:@” srcUrl”
success:^{
// do something
}                        failure:^(long errorcode){
// do something
}];
```

8.2.7.4 取消正在下载的视频录像

接口功能

取消正在下载的视频录像，并且把已下载的部分数据清除。

接口原型

```
+ (void)cancelDownloadingVideoRecord
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

当正在下载视频录像的时候，想取消本次下载，就可以调用本接口。

调用示例

```
#import “DeviceApi.h”
[DeviceApi cancelDownloadingVideoRecord];
```

8.2.7.5 抓取摄像头当前图像

接口功能

抓取摄像头当前图像，成功时返回图像URL。

接口原型

```
+ (void)captureImage:(NSString *)deviceId success:(void(^)(NSString *imageUrl))success failure:
(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	摄像头设备标识
success	可选	void(^)(NSString *)	接口操作成功回调， 携带IP摄像头所抓拍的图片URL
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
NSString	携带一个图像URL

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”
[DeviceApi captureImage:@” deviceId”
success:^(NSString *imageUrl){
// do something
}                failure:^(long errorcode){
// do something
}                timeout:^{
// do something
}];
```

8.2.7.6 视频录像完成广播

接口功能

视频录像完成之后，SDK会发送此广播。

接口原型

```
EVENT_VIDEO_RECORDING_DONE
```

参数说明

字段	必选/可选	类型	描述
PARAM_VIDEO_RECORDING_INFO	必选	VideoRecordingInfo	广播参数，携带视频录制结果对象

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
#import “DeviceApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(handleVideoRecordingDone:) name: EVENT_VIDEO_RECORDING_DONE object:nil];
```

8.2.7.7 查询云存储信息

接口功能

查询云存储信息。

接口原型

```
+ (void)queryCloudStorageInfo:(void(^)(CloudStorageSpace *space))success failure:(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
success	可选	void(^)(CloudStorageSpace *)	接口操作成功回调，携带CloudStorageSpace对象
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
CloudStorageSpace	一个CloudStorageSpace对象，里面包含dm配置信息

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”
[DeviceApi queryCloudStorageInfo:^(CloudStorageSpace *space){
}    failure:^(long errorcode){
}    timeout:^{
}];
```

8.2.7.8 查询视频录制最大时长

接口功能

查询视频录制时，允许录制的最大时长。

接口原型

```
+ (void)queryMaxRecordTime:(void(^)(NSIntegermaxRecordTime))success failure:(void(^)(long
errorCode))failure timeout:(void(^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void(^)(NSInteger)	接口操作成功回调，携带一个视频录制最大时长的整型参数
failure	可选	void(^)(long errorcode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
NSInteger	携带一个视频录制最大时长

注意事项

本接口必须在登录成功之后调用。

调用示例

```
#import “DeviceApi.h”
[DeviceApi queryMaxRecordTime:^(NSInteger maxRecordTime) {
    // do something
} failure:^(long errorcode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.7.9 云存储状态变化广播

接口功能

云存储空间快要满时，会收到这个广播。

接口原型

```
EVENT_CLOUD_STORAGE_STATUS_CHANGED
```

参数说明

参数	必选/可选	类型	描述
PARAM_CLOUD_STORAGE_STATUS	必选	NSNumber	广播参数，携带云存储空间状态

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “DeviceApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(cloudStorageStatusDidChanged:) name: EVENT_CLOUD_STORAGE_STATUS_CHANGED
object:nil];
…
- (void) cloudStorageStatusDidChanged:(NSNotification *)noti
{
    NSDictionary *userInfo = noti.userInfo;
    NSInteger status = [userInfo [PARAM_CLOUD_STORAGE_STATUS] integerValue];
    // do something
}
```

8.2.8 视频播放

视频播放依赖DeviceApi组件，支持播放实时视频，语音对讲和历史视频功能。

8.2.8.1 初始化播放实时视频对象

接口功能

初始化播放实时视频对象,使用此对象可以查看实时视频。

接口原型

```
- (instancetype)initWithMediaPlayerType:(MEDIA_PLAYER_TYPE)type;
```

参数说明

参数	必选/可选	类型	描述
type	必选	MEDIA_PLAYER_TYPE	媒体播放器类型，参数设置为MEDIA_PLAYER_TYPE_LIVE，可以初始化播放实时视频对象

返回结果

类型	描述
MediaPlayer	媒体播放器对象

调用示例

```
#import “DeviceApi.h”
MediaPlayer *player = [[MediaPlayer alloc] initWithMediaPlayerType: MEDIA_PLAYER_TYPE_LIVE];
```

8.2.8.2 初始化播放历史视频对象

接口功能

初始化播放历史视频对象,使用此对象可以播放摄像头已录制的视频。

接口原型

```
- (instancetype)initWithMediaPlayerType:(MEDIA_PLAYER_TYPE)type;
```

参数说明

参数	必选/可选	类型	描述
type	必选	MEDIA_PLAYER_TYPE	媒体播放器类型，参数设置为MEDIA_PLAYER_TYPE_VOD，可以初始化播放历史视频对象

返回结果

类型	描述
MediaPlayer	媒体播放器对象

调用示例

```
#import “DeviceApi.h”
MediaPlayer *player = [[MediaPlayer alloc] initWithMediaPlayerType: MEDIA_PLAYER_TYPE_VOD];
```

8.2.8.3 获取播放界面

接口功能

获取播放视频时的界面

接口原型

```
@property (nonatomic, readonly) UIView *view;
```

返回结果

类型	描述
UIView	播放视频界面

调用示例

```
#import “DeviceApi.h”
MediaPlayer *player = [[MediaPlayer alloc] initWithMediaPlayerType: MEDIA_PLAYER_TYPE_LIVE];
[self.view addSubview:player.view];
```

8.2.8.4 播放实时视频

接口功能

播放实时视频。

接口原型

```
- (void)playWithLiveVideo:(NSString *)deviceId;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	摄像头设备标识

返回结果

类型	描述
void	无返回值

调用示例

```
#import “DeviceApi.h”
MediaPlayer *player = [[MediaPlayer alloc] initWithMediaPlayerType: MEDIA_PLAYER_TYPE_LIVE];
[self.view addSubview:player.view];
[player playWithLiveVideo:@”deviceId”];
```

8.2.8.5 播放历史视频

接口功能

播放历史视频。

接口原型

```
- (void)playWithVodVideo:(NSString *)videoUrl;
```

参数说明

参数	必选/可选	类型	描述
videoUrl	必选	NSString	视频录像URL

返回结果

类型	描述
void	无返回值

调用示例

```
#import “DeviceApi.h”
MediaPlayer *player = [[MediaPlayer alloc] initWithMediaPlayerType: MEDIA_PLAYER_TYPE_VOD];
[self.view addSubview:player.view];
[player playWithVodVideo:@”videoUrl”];
```

8.2.8.6 停止视频播放

接口功能

停止视频播放。

接口原型

```
- (void)stop;
```

参数说明

无。

返回结果

类型	描述
void	无返回值

调用示例

```
#import “DeviceApi.h”
MediaPlayer *player = [[MediaPlayer alloc] initWithMediaPlayerType: MEDIA_PLAYER_TYPE_VOD];
[player playWithVodVideo:@” videoUrl”];
//…
[player stop];
```

8.2.8.7 获取历史视频本地播放路径

接口功能

历史视频下载到系统相册后，生成的一个FileUrl路径，如果历史视频未下载，则返回空。

接口原型

```
- (void)getLocalHistoryVideoPathWithVideoUrl:(NSString*)videoUrl completionHandler:(void (^)(NSString *path))handler
```

参数说明

参数	必选/可选	类型	描述
videoUrl	必选	NSString	历史视频Url
handler	可选	void (^)(NSString *)	接口操作完成回调，返回历史视频的本地播放路径

返回结果

类型	描述
NSString	返回历史视频的本地播放路径

调用示例

```
#import “DeviceApi.h”
NSString *videoUrl = @”xxxxxxxxx”;
MediaPlayer *player = [[MediaPlayer alloc] initWithMediaPlayerType:MEDIA_PLAYER_TYPE_VOD];
[player getLocalHistoryVideoPathWithVideoUrl:@”videoUrl” completionHandler:^(NSString *path) {
    if (path.length> 0) {
        [player playWithVodVideo:path];
    }
}];
```

8.2.9 事件处理

事件处理依赖EventApi组件，提供事件（设备状态变化产生历史记录及人为操作产生的历史记录）的查询、删除、上报功能。

8.2.9.1 初始化

接口功能

该接口建议在APP登录成功后进行调用，提供了事件实时上报的广播推送。

接口原型

```
+(void)init;
```

参数说明

无

返回结果

类型	结果
void	无返回值

注意事项

调用事件API接口之前，必须先进行初始化，即调用本接口。

调用示例

```
[EventApi  init];
```

8.2.9.2 销毁

接口功能

事件API销毁，释放内存。

接口原型

```
+ (void)destroy;
```

参数说明

无

返回结果

类型	结果
void	无返回值

注意事项

退出登录时必须调用该接口销毁内存。

调用示例

```
//objc code
[EventApi  destroy];
```

8.2.9.3 查询所有事件

接口功能

按时间倒序分页查询所有事件的功能。

接口原型

```
+ (void)queryAllEvents:(NSInteger)pageNumber pageSize:(NSInteger)pageSize success:(void(^)(NSArray
<Event *>*everythingEvents))success failure:(void(^)(long errorCode))failure timeout:(void(^)
(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
pageNumber	必选	NSInteger	页码
pageSize	必选	NSInteger	每页容纳事件数量

参数	必选/可选	类型	描述
success	可选	void(^)(NSArray <Event *>*everythingEvents)	接口操作成功回调，携带一个Event的数组
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	结果
NSArray	返回一个数组，数组中存放Event对象

注意事项

接口必须是在登录成功之后调用。

调用示例

```
[EventApi queryAllEvents:0 pageSize:40 success:^(NSArray<Event *> *everythingEvents) {
} failure:^(long errorCode) {
} timeout:^{
}];
```

8.2.9.4 查询设备事件

接口功能

按时间倒序分页查询所有事件的功能。

接口原型

```
+ (void)queryDeviceEvents:(NSString *)deviceId pageNumber:(NSInteger)pageNumber pageSize:
(NSInteger)pageSize success:(void (^)(NSArray <Event *>*deviceEvents))success failure:(void (^)
(long errorCode))failure timeout:(void (^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	设备Id
pageNumber	必选	NSInteger	页码
pageSize	必选	NSInteger	每页容纳事件数量

参数	必选/可选	类型	描述
success	可选	void(^)(NSArray <Event *>*deviceEvents)	接口操作成功回调，携带一个Event的数组
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	结果
NSArray	返回一个数组，数组中存放Event对象

注意事项

接口必须是在登录成功之后调用。

调用示例

```
[EventApi queryDeviceEvents:@"ccc677dc-7495-40f4-b7b1-08711fb3470d" pageNumber:0 pageSize:40
success:^(NSArray<Event *> *deviceEvents) {
NSLog(@"queryDeviceEvents success");
} failure:^(long errorCode) {
NSLog(@"queryDeviceEvents fail");
} timeout:^{
NSLog(@"queryDeviceEvents timeout");
}];
```

8.2.9.5 查询录像事件

接口功能

按时间倒序分页查询录像事件的功能。

接口原型

```
+ (void)queryRecordEvents:(NSInteger)pageNumber pageSize:(NSInteger)pageSize success:(void (^)
(NSArray<Event *> *))success failure:(void (^)(long errorCode))failure timeout:(void (^)
(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
pageNumber	必选	NSInteger	页码
pageSize	必选	NSInteger	每页容纳事件数量

参数	必选/可选	类型	描述
success	可选	void(^)(NSArray <Event *>*recordEvents)	接口操作成功回调，携带一个Event的数组
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	结果
NSArray	返回一个数组，数组中存放Event对象

注意事项

接口必须是在登录成功之后调用。

调用示例

```
[EventApi queryRecordEvents:0 pageSize:40 success:^(NSArray<Event *> *recordEvents) {
NSLog(@"queryRuleEvents success");
} failure:^(long errorCode) {
NSLog(@"queryRuleEvents fail");
} timeout:^{
NSLog(@"queryRuleEvents timeout");
}];
```

8.2.9.6 删除所有事件

接口功能

提供了一次性删除所有事件的功能。

接口原型

```
+ (void)deleteAllEventsSuccess:(void(^)(void))success failure:(void(^)(long errorCode))failure
timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
success	可选	void(^)(void)	接口操作成功回调

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	结果
void	无返回值

注意事项

接口必须是在登录成功之后调用,该接口执行成功后，包括存储的图像和录像数据都会被删除且无法恢复。

调用示例

```
[EventApi deleteAllEventsSuccess:^{
NSLog(@"deleteAllEvents success");
} failure:^(long errorCode) {
NSLog(@"deleteAllEvents fail");
} timeout:^{
NSLog(@"deleteAllEvents timeout");
}];
```

8.2.9.7 删除指定事件

接口功能

提供了删除部分事件功能。

接口原型

```
+ (void)deleteEvents:(NSArray <NSString *>*)eventIds success:(void(^)(void))success failure:
(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
eventIds	必选	NSArray	存储事件Id的数组
success	可选	void(^)(void)	接口操作成功回调

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	结果
void	无返回值

注意事项

接口必须是在登录成功之后调用,该接口执行成功后，包括存储的图像和录像数据都会被删除且无法恢复。

调用示例

```
[EventApi deleteEvents:@[@"f988f52a-63c8-48cd-a5b3-0a4306533afd"] success:^{
NSLog(@"deleteEvents success");
} failure:^(long errorCode) {
NSLog(@"deleteEvents fail");
} timeout:^{
NSLog(@"deleteEvents timeout");
}];
```

8.2.9.8 新事件消息广播

接口功能

当有设备状态变更和认为操作产生历史事件记录时会生成一条广播。

接口原型

```
EVENT_NEW_EVENT_INCOMING
```

参数说明

字段	必选/可选	类型	描述
PARAM_EVENT	必选	NSString	广播参数，携带一个Event实体

返回结果

类型	结果
void	无返回值

注意事项

无

调用示例

```
#import “EventApi.h”

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(newEventIncoming:) name:
EVENT_NEW_EVENT_INCOMING
object:nil];
```

8.2.10 离线推送

离线推送依赖PushApi组件，包含开启离线推送等接口，以及开启离线推送结果的广播、收到离线推送消息的广播。

8.2.10.1 开启离线推送

接口功能

该接口必须在APP登录成功后调用，接口内部会注册苹果推送通知服务(APNs)，调用成功会开启离线推送功能。

接口原型

```
+ (void)enablePush;
```

参数说明

无

返回结果

类型	结果
void	无返回值

注意事项

该接口必须在APP登录成功后进行调用，否则会失败。

根据苹果规定的iOS注册远程通知的步骤，在注册苹果推送通知服务(APNs)前，必须注册App支持的用户通知交互类型，用户通知交互类型有App图标badge、声音、告警三种类型，UI可根据不同的需求选择。

调用示例

```
UIUserNotificationType types = (UIUserNotificationType) (UIUserNotificationTypeSound |
UIUserNotificationTypeAlert);
UIUserNotificationSettings *mySettings =  [UIUserNotificationSettings settingsForTypes:types
categories:nil];
```

```
[[UIApplication sharedApplication] registerUserNotificationSettings:mySettings];

[PushApi enablePush];
```

8.2.10.2 App 注册苹果离线推送服务成功

接口功能

UI在调用开启离线推送接口后，如果成功注册了苹果推送通知服务(APNs)，则iOS系统会调用UIApplicationDelegate协议的- (void)application:(**UIApplication** *)*application*

didRegisterForRemoteNotificationsWithDeviceToken:(**NSData** *)*deviceToken方法*，UI需要在上述方法中调用该接口。

接口原型

```
+ (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken;
```

参数说明

参数	必选/可选	类型	描述
application	必选	UIApplication	应用程序对象
deviceToken	必选	NSData	APNs返回的deviceToken

返回结果

类型	结果
void	无返回值

注意事项

AppDelegate中需要实现UIApplicationDelegate协议的- (void)application:(UIApplication *)application

didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken方法，UI需要在上述方法中调用该接口。

调用示例

```
@implementation AppDelegate
//…
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
[PushApi application:application didRegisterForRemoteNotificationsWithDeviceToken: deviceToken];
}
```

8.2.10.3 App 注册苹果离线推送服务失败

接口功能

UI在调用开启离线推送接口后，如果注册苹果推送通知服务(APNs)失败，则iOS系统会调用UIApplicationDelegate协议的- (void)application:(UIApplication *)application

didFailToRegisterForRemoteNotificationsWithError:(NSError *)error*方法*，UI需要在上述方法中调用该接口。

接口原型

```
+ (void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWithError:(NSError *)error;
```

参数说明

参数	必选/可选	类型	描述
application	必选	UIApplication	应用程序对象
error	必选	NSError	注册苹果离线推送服务失败的error对象

返回结果

类型	结果
void	无返回值

注意事项

AppDelegate中需要实现UIApplicationDelegate协议的- (void)application:(UIApplication *)application

didFailToRegisterForRemoteNotificationsWithError:(NSError *)error方法，UI需要在上述方法中调用该接口。

调用示例

```
@implementation AppDelegate
//…
- (void)application:(UIApplication *)application
didFailToRegisterForRemoteNotificationsWithError:(NSError *)error {
[PushApi application:application didFailToRegisterForRemoteNotificationsWithError:error];
}
```

8.2.10.4 App 收到离线推送消息

8.2.10.5 开启离线推送结果广播

接口功能

UI调用开启离线推送接口后，会收到此广播，该广播会指示开启离线推送成功或失败。

接口原型

```
EVENT_ENABLE_PUSH_RESULT
```

参数说明

字段	必选/可选	类型	描述
PARAM_ENABLE_PUSH_RESULT	必选	NSNumber	开启离线推送成功或失败，对应的值参考枚举ENABLE_PUSH_RESULT
PARAM_ENABLE_PUSH_RESULT_FAILED_REASON	可选	NSNumber	开启离线推送失败的原因，对应的值可能有： ERROR_CODE_PUSH_LOGIN_STATUS_NOT_CONNECTED， ERROR_CODE_PUSH_FAIL_TO_REGISTER_FOR_REMOTE_NOTIFICATIONS， ERROR_CODE_PUSH_FAIL_TO_SEND_TOKEN_TO_SERVER

返回结果

类型	结果
void	无返回值

注意事项

无

调用示例

```
#import “PushApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(handleEnablePushResult:)
name: EVENT_ENABLE_PUSH_RESULT object:nil];
```

8.2.10.6 收到离线推送消息广播

接口功能

App在运行状态下，如果收到了服务器的推送消息，则在UI调用了App收到离线推送消息接口后，SDK会发送此广播。

接口原型

```
EVENT_RECEIVE_OFFLINE_PUSH
```

参数说明

字段	必选/可选	类型	描述
PARAM_OFFLINE_PUSH_EVENT_ID	必选	NSString	事件ID，事件的唯一标示
PARAM_OFFLINE_PUSH_MESSAGE	必选	NSString	离线推送消息

返回结果

类型	结果
void	无返回值

注意事项

无

调用示例

```
#import “PushApi.h”
[[NSNotificationCenter defaultCenter]
addObserver:selfselector:@selector(handleReceiveOfflinePush:) name: EVENT_RECEIVE_OFFLINE_PUSH
object:nil];
```

8.2.11 规则处理

规则处理依赖RuleApi组件，包括规则创建、删除、编辑、激活去激活、手动触发、广播上报等功能。

8.2.11.1 初始化

接口功能

该接口建议在APP登陆成功后进行调用，提供了规则实时上报的广播推送。

接口原型

```
+(void)init;
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

在调用其他规则API接口之前，必须先进行初始化，即调用本接口。

调用示例

```
[RuleApi  init];
```

8.2.11.2 销毁

接口功能

规则API销毁，释放内存

接口原型

```
+ (void)destroy;
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

应用程序退出前，必须销毁规则API。

调用示例

```
[RuleApi  destroy];
```

8.2.11.3 创建规则

接口功能

创建规则

接口原型

```
+ (void)createRule:(Rule *)rule success:(void(^)(Rule *rule))success failure:(void(^)(long
errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
rule	必选	Rule	需要创建的规则
success	可选	void(^)(Rule *)	接口操作成功回调。返回创建好的规则
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.7.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
Rule	返回创建好的规则

注意事项

接口必须是在登陆成功之后调用

调用示例

```
[RuleApi createRule:myRule success:^(Rule *rule){
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}];
```

8.2.11.4 更新规则

接口功能

更新规则

接口原型

```
+ (void)updateRule:(Rule *)rule success:(void(^)(Rule *rule))success failure:(void(^)(long
errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
rule	必选	Rule	需要更新的规则

参数	必选/可选	类型	描述
success	可选	void(^)(Rule *)	接口操作成功回调。返回更新后的规则
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.7.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
Rule	返回的更新过后的规则

注意事项

接口必须是在登陆成功之后调用

调用示例

```
[RuleApi updateRule:myRule success:^(Rule *rule){
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}];
```

8.2.11.5 查询所有规则

接口功能

查询所有规则

接口原型

```
+ (void)queryAllRulesSuccess:(void(^)(NSArray <Rule *> *rules))success failure:(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
success	可选	void(^)(NSArray <Rule *> *rules)	接口操作成功回调，携带查询到的规则数组

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.7.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
NSArray<Rule *>	返回规则列表

注意事项

接口必须是在登陆成功之后调用

调用示例

```
[RuleApi queryAllRulesSuccess:^(NSArray<Rule *> *rules) {
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}];
```

8.2.11.6 查询指定规则

接口功能

查询指定规则

接口原型

```
+ (void)queryRule:(NSInteger)ruleId success:(void(^)(Rule *rule))success failure:(void(^)(long
errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	NSInteger	规则id
success	可选	void(^)(Rule *rule)	接口操作成功回调，携带查询到的规则

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.7.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
Rule	返回查询的指定规则

注意事项

接口必须是在登陆成功之后调用

调用示例

```
[RuleApi queryRule:11111 success:^(Rule *rule) {
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}];
```

8.2.11.7 激活规则

接口功能

激活规则，规则可以触发。

接口原型

```
+ (void)activeRule:(NSInteger)ruleId success:(void(^)(void))success failure:(void(^)(long
errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	NSInteger	规则id
success	可选	void(^)(void)	接口操作成功回调

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.7.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
[RuleApi activeRule:11111 success:^{
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}];
```

8.2.11.8 去激活规则

接口功能

去激活规则，规则不可以触发。

接口原型

```
+ (void)inactiveRule:(NSInteger)ruleId success:(void(^)(void))success failure:(void(^)(long
errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	NSInteger	规则id
success	可选	void(^)(void)	接口操作成功回调

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.7.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
[RuleApi inactiveRule:11111 success:^{
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}];
```

8.2.11.9 手动触发规则

接口功能

创建的手动场景（规则）可以调用该接口直接触发。

接口原型

```
+ (void)runRule:(NSInteger)ruleId success:(void(^)(NSArray *resultArray))success failure:(void(^)
(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	NSInteger	规则id
success	可选	void(^)(NSArray *)	接口操作成功回调。返回触发失败的设备id数组

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.7.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
NSArray	返回触发失败的设备id数组

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
[RuleApi runRule:11111 success:^(NSArray *resultArray) {
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}];
```

8.2.11.10 批量激活规则

接口功能

批量激活多个规则，多个规则可以触发。

接口原型

```
+ (void)batchActiveRules:(NSArray <NSNumber *>*)ruleIds success:(void(^)(void))success failure:
(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
ruleIds	必选	NSArray	规则id数组
success	可选	void(^)(void)	接口操作成功回调

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.7.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
[RuleApi batchActiveRules:@[@(11111), @(22222), @(33333)] success:^{
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}];
```

8.2.11.11 批量去激活规则

接口功能

批量去激活规则，多个规则不可以触发。

接口原型

```
+ (void)batchInactiveRules:(NSArray <NSNumber *>*)ruleIds success:(void(^)(void))success failure:
(void(^)(long errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
ruleIds	必选	NSArray	规则id数组
success	可选	void(^)(void)	接口操作成功回调

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.7.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
[RuleApi batchInactiveRules:@[@(11111), @(22222), @(33333)] success:^{
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}];
```

8.2.11.12 删除规则

接口功能

删除指定规则

接口原型

```
+ (void)deleteRule:(NSInteger)ruleId success:(void(^)(void))success failure:(void(^)(long
errorCode))failure timeout:(void(^)(void))timeout;
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选		规则id
success	可选	void(^)(void)	接口操作成功回调
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码

参数	必选/可选	类型	描述
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

接口必须是在登陆成功之后调用。

调用示例

```
[RuleApi deleteRule:11111 success:^{
//success block
} failure:^(long errorCode) {
//failure block
} timeout:^{
//timeout block
}]
```

8.2.11.13 查询规则录像时长

接口功能

查询规则录像时，允许设置的录像时长

接口原型

```
+ (void)queryRuleRecordTime:(void (^)(NSString *recordTime))successfailure:(void (^)(long
errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void(^)(NSString *)	接口操作成功回调。返回规则录像时长
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
NSString	返回规则录像时长

注意事项

接口必须是在登录成功之后调用。

调用示例

```
[RuleApi queryRuleRecordTime:^( NSString *recordTime) {
    // success block
} failure:^(long errorCode) {
    // failure block
} timeout:^{
    // timeout block
}];
```

8.2.11.14 新规则被创建消息广播

接口功能

在同一个homeId下，当A用户创建了新规则后，除了A用户以外，所有用户都会收到新规则被创建的消息广播。

接口原型

```
EVENT_RULE_CREATED
```

参数说明

字段	必选/可选	类型	描述
PARAM_RULE	必选	NSString	广播参数，携带新创建的规则实例

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(ruleDidCreateNty:)
name:EVENT_RULE_CREATED object:nil];
- (void)ruleDidCreateNty:(NSNotification *)sender {
NSDictionary *userInfo = sender.userInfo;
```

```
Rule *createRule = userInfo[PARAM_RULE];
}
```

8.2.11.15 规则更新消息广播

接口功能

在同一个homeId下，当A用户编辑某个规则后，除了A用户以外，所有用户都会收到规则更新的消息广播。

接口原型

```
EVENT_RULE_INFO_CHANGED
```

参数说明

字段	必选/可选	类型	描述
PARAM_RULE	必选	NSString	广播参数，携带更新后的规则实例

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(ruleDidUpdateNty:)
name:EVENT_RULE_CREATED object:nil];
- (void)ruleDidUpdateNty:(NSNotification *)sender {
NSDictionary *userInfo = sender.userInfo;
Rule *updateRule = userInfo[PARAM_RULE];
}
```

8.2.11.16 删除规则消息广播

接口功能

在同一个homeId下，当A用户删除某个规则后，除了A用户以外，所有用户都会收到规则被删除的消息广播。

接口原型

```
EVENT_RULE_DELETED
```

参数说明

字段	必选/可选	类型	描述
PARAM_RULE	必选	NSString	广播参数，携带被删除的规则实例

返回结果

类型	描述
void	无返回值

注意事项

无

消息示例

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(ruleDidDeleteNty:)
name:EVENT_RULE_CREATED object:nil];
- (void)ruleDidDeleteNty:(NSNotification *)sender {
NSDictionary *userInfo = sender.userInfo;
Rule *deleteRule = userInfo[PARAM_RULE];
}
```

8.2.12 房间管理

房间管理依赖RoomApi组件，包括查询房间列表、创建房间、更新房间、删除房间、添加设备到房间这些功能。

8.2.12.1 初始化

接口功能

初始化RoomApi

接口原型

```
+ (void) init
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[RoomApi init];
```

8.2.12.2 销毁

接口功能

释放RoomApi的资源。

接口原型

```
+ (void) destroy
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[RoomApi destroy];
```

8.2.12.3 创建房间

接口功能

创建一个房间

接口原型

```
+ (void)createRoom:(Room *)room success:(void (^)(Room *room)successfailure:(void (^)(long
errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
room	必选	Room	要创建的房间实体
success	可选	void(^)(Room *)	接口操作成功回调。返回创建好的房间实体

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.8.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
Room	返回创建好的房间实体

注意事项

不能创建默认房间default

调用示例

```
#import “RoomApi.h”
Room *newRoom = [[Room alloc] init];
newRoom.roomName = @”This is a room”;
newRoom.roomDescription = @”Test”;
[RoomApi createRoom:newRoom success:^(Room *room) {
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.12.4 删除房间

接口功能

删除一个房间

接口原型

```
+ (void)deleteRoom:(NSInteger)roomId success:(void (^)(void)successfailure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
roomId	必选	NSInteger	要删除的房间Id
success	可选	void(^)(void)	接口操作成功回调。

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.8.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

不能删除默认房间default

调用示例

```
#import “RoomApi.h”
Room *deleteRoom;
[RoomApi deleteRoom:deleteRoom.roomId success:^{
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.12.5 查询房间列表

接口功能

查询房间列表

接口原型

```
+ (void)queryAllRooms:(void (^)(NSArray<Room *> *rooms)successfailure:(void (^)(long
errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void(^)(NSArray *)	接口操作成功回调。返回房间列表

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
NSArray	返回房间列表

注意事项

无

调用示例

```
#import “RoomApi.h”
[RoomApi queryAllRooms:^(NSArray<Room*> *rooms) {
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.12.6 更新房间

接口功能

更新房间信息

接口原型

```
+ (void)updateRoom:(Room *)room success:(void (^)(Room *room)successfailure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
room	必选	Room	要更新的房间实体
success	可选	void(^)(Room *)	接口操作成功回调。返回更新后的房间实体

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
Room	返回更新后的房间实体

注意事项

无

调用示例

```
#import “RoomApi.h”
Room *updateRoom;
updateRoom.roomName = @”This is a room”;
updateRoom.roomDescription = @”Test”;
[RoomApi updateRoom:updateRoom
success:^( Room *room) {
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.12.7 添加设备到房间

接口功能

添加设备到房间，可以把多个设备添加到同一房间，也可以把多个设备各自添加到不同的房间。

接口原型

```
+ (void)addDevicesToRoom:(NSDictionary<NSString*,NSNumber *> *)dict success:(void(^)(void)success
failure:(void (^)(long errorCode))failure timeout:(void(^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
dict	必选	NSDictionary	设备Id与房间Id的映射字典

参数	必选/可选	类型	描述
success	可选	void(^)(void)	接口操作成功回调。
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.8.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “RoomApi.h”
NSDictionary *dict = @{@”deviceId1” : @(roomId1), @”deviceId2”
: @(roomId1), @”deviceId3” : @”roomId2”};
[RoomApi addDevicesToRoom:dict success:^{
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.12.8 创建房间广播通知

接口功能

其它用户创建房间成功后，上报的广播通知。

接口原型

```
EVENT_ROOM_CREATED
```

参数说明

参数	必选/可选	类型	描述
PARAM_ROOM	必选	NSString	广播参数，携带创建成功后的房间信息（Room对象）

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “RoomApi.h”
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(roomDidCreated:) name: EVENT_ROOM_CREATED object:nil];

……

- (void)roomDidCreated:(NSNotification *)noti {
    Room *room = noti.userInfo[PARAM_ROOM];
    // do something
}
```

8.2.12.9 删除房间广播通知

接口功能

其它用户删除房间成功后，上报的广播通知。

接口原型

```
EVENT_ROOM_DELETED
```

参数说明

参数	必选/可选	类型	描述
PARAM_ROOM_ID	必选	NSNumber	广播参数，携带删除成功后的房间Id

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(roomDidDeleted:) name: EVENT_ROOM_ DELETED object:nil];
```

```
......

- (void)roomDidDeleted:(NSNotification *)noti {
    NSNumber *roomId = noti.userInfo[PARAM_ ROOM_ID];
    // do something
}
```

8.2.12.10 更新房间广播通知

接口功能

其它用户更新房间信息成功后，上报的广播通知。

接口原型

```
EVENT_ROOM_UPDATED
```

参数说明

参数	必选/可选	类型	描述
PARAM_ROOM	必选	NSString	广播参数，携带更新成功后的房间信息（Room对象）

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(roomDidUpdated:) name: EVENT_ROOM_UPDATED object:nil];

......

- (void) roomDidUpdated:(NSNotification *)noti {
    Room *room = noti.userInfo[PARAM_ROOM];
    // do something
}
```

8.2.13 用户设置

用户设置依赖SettingsApi组件，包括查询事件push通知设置、开启\关闭事件push通知设置、查询与修改云存储空间满时自动清理开关配置、设置自动收集日志配置等功能。

8.2.13.1 初始化

接口功能

初始化SettingsApi

接口原型

```
+ (void) init
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[Settings init];
```

8.2.13.2 销毁

接口功能

释放SettingsApi的资源。

接口原型

```
+ (void) destroy
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

应用程序退出前，必须销毁设置API

调用示例

```
[SettingsApi destroy];
```

8.2.13.3 查询事件的通知设置

接口功能

查询事件的push通知设置

接口原型

```
+ (void)queryEventNotificationSetting:(void (^)(NSDictionary<NSString*, NSString *>
*settings)success failure:(void (^)(long errorCode))failure timeout:(void(^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void(^)(NSDictionary *)	接口操作成功回调。返回通知类型与激活状态的映射关系字典
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
NSDictionary	返回通知类型与激活状态的映射关系字典

注意事项

无

调用示例

```
#import “SettingsApi.h”
[SettingsApi queryEventNotificationSetting:^(NSDictionary<NSString*, NSString *> *settings) {
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.13.4 激活事件的通知设置

接口功能

激活事件的push通知设置

接口原型

```
+ (void)enableEventNotificationSettingWithEventTypes:(NSArray<NSString*> *)eventTypes success:
(void (^)(void)success failure:(void (^)(longerrorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
eventTypes	必选	NSArray<NSString *>	事件通知的类型
success	可选	void(^)(void)	接口操作成功回调。
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “SettingsApi.h”
NSArray *eventTypes = @[@”low power”, @”device disconnection”];
[SettingsApi enableEventNotificationSettingWithEventTypes:eventTypes success:^{
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.13.5 去激活事件的通知设置

接口功能

去激活事件的push通知设置

接口原型

```
+ (void)disableEventNotificationSettingWithEventTypes:(NSArray<NSString*> *)eventTypes success:
(void (^)(void)success failure:(void (^)(longerrorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
eventTypes	必选	NSArray<NSString *>	事件通知的类型
success	可选	void(^)(void)	接口操作成功回调。
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “SettingsApi.h”
NSArray *eventTypes = @[@” low power”, @” device disconnection”];
[SettingsApi disableEventNotificationSettingWithEventTypes:eventTypes success:^{
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.13.6 设置云存储空间满时自动清理配置开关

接口功能

设置云存储空间满时自动清理配置开关

接口原型

```
+ (void)setCloudStorageSpaceAutoClean:(BOOL)autoCleansuccess:(void (^)(void)success failure:(void (^)(long errorCode))failure timeout:(void(^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
autoClean	必选	BOOL	是否自动清理

参数	必选/可选	类型	描述
success	可选	void(^)(void)	接口操作成功回调。
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “SettingsApi.h”
[SettingsApi setCloudStorageSpaceAutoClean:YES success:^{
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.13.7 查询云储存空间自动清理开关

接口功能

查询云存储空间自动清理开关

接口原型

```
+ (void)queryCloudStorageSpaceAutoClean:(void (^)(BOOL autoClean)success failure:(void (^)(long errorCode))failure timeout:(void(^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
success	可选	void(^)(BOOL)	接口操作成功回调。返回是否自动清理

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
BOOL	返回是否自动清理

注意事项

无

调用示例

```
#import “SettingsApi.h”
[SettingsApi queryCloudStorageSpaceAutoClean:^(BOOL autoClean) {
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.13.8 设置自动收集日志配置开关

接口功能

设置自动收集日志配置开关

接口原型

```
+ (void)setAutoUploadLog:(BOOL)autoUpload success:(void (^)(void)successfailure:(void (^)(long errorCode))failure timeout:(void (^)(void))timeout
```

参数说明

参数	必选/可选	类型	描述
autoUpload	必选	BOOL	是否允许自动收集日志
success	可选	void(^)(void)	接口操作成功回调。

参数	必选/可选	类型	描述
failure	可选	void(^)(long errorCode)	接口操作失败回调，携带一个整型错误码 errorCode定义详见4.2
timeout	可选	void(^)(void)	接口操作超时回调

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “SettingsApi.h”
[SettingsApi setAutoUploadLog:YES success:^{
    // do something
} failure:^(long errorCode) {
    // do something
} timeout:^{
    // do something
}];
```

8.2.13.9 云存储空间自动清理开关变化广播通知

接口功能

云存储空间自动清理配置开关变更后，上报的广播通知

接口原型

```
EVENT_CLOUD_STORAGE_SPACE_AUTO_CLEAN_CHANGD
```

参数说明

参数	必选/可选	类型	描述
PARAM_AUTO_CLEAN	必选	BOOL	广播参数，携带变更后的自动清理开关配置

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(cloudStorageSpaceAutoCleanDidChanged:) name:
EVENT_CLOUD_STORAGE_SPACE_AUTO_CLEAN_CHANGD object:nil];

......

- (void)cloudStorageSpaceAutoCleanDidChanged:(NSNotification *)noti {
    BOOL autoClean = [noti.userInfo[PARAM_AUTO_CLEAN] boolValue];
    // do something
}
```

8.2.14 本地存储

本地存储依赖LocalStorageApi组件，包括房间、设备、规则、事件等模型的存储功能，仅开放给VIP用户使用。

8.2.14.1 初始化

接口功能

初始化LocalStorageApi组件

接口原型

```
+ (void) init
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[LocalStorageApi init];
```

8.2.14.2 销毁

接口功能

LocalStorageApi销毁，释放内存。

接口原型

```
+ (void) destroy
```

参数说明

无

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
[LocalStorageApi destroy];
```

8.2.14.3 获取所有房间列表

接口功能

从本地存储中获取所有的房间列表，已排序

接口原型

```
+ (NSArray<Room *> *)getAllRooms
```

参数说明

无

返回结果

类型	描述
NSArray<Room *>	返回排好序的房间列表

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *rooms = [LocalStorageApi getAllRooms];
```

8.2.14.4 根据房间标识获取单个房间

接口功能

从本地存储中根据roomId获取单个房间

接口原型

```
+ (Room *)getRoomWithRoomId:(NSInteger)roomId
```

参数说明

参数	必选/可选	类型	描述
roomId	必选	NSInteger	房间标识

返回结果

类型	描述
Room	返回房间实体

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
Room *room = [LocalStorageApi getRoomWithRoomId:@(1001)];
```

8.2.14.5 保存排好序的房间标识数组

接口功能

把排好序的房间，取roomId存到本地存储中

接口原型

```
+ (void)saveSortedRoomIds:(NSArray<NSNumber *> *)roomIds
```

参数说明

参数	必选/可选	类型	描述
roomIds	必选	NSArray<NSNumber *>	房间标识数组

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *roomIds = @[@(1001), @(1002), @(1003)];
[LocalStorageApi saveSortedRoomIds:roomIds];
```

8.2.14.6 获取所有设备列表

接口功能

从本地存储中获取所有设备列表，已排序

接口原型

```
+ (NSArray<Device *> *)getAllDevices
```

参数说明

无

返回结果

类型	描述
NSArray<Device *>	返回排好序的设备列表

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *devices = [LocalStorageApi getAllDevices];
```

8.2.14.7 根据设备类型获取设备列表

接口功能

根据设备类型从本地存储中获取设备列表

接口原型

```
+ (NSArray<Device *> *)getDevicesByDeviceTypes:(NSArray<NSString*> *)deviceTypes
```

参数说明

参数	必选/可选	类型	描述
deviceTypes	必选	NSArray<NSString *>	设备类型数组

返回结果

类型	描述
NSArray<Device *>	返回该类设备列表

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *devices = [LocalStorageApi getDevicesByDeviceTypes:@[@”camera”]];
```

8.2.14.8 根据设备标识获取单个设备

接口功能

根据deviceId，从本地存储中获取单个设备

接口原型

```
+ (Device *)getDeviceWithDeviceId:(NSString *)deviceId
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	设备标识

返回结果

类型	描述
Device	返回一个设备实体

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
Device *device = [LocalStorageApi getDeviceWithDeviceId:@”xxxxxxx”];
```

8.2.14.9 保存排好序的设备标识数组

接口功能

把排好序的设备，取deviceId存到本地存储中

接口原型

```
+ (void)saveSortedDeviceIds:(NSArray<NSString *> *)deviceIds
```

参数说明

参数	必选/可选	类型	描述
deviceIds	必选	NSArray<NSString *>	设备标识数组

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *deviceIds = @[@”xxxx1”, @”xxxx2”, @”xxxx3”];
[LocalStorageApi saveSortedDeviceIds:deviceIds];
```

8.2.14.10 判断设备是否在规则中

接口功能

根据设备标识，判断某个设备是否在规则列表中存在

接口原型

```
+ (BOOL)isDeviceExistedInRule:(NSString *)deviceId
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	设备标识

返回结果

类型	描述
BOOL	返回一个布尔值表示是否存在

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
BOOL isExisted = [LocalStorageApi isDeviceExistedInRule:@”xxxx”];
```

8.2.14.11 获取所有规则列表

接口功能

从本地存储中获取所有规则列表，已排序

接口原型

```
+ (NSArray<Rule *> *)getAllRules
```

参数说明

无

返回结果

类型	描述
NSArray< Rule *>	返回排好序的规则列表

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *rules = [LocalStorageApi getAllRules];
```

8.2.14.12 根据规则标识获取单条规则

接口功能

从本地存储中根据ruleId获取单条规则

接口原型

```
+ (Rule *)getRuleWithRuleId:(NSInteger)ruleId
```

参数说明

参数	必选/可选	类型	描述
ruleId	必选	NSInteger	规则标识

返回结果

类型	描述
Rule	返回一个规则实体

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
Rule *rule = [LocalStorageApi getRuleWithRuleId:@(1001)];
```

8.2.14.13 保存排好序的规则标识数组

接口功能

把排好序的规则，取ruleId存到本地存储中

接口原型

```
+ (void)saveSortedRuleIds:(NSArray<NSNumber *>*)ruleIds
```

参数说明

参数	必选/可选	类型	描述
ruleIds	必选	NSArray<NSNumber *>	规则标识数组

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *ruleIds = @[@(1001), @(1002), @(1003)];
[LocalStorageApi saveSortedRuleIds:ruleIds];
```

8.2.14.14 获取所有事件列表

接口功能

从本地存储中获取所有事件列表

接口原型

```
+ (NSArray<Event *> *)getAllEvents
```

参数说明

无

返回结果

类型	描述
NSArray< Event *>	返回所有事件列表

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *events = [LocalStorageApi getAllEvents];
```

8.2.14.15 根据事件标识获取单条事件

接口功能

从本地存储中根据eventId获取单条事件

接口原型

```
+ (Event *)getEventWithEventId:(NSString *)eventId
```

参数说明

参数	必选/可选	类型	描述
eventId	必选	NSString	事件标识

返回结果

类型	描述
Event	返回一个事件实体

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
Event *event = [LocalStorageApi getEventWithEventId:@”xxxx”];
```

8.2.14.16 根据设备标识获取设备事件列表

接口功能

从本地存储中根据deviceId获取设备事件列表

接口原型

```
+ (NSArray<Event *> *)getDeviceEventsWithDeviceId:(NSString*)deviceId
```

参数说明

参数	必选/可选	类型	描述
deviceId	必选	NSString	设备标识

返回结果

类型	描述
NSArray<Event *>	返回该设备下的事件列表

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *deviceEvents = [LocalStorageApi getDeviceEventsWithDeviceId:@”xxxx”];
```

8.2.14.17 获取所有用户列表

接口功能

从本地存储中获取所有的用户列表

接口原型

```
+ (NSArray<User *> *)getAllUsers
```

参数说明

无

返回结果

类型	描述
NSArray< User *>	返回所有的用户列表

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *users = [LocalStorageApi getAllUsers];
```

8.2.14.18 根据用户标识获取单个用户

接口功能

从本地存储中根据userId获取单个用户

接口原型

```
+ (User *)getUserWithUserId:(NSString *)userId
```

参数说明

参数	必选/可选	类型	描述
userId	必选	NSString	用户标识，这里指的是用户的电话号码

返回结果

类型	描述
User	返回一个用户实体

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
User *user = [LocalStorageApi getUserWithUserId:@@”0051900000001”];
```

8.2.14.19 获取所有快捷方式列表

接口功能

从本地存储中获取所有的快捷方式列表

接口原型

```
+ (NSArray<Shortcut *> *)getAllShortcuts
```

参数说明

无

返回结果

类型	描述
NSArray< Shortcut *>	返回所有的快捷方式列表

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *shortcuts = [LocalStorageApi getAllShortcuts];
```

8.2.14.20 设置快捷方式列表

接口功能

把快捷方式列表保存到本地存储中

接口原型

```
+ (void)setShortcuts:(NSArray<Shortcut *> *)shortcuts
```

参数说明

参数	必选/可选	类型	描述
shortcuts	必选	NSArray<Shortcut *>	要保存的快捷方式数组

返回结果

类型	描述
void	无返回值

注意事项

无

调用示例

```
#import “LocalStorageApi.h”
NSArray *shortcuts = @[@”xxxx_deviceId1”, @(xxxx_ruleId1)];
[LocalStorageApi setShortcuts:shortcuts];
```

8.3 常见数据结构定义

该章节主要将SDK提供功能所涉及的数据模型在此呈现出来，方便开发者了解数据模型中具体的属性所代表的意思，当开发者在开发接口或功能时遇到一些数据模型的操作均可以在此找到相对应的结构供参考。

8.3.1 Device

字段	必选/可选	类型	描述
createTime	必选	NSString	设备添加时间, 时间格式 yyyymmddThhmissZ , 例如, 20151212T121212Z .

字段	必选/可选	类型	描述
lastModifiedTime	必选	NSString	设备最后修改时间,时间格式yyyymmddThhmissZ , 例如,20151212T121212Z .
deviceId	必选	NSString	设备标识
gatewayId	必选	NSString	网关标识
nodeType	必选	NSString	节点
deviceInfo	必选	DeviceInfo	设备详情
services	必选	NSArray	设备的服务信息

8.3.2 DeviceInfo

字段	必选/可选	类型	描述
name	可选	NSString	设备名称
deviceDescription	可选	NSString	设备描述
bridgeId	可选	NSString	桥接标识
manufacturerId	必选	NSString	供应商标识
manufacturerName	可选	NSString	供应商名字
mac	可选	NSString	物理地址
location	可选	NSString	设备位置
deviceType	必选	NSString	设备种类： MultiSensor ContactSensor Camera Gateway

字段	必选/可选	类型	描述
model	必选	NSString	设备型号. 如果是Z-Wave协议，型号就是ProductType + ProductId（用零填充，如果需要） 格式的十六进制数,例如 001A-0A12. 如果是其他协议，型号需要另一种格式表示
swVersion	可选	NSString	使用Z-Wave协议时的软件版本号。 其格式是 major version number.minor version number, 例如, 1.1.
fwVersion	可选	NSString	固件版本号
hwVersion	可选	NSString	硬件版本号
sigVersion	可选	NSString	信号版本号
protocolType	必选	NSString	协议类型 (Z-Wave).
signalStrength	可选	NSString	信号强度
nodeId	可选	NSString	节点标识
status	可选	NSString	设备状态. ONLINE: 在线 OFFLINE: 下线 INBOX: 停用.
statusDetail	可选	NSString	设备状态的详细情况
mute	可选	BOOL	设备是否进行屏蔽 TRUE: 被屏蔽. FALSE: 没被屏蔽.
isSecurity	可选	BOOL	是否安全模式
supportedSecurity	可选	BOOL	是否支持安全模式

8.3.3 Service

字段	必选/可选	类型	描述
serviceType	必选	NSString	服务类型
serviceId	必选	NSString	服务ID标识
data	必选	NSString	指定一个属性值对(AVP).
eventTime	必选	NSString	时间格式 yyyymmddThhmissZ , 例如, 20151212T121212Z .
serviceInfo	可选	ServiceInfo	服务详情

8.3.4 ServiceInfo

字段	必选/可选	类型	描述
muteCmds	可选	NSArray	隐私模式屏蔽命令列表

8.3.5 ServiceCommand

字段	必选/可选	类型	描述
commandName	必选	NSString	设备服务命令名称
commandParameters	必选	NSMutableDictionary	设备服务命令参数，需要根据平台的servcie_profile来设置每个命令的具体参数的键值对
serviceId	必选	NSString	设备服务标识

8.3.6 CloudStorageSpace

字段	必选/可选	类型	描述
quotaByte	必选	NSString	总共大小
spaceSizeByte	必选	NSString	使用大小

8.3.7 VideoRecordingInfo

字段	必选/可选	类型	描述
result	必选	VIDEO_RECORD_RESULT	视频录像成功或失败
deviceId	必选	NSString	摄像头设备标识
firstFrameUrl	必选	NSString	视频录像首帧图像URL
availableSize	必选	int	云存储服务器剩余空间大小
duration	必选	int	视频录像时长
frameSize	必选	int	视频录像首帧图像大小
videoUrl	必选	NSString	视频录像URL
totalSize	必选	int	视频录像大小

8.3.8 MediaPlayer

字段	必选/可选	类型	描述
type	必选	MEDIA_PLAYER_TYPE	多媒体播放器类型
view	必选	UIView	视频播放view

8.3.9 Event（所有的事件子类继承自该父类）

字段	必选/可选	类型	描述
eventId	必选	NSString	事件的唯一标示
startTime	必选	NSString	事件上报时间, 时间格式 yyyymmddThhmmssZ , 例如, 20151212T121212Z .
homeId	必选	NSString	智慧家庭Id

字段	必选/可选	类型	描述
triggerId	必选	NSString	触发Id，规则触发为ruleId，手动录像为录像用户的用户名
triggerType	必选	NSString	触发类型，不同的类型对应不同的子类 apiTriggerTypeRule（规则事件） apiTriggerTypeDevice（设备事件） apiTriggerTypeManualRecording（手动录像事件） apiTriggerTypeManualRuleAlteration（创建修改删除规则事件） apiTriggerTypeDmUpgrade（设备网关升级事件） apiTriggerTypeStorage（uds空间告警事件）

8.3.10 RuleEvent(继承自 Event)

字段	必选/可选	类型	描述
name	必选	NSString	规则名称
type	必选	NSInteger	规则类型。取值范围：1为手动触发规则，2为定时规则，3为条件规则
timezoneId	必选	NSString	时区
actions	必选	NSArray<RuleEventAction *>	规则动作列表
conditions	必选	NSArray<RuleEventCondition *>	规则条件列表
timer	必选	RuleEventTimer	规则触发时间

字段	必选/可选	类型	描述
notification	可选	RuleEventNotification	规则通知

8.3.11 RuleEventAction

字段	必选/可选	类型	描述
eventType	必选	NSString	执行动作的事件类型：device upgrade、device deleted、device disconnection、low power、recording、other
result	可选	NSString	录像结果。录像成功”SUCCESS_VIDEO”
deviceId	必选	NSString	设备Id
deviceType	必选	NSString	设备类型
deviceName	必选	NSString	设备名称
actionId	必选	NSString	动作Id。用于录像动作
action	必选	NSDictionary	设备动作
startTime	必选	NSString	事件的触发时间
data	必选	JSONObject	设备执行动作后的上报结果。根据eventType而不同

8.3.12 RuleEventCondition

字段	必选/可选	类型	描述
deviceId	必选	NSString	设备Id
deviceType	必选	NSString	设备类型
deviceName	必选	NSString	设备名称
property	必选	NSString	属性名称
rightValue	必选	NSString	属性值

字段	必选/可选	类型	描述
duration	必选	NSInteger	规则执行延迟时间，单位min，默认为0
operator	必选	NSString	条件操作符，[“>”,”<”,”=”]

8.3.13 RuleEventTimer

字段	必选/可选	类型	描述
startTime	必选	NSString	开始时间点，格式为：“xx：xx”，二十四小时制，以UTC+0为基准
endTime	必选	NSString	结束时间点，格式为：“xx：xx”，二十四小时制，以UTC+0为基准
time	可选	NSString	固定触发时间，格式为：“xx：xx”，二十四小时制，以UTC+0为基准
type	必选	NSString	类型
weekDays	必选	NSArray	星期列表，1代表星期日，2代表星期一，以此类推
regularityLen	可选	NSNumber	周期

8.3.14 RuleEventNotification

字段	必选/可选	类型	描述
smsResult	可选	NSArray<EventNotificationSms *>	短信通知结果列表
emailResult	可选	NSArray<EventNotificationEmail>	邮件通知结果列表

8.3.15 RuleAlterationEvent(继承自 Event)

字段	必选/可选	类型	描述
eventType	必选	NSString	事件类型
userName	必选	NSString	操作的用户名
sceneId	必选	NSString	操作的规则Id
sceneName	必选	NSString	操作的规则名
optName	必选	NSString	操作名称。取值范围：create rule、modify rule、delete rule
optResult	必选	NSString	操作结果

8.3.16 EventNotificationSms

字段	必选/可选	类型	描述
phone	可选	NSString	发短信信息的手机号码
result	可选	NSString	发送信息的内容

8.3.17 EventNotificationEamil

字段	必选/可选	类型	描述
email	可选	NSString	发送email的email地址
result	可选	NSString	发送email的内容

8.3.18 DeviceEvent（继承自 Event）

字段	必选/可选	类型	描述
deviceId	必选	NSString	发生状态变化的设备ID
deviceName	必选	NSString	发生状态变化的设备名

字段	必选/可选	类型	描述
deviceType	必选	NSString	发生状态变化的设备类型
eventType	必选	NSString	事件类型（具体有 apiEventTypeDeviceUpgrade（设备升级） apiEventTypeDeviceDelete（删除设备） apiEventTypeDeviceConnect（设备上线） apiEventTypeDeviceDisconnect（网关断连） apiEventTypeSensorDisconnect（设备断连） apiEventTypeLowpower（低电量） apiEventTypeOther（设备状态变化） apiEventTypePrivacyMode（隐私模式）
data	必选	NSDictionary	低电量时存储的当前电量，其他事件(apiEventTypeOther)存储的当前设备的发生变化的属性值，如门磁开关事件 {@”status”:@”OPEN”}
deviceStartTime	必选	NSString	事件发生的时间，格式同上

8.3.19 ManualRecordingEvent（继承自 Event）

字段	必选/可选	类型	描述
deviceId	必选	NSString	录像的设备ID
deviceName	必选	NSString	录像的设备名

字段	必选/可选	类型	描述
deviceType	必选	NSString	录像的设备类型
recordStartTime	必选	NSString	录像发生的时间
data	必选	RecordResult	录像的结果

8.3.20 StorageEvent（继承自 Event）

字段	必选/可选	类型	描述
eventType	必选	NSString	apiEventTypeUdsLowSpace
usedPercent	必选	NSInteger	已使用的空间百分比
storageStatus	必选	NSInteger	当前存储状态（1、正常；3、满了）

8.3.21 RecordResult

字段	必选/可选	类型	描述
result	必选	NSString	录像结果[SUCCESS_VIDEO,ERROR_VIDEO_RECORDING,ERROR_VIDEO_IPC_ERROR]
duration	必选	NSString	录像时间
videoUrl	必选	NSString	录像视频的url
fileSize	必选	NSString	录像视频的文件大小
firstFrameUrl	必选	NSString	录像视频的首帧图

8.3.22 DmUpgradeEvent（继承自 Event）

字段	必选/可选	类型	描述
deviceId	必选	NSString	设备Id

字段	必选/可选	类型	描述
deviceType	必选	NSString	设备类型
deviceName	必选	NSString	设备名称
eventType	必选	NSString	事件类型：apiEventTypeDeviceUpgrade
upgradeResult	必选	NSString	设备升级的操作结果

8.3.23 Rule

字段	必选/可选	类型	描述
ruleId	必选	NSInteger	规则id
name	必选	NSString	规则名字
ruleDescription	可选	NSString	规则描述
imageIndex	必选	NSInteger	规则图片
type	必选	NSInteger	规则类型1：手动规则，2：定时规则，3：条件规则，4：预定义规则
status	可选	NSString	规则状态，取值为：“active”和“inactive”
actions	可选	NSArray<RuleAction *>	规则动作, 动作类型type取值： 1、“DEVICE_CMD”对应RuleAction

字段	必选/可选	类型	描述
conditions	可选	NSArray	规则条件，条件类型type取值： 1、“DEVICE_DATA”对应RuleDeviceDataCondition 2、“DAILY_TIMER”对应RuleDailyTimerCondition 3、“CYCLE_TIMER”对应RuleCycleTimerCondition
timeRange	可选	RuleTimeRange	定时规则的时间段
notifies	可选	NSArray<RuleNotification *>	规则通知对象列表
matchNow	可选	NSString	是否马上触发：取值 yes | no
timeZoneID	可选	NSString	用户家庭时区ID

8.3.24 RuleAction

字段	必选/可选	类型	描述
type	必选	NSString	动作类型
actionId	可选	NSInteger	录像动作Id
deviceId	必选	NSString	设备id
serviceId	必选	NSString	服务id
serviceType	可选	NSString	服务类型
messageType	必选	NSString	控制命令
messageBody	必选	NSDictionary	命令参数

8.3.25 RuleDeviceDataCondition

字段	必选/可选	类型	描述
type	条件场景必选	NSString	条件类型
deviceId	条件场景必选	NSString	设备id
serviced	条件场景必选	NSString	服务id
serviceType	可选	NSString	服务类型
property	条件场景必选	NSString	传感器属性字段
operator	条件场景必选	NSString	操作符，“>”、“<”、“=”
rightValue	条件场景必选	NSString	传感器属性值
duration	可选	NSInteger	条件满足后规则触发的等待时长，单位为min

8.3.26 RuleDailyTimerCondition

字段	必选/可选	类型	描述
type	必选	NSString	条件类型
time	必选	NSString	固定触发时间，格式为：“xx：xx”，二十四小时制，以UTC+0为基准
daysOfWeek	必选	NSArray<NSNumber *>	星期列表，1代表星期日，2代表星期一，以此类推

8.3.27 RuleCycleTimerCondition

字段	必选/可选	类型	描述
type	必选	NSString	条件类型
timeRange	必选	RuleTimeRange	条件时间
regularityLen	必选	NSString	周期

8.3.28 RuleTimeRange

字段	必选/可选	类型	描述
type	必选	NSString	条件类型
startTime	必选	NSString	开始时间点，格式为：“xx：xx”，二十四小时制，以UTC+0为基准
endTime	必选	NSString	结束时间点，格式同上
daysOfWeek	必选	NSArray<NSNumber *>	星期列表，1代表星期日，2代表星期一，以此类推

8.3.29 RuleNotification

字段	必选/可选	类型	描述
accountId	可选	NSInteger	用户id，需要push通知时必填
phone	可选	NSString	Sms通知的手机号码
email	可选	NSString	邮件通知的邮箱地址
push	可选	NSString	Y代表需要push通知，N代表不必要
title	可选	NSString	预留用户昵称

8.3.30 User

字段	必选/可选	类型	描述
userId	必选	NSInteger	用户Id
name	可选	NSString	用户的名字
account	必选	NSString	用户的账号，这里是电话号码
email	必选	NSString	用户的邮箱

字段	必选/可选	类型	描述
roleType	可选	NSString	用户的角色，“admin”表示子用户，“master”表示主用户
confirmStatus	可选	NSInteger	用户的确认状态，0表示用户初始化状态；1表示用户待确认是否接受邀请状态；2表示用户已接受邀请状态；3代表用户已拒绝邀请状态
lastLoginTime	可选	NSString	用户最后登录时间

8.3.31 ServicePackage

字段	必选/可选	类型	描述
accountId	必选	NSString	用户的账号
name	必选	NSString	套餐名
packageDescription	可选	NSString	套餐描述
services	必选	NSArray	套餐中包含的服务
subscriptionId	可选	NSString	套餐订阅Id
subscriptionStatus	可选	NSString	套餐订阅状态，“activated”已激活，“inactivated”未激活

8.3.32 ServicePackageInfo

字段	必选/可选	类型	描述
quota	可选	NSString	当套餐服务类型是cloudStorage时，用来表示总的存储空间大小
status	必选	NSString	套餐服务的状态
type	必选	NSString	套餐服务的类型

字段	必选/可选	类型	描述
usage	可选	NSString	当套餐服务类型是sms是，用来表示短信服务中短信的总用量

8.3.33 Room

字段	必选/可选	类型	描述
roomId	可选	NSString	房间Id
roomName	必选	NSString	房间名
roomDescription	可选	NSString	房间描述
devices	可选	NSArray<Device *>	房间中的设备

8.3.34 Shortcut

字段	必选/可选	类型	描述
shortcutId	必选	NSString	快捷方式id，可以是deviceId或ruleId
shortcutType	必选	NS_ENUM	整型枚举，取值：ShortcutTypeRule 规则快捷方式 ShortcutTypeDevice 设备快捷方式

8.3.35 NewVersionInfo

字段	必选/可选	类型	描述
fileName	必选	NSString	升级包文件名
version	必选	NSString	新版本号
oldVersion	可选	NSString	旧版本号
date	可选	NSString	日期
size	必选	long	升级包大小
desc	可选	NSString	升级包描述

字段	必选/可选	类型	描述
isForceUpgrade	可选	BOOL	是否强制升级
forceUpgVersion	可选	NSString	强制升级的版本号
urls	必选	NSArray<NSString *>	升级包urls

8.4 常量定义

该章节主要提供了SDK接口和功能中所涉及的常量的解释，如配置项常量，通用错误码，登陆注销常量，用户信息管理常量，设备相关常量，视频相关常量。在开发过程中遇到不理解其含义的常量时，可以在此章节进行查找。

8.4.1 配置项常量

常量名	描述
NSString SYS_CFG_NA_SERVER_IP	配置项：服务器IP
NSString SYS_CFG_NA_SERVER_STANDBY_IP	配置项：服务器备用IP
NSString SYS_CFG_NA_SERVER_PORT	配置项：服务器端口
NSString SYS_CFG_MQTT_SERVER_IP	配置项：推送服务器IP
NSString SYS_CFG_MQTT_SERVER_PORT	配置项：推送服务器端口
NSString SYS_CFG_COUNTRY_CODE	配置项：国家码
NSString SYS_CFG_OPERATION_TIMEOUT_TIME	配置项：操作超时时间

8.4.2 通用错误码

常量名	描述
long ERROR_CODE_SYS_NULL	通用错误码：无错误
long ERROR_CODE_SYS_NETWORK_UNAVAILABLE	通用错误码：网络不可用
long ERROR_CODE_SYS_DEVICE_OFFLINE	通用错误码：设备离线

常量名	描述
long ERROR_CODE_SYS_GATEWAY_OFFLINE	通用错误码：网关离线
long ERROR_CODE_SYS_USER_OFFLINE	通用错误码：用户离线
long ERROR_CODE_SYS_POOR_NETWORK	通用错误码：网络不好
long ERROR_CODE_SYS_INVALID_PARAM	通用错误码：无效参数

8.4.3 登录登出常量

1 登录常量

常量名	描述
long LOGIN_STATUS_CONNECTING	登录状态：连接中
long LOGIN_STATUS_CONNECTED	登录状态：已经连接
long LOGIN_STATUS_DISCONNECTED	登录状态：断开连接
NSString LOGIN_EXTENDING_PARAM_VERIFY_CODE	登录扩展参数：登录验证码
NSString LOGIN_EXTENDING_PARAM_TRUST_DEVICE_VERIFY_CODE	登录扩展参数：可信设备验证码

2 登录错误码

常量名	描述
long ERROR_CODE_LOGIN_AUTH_FAILED	登录错误码：鉴权失败
long ERROR_CODE_LOGIN_ACCOUNT_INACTIVE	登录错误码：账号未激活
long ERROR_CODE_LOGIN_ACCOUNT_LOCKED	登录错误码：账号因多次鉴权失败被锁定

常量名	描述
long ERROR_CODE_LOGIN_INVALID_VERIFY_CODE	登录错误码：验证码错误
long ERROR_CODE_LOGIN_FORCE_LOGOUT	登录错误码：账号被踢
long ERROR_CODE_LOGIN_VERIFY_CODE_EXPIRED	登录错误码：验证码过期
long ERROR_CODE_LOGIN_TRUST_DEVICE_UNREGISTER	登录错误码：在未信任的设备上登录
long ERROR_CODE_LOGIN_TRUST_DEVICE_VERIFY_CODE_ERROR	登录错误码：设备验证码错误
long ERROR_CODE_LOGIN_TRUST_DEVICE_VERIFY_CODE_RETRY_LIMIT	登录错误码：设备验证码错误且重试次数已用完
long ERROR_CODE_LOGIN_TRUST_DEVICE_VERIFY_CODE_EXPIRED	登录错误码：设备验证码过期

3 激活账号错误码

常量名	描述
long ERROR_CODE_ACTIVATE_ACCOUNT_ACTIVATION_CODE_EXPIRED	激活账号错误码：激活码过期
long ERROR_CODE_ACTIVATE_ACCOUNT_ATTEMPTS_TIMES_REACH_MAX	激活账号错误码：激活失败次数达到上限
long ERROR_CODE_ACTIVATE_ACCOUNT_ACTIVATION_CODE_EXISTED	激活账号错误码：激活码已经被使用

4 获取激活码错误码

常量名	描述
long ERROR_CODE_REQUEST_ACTIVATION_CODE_FAILED	获取激活码错误码：获取失败
long ERROR_CODE_ACTIVATE_ACCOUNT_ATTEMPTS_TIMES_REACH_MAX	获取激活码错误码：获取激活码次数达到上限

8.4.4 用户管理常量

1 参数定义

常量名	描述
NSString USER_EXTENDING_PARAM_EMAIL	注册和设置个人信息扩展参数：邮箱
NSString USER_EXTENDING_PARAM_USER_NAME	设置个信息扩展参数：用户名
NSString USER_ EXTENDING_PARAM_USER_ID	个人信息扩展参数：用户ID
NSString USER_ EXTENDING_PARAM_ROLE_TYPE	个人信息扩展参数：用户类型（admin、master）

2 错误码定义

常量名	描述
long ERROR_CODE_USER_EMAIL_OCCUPIED	注册错误码：邮箱被占用
long ERROR_CODE_USER_ACCOUNT_NAME_OCCUPIED	注册错误码：用户名被占用
long ERROR_CODE_USER_ACCOUNT_NAME_AND_EMAIL_OCCUPIED	注册错误码：用户名和邮箱被占用
long ERROR_CODE_USER_PASSWORD_SAME_AS_ACCOUNT_NAME	注册错误码：密码与账号一样

常量名	描述
long ERROR_CODE_USER_PASSWORD_SAME_AS_ACCOUNT_NAME_REVERSE	注册错误码：密码与账号倒序一样
long ERROR_CODE_USER_PASSWORD_SAME_AS_USER_NAME	注册错误码：密码与用户名一样
long ERROR_CODE_USER_PASSWORD_SAME_AS_USER_NAME_REVERSE	注册错误码：密码与用户名倒序一样
long ERROR_CODE_USER_PASSWORD_SAME_AS_EMAIL	注册错误码：密码与邮箱一样
long ERROR_CODE_USER_PASSWORD_SAME_AS_EMAIL_REVERSE	注册错误码：密码与邮箱倒序一样
long ERROR_CODE_USER_PASSWORD_TOO_SHORT	注册错误码：密码太短。要：8--64位
long ERROR_CODE_USER_PASSWORD_TOO_LONG	注册错误码：密码太长。要：8--64位
long ERROR_CODE_USER_PASSWORD_FORMAT_INVALID	注册错误码：密码格式不对。要求：必须包括大、小写字母和数字
long ERROR_CODE_USER_RESET_PASSWORD_INVALID_VERIFY_CODE	重设密码错误码：重设密码验证码不对
long ERROR_CODE_USER_RESET_PASSWORD_ERROR_VERIFY_CODE_EXPIRED	重设密码错误码：重设密码验证码过期
long ERROR_CODE_USER_RESET_PASSWORD_PASSWORD_MATCHES_OLD	修改密码、重设密码错误码：重设密码新老密码一样
long ERROR_CODE_MODIFY_PASSWORD_OLD_PASSWORD_INCORRECT	修改密码错误码：旧密码不正确
long ERROR_CODE_INVITE_USER_EXISTED_IN_OTHER_HOME	邀请子用户错误码：被邀请的用户已经在其它家庭中

常量名	描述
long ERROR_CODE_INVITE_USER_CAN_NOT_BE_INVITED	邀请子用户错误码：该用户不能被邀请
long ERROR_CODE_INVITE_USER_NUMBER_REACH_MAX	邀请子用户错误码：被邀请的用户数量达到最大值
long ERROR_CODE_INVITE_USER_CAN_NOT_INVITE_SELF	邀请子用户错误码：不能邀请自己

8.4.5 设备常量

1 设备状态定义

常量名	描述
NSString DEVICE_STATUS_ONLINE	设备在线状态：在线
NSString DEVICE_STATUS_OFFLINE	设备在线状态：离线
NSString DEVICE_STATUS_DETAIL_NONE	设备升级状态：没有在升级
NSString DEVICE_STATUS_DETAIL_UPDATING	设备升级状态：正在升级
NSString DEVICE_STATUS_DETAIL_NOT_ACTIVE	设备状态：设备未被激活

2 设备广播及参数定义

常量名	描述
NSString EVENT_DEVICE_UPGRADE_STATUS_CHANGED	设备升级状态变化广播
NSString EVENT_DEVICE_ADDED	设备添加成功广播
NSString EVENT_DEVICE_DELETED	设备删除成功广播
NSString EVENT_DEVICE_INFO_CHANGED	设备信息变化广播

常量名	描述
NSString EVENT_DEVICE_ROOM_INFO_CHANGED	设备所属房间的信息发生变化广播
NSString EVENT_SERVICE_INFO_CHANGED	隐私模式广播
NSString PARAM_UPGRADE_STATUS	设备升级状态广播带的参数： DEVICE_STATUS_DETAIL_NONE、 DEVICE_STATUS_DETAIL_UPDATING
NSString PARAM_DEVICE_ID	设备删除广播带的参数： deviceId设备标识
NSString PARAM_DEVICE	设备升级、设备添加、设备信息变化广播带的参数： Device *device
NSString PARAM_DEVICE_ROOM_INFO	设备所属房间信息变化广播带的参数：NSDictionary
NSString PARAM_SERVICE_INFO	隐私模式广播带的参数：ServiceInfo
NSString EVENT_VIDEO_RECORDING_DONE	视频录制完成广播
NSString PARAM_VIDEO_RECORDING_INFO	视频录制完成广播带的参数：VideoRecordingInfo

3 设备错误码定义

常量名	描述
long ERROR_CODE_DEVICE_USER_NOT_FOUND	错误码：查询不到用户
long ERROR_CODE_DEVICE_DEVICE_NOT_FOUND	错误码：查询不到设备
long ERROR_CODE_DEVICE_NO_DEVICE_ACCESS	错误码：用户对设备无权限
long ERROR_CODE_DEVICE_DEVICE_ALREADY_ACTIVED	绑定网关错误码：设备已经被激活

常量名	描述
long ERROR_CODE_DEVICE_RENAME_NO_ACCESS	设备重命名错误码：用户对设备无权限
long ERROR_CODE_DEVICE_RENAME_DEVICE_INACTIVATED	设备重命名错误码：设备未被激活
long ERROR_CODE_DEVICE_RENAME_NAME_ALREADY_EXIST	设备重命名错误码：设备名已存在
long VIDEO_RECORD_SUCCESS	视频录制成功
long VIDEO_RECORD_FAILED	视频录制失败
long ERROR_CODE_DEVICE_QUERY_CLOUD_STORAGE_FAILED	查询云储存空间错误码：查询失败
long ERROR_CODE_DEVICE_CLOUD_STORAGE_NO_ENOUGH_SPACE	云存储空间不足
long ERROR_CODE_DEVICE_VIDEO_RECORD_ALREADY_DOWNLOADED	视频下载错误码：视频已经下载
long ERROR_CODE_DEVICE_VIDEO_RECORD_IS_DOWNLOADING	视频下载错误码：视频正在下载
long ERROR_CODE_DEVICE_VIDEO_RECORD_DOWNLOAD_ERROR_FROM_SERVER	视频下载错误码：视频下载失败
long ERROR_CODE_DEVICE_VIDEO_RECORD_NOT_EXIST	视频下载错误码：下载的视频不存在

8.4.6 视频常量

常量名	描述
MEDIA_PLAYER_TYPE_LIVE	实时视频媒体播放器类型
MEDIA_PLAYER_TYPE_VOD	历史视频媒体播放器类型
EVENT_MEDIA_LIVE_PLAYER_START_PLAY	实时视频开始播放广播

常量名	描述
EVENT_MEDIA_LIVE_PLAYER_PLAY_ERROR	实时视频播放发生错误广播
EVENT_MEDIA_LIVE_PLAYER_CAMERA_BUSY	实时视频播放失败时，摄像头处于忙碌状态广播
EVENT_MEDIA_LIVE_PLAYER_NO_MEDIA	实时视频播放失败时，没有播放内容太广播
EVENT_LIVE_PLAYER_PLAY_ERROR	实时视频播放失败广播
EVENT_MEDIA_VOD_PLAYER_READY_TO_PLAY	历史视频准备播放广播
EVENT_MEDIA_VOD_PLAYER_DID_STOP	历史视频已经停止广播
EVENT_MEDIA_VOD_PLAYER_DID_FAILED	历史视频播放失败广播
EVENT_MEDIA_VOD_PLAYER_DID_PAUSE	历史视频播放暂停广播
EVENT_MEDIA_VOD_PLAYER_DID_PLAY	历史视频已经播放广播
EVENT_MEDIA_VOD_PLAYER_POSITION_CHANGED	历史视频播放位置调整广播
EVENT_MEDIA_VOD_PLAYER_DURATION_AVAILABLE	历史视频时长已经可以获取广播
EVENT_MEDIA_VOD_PLAYER_CACHE_UPDATE	历史视频缓存更新广播
PARAM_MEDIA_VOD_PLAYER_CURRENT_PLAY_TIME	历史视频播放位置调整广播带的参数：float类型，表示当前播放位置
PARAM_MEDIA_VOD_PLAYER_DURATION	历史视频时长已经可以获取广播带的参数：NSTimeInterval类型，表示历史视频时长
PARAM_MEDIA_VOD_PLAYER_ERROR_CODE	历史视频播放失败广播带的参数：long类型错误码
PARAM_MEDIA_VOD_PLAYER_CACHE_PROCESS	历史视频缓存更新广播带的参数：float类型，表示缓存进度

8.4.7 规则常量

1 规则常量

常量名	描述
NSString EVENT_RULE_CREATED	规则创建广播
NSString EVENT_RULE_DELETED	规则删除广播
NSString EVENT_RULE_INFO_CHANGED	规则变更广播
NSString PARAM_RULE	规则广播参数

2 规则错误码

常量名	描述
long ERROR_CODE_RULE_TYPE_INVALID	规则错误码：规则类型无效
long ERROR_CODE_RULE_NOT_FOUND	规则错误码：规则查找不到
long ERROR_CODE_RULE_NAME_EXISTED	规则错误码：规则名已存在
long ERROR_CODE_RULE_STATUS_CHANGE_FAILED	规则错误码：规则状态变化失败
long ERROR_CODE_RULE_INVALID_ACTION	规则错误码：动作无效
long ERROR_CODE_RULE_NULL_ACTIONS	规则错误码：动作为空
long ERROR_CODE_RULE_INVALID_CONDITION	规则错误码：条件无效
long ERROR_CODE_RULE_NULL_CONDITIONS	规则错误码：条件为空
long ERROR_CODE_RULE_INVALID_TIMER	规则错误码：触发时间无效
long ERROR_CODE_RULE_NULL_TIMER	规则错误码：触发时间为空

常量名	描述
long ERROR_CODE_RULE_EXCEED_MAXINUM	规则错误码：规则数达到最大值
long ERROR_CODE_RULE_INVALID_NOTIFY	规则错误码：规则通知无效
long ERROR_CODE_RULE_ACTIVE_FAILED	规则错误码：激活失败
long ERROR_CODE_RULE_TIMER_CONFLICT	规则错误码：规则触发时间与其它规则有冲突
long ERROR_CODE_RULE_INVALID_TIME_RANGE	规则错误码：规则触发时间段无效
long ERROR_CODE_RULE_INVALID_TRIGGER_SOURCE	规则错误码：无效的触发源
long ERROR_CODE_RULE_ACTION_OVER_MAX	规则错误码：规则的动作数超过最大值
long ERROR_CODE_RULE_INVALID_PARAMS	规则错误码：无效的参数

8.4.8 房间管理常量

1 房间管理常量

常量名	描述
NSString EVENT_ROOM_CREATED	房间创建广播
NSString EVENT_ROOM_DELETED	房间删除广播
NSString EVENT_ROOM_UPDATED	房间更新广播
NSString PARAM_ROOM	房间广播参数
NSString PARAM_ROOM_ID	房间广播参数

2 房间管理错误码

常量名	描述
long ERROR_CODE_ROOM_ALREADY_EXISTED	房间错误码：房间名已经存在
long ERROR_CODE_ROOM_NOT_FOUND	房间错误码：找不到房间
long ERROR_CODE_ROOM_DEFAULT_FORBIDEN	房间错误码：不能操作默认房间，不能使用默认房间名

9 Platform API 参考(北向)

9.1 接口列表

Application侧接口除了鉴权接口Auth，其他接口调用都需要在request header中携带参数app_key和Authorization:Bearer {accessToken}。Authorization中{accessToken}的值即为调用Auth接口获取到的accessToken。

应用开发语言若是JAVA请使用JDK1.8。

北向提供的API 是平台与Application之间的https接口， 请使用安全传输协议TLS 1.1/1.2。

9.1.1 应用安全接入

Application携带在IoT Platform（下文简称平台）产生的appId和secret过来，调用鉴权接口，获取鉴权token。请参考本文档 2.1.1鉴权章节进行开发。

9.1.1.1 Auth(鉴权)

接口功能

实现第三方系统在访问开放API之前的认证。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/sec/v1.1.0/login
```

注意事项

鉴权接口是调用其他API的前提，北向接口除了鉴权接口（Auth），其他接口调用都需要在request header中携带参数app_key和Authorization:Bearer {accessToken}。app_key为

参数中的appId，Authorization中{accessToken}的值即为调用Auth接口获取到的accessToken。

如果多次获取令牌，则之前的令牌失效, 最后一次获取的令牌才有效。请勿并发获取令牌。

参数说明

参数	必选/可选	类型	描述
appId	必选	String	用户名，填写应用程序ID
secret	必选	String	登录用户口令，填写应用程序密码

返回结果

字段	类型	描述
scope	String	范围，默认值default
tokenType	String	鉴权token类型，默认值bearer
expiresIn	Integer	平台生成并返回accessToken的有效时间，单位秒
accessToken	String	Oauth 2.0 鉴权参数
refreshToken	String	Oauth 2.0 鉴权参数，用来刷新accessToken。（1个月的有效期）

消息事例

Method：

```
POST
```

request：(非JSON格式)

```
https://server:port/iocm/app/sec/v1.1.0/login
Content-Type:application/x-www-form-urlencoded
Body:
appId={appId}&secret={secret}
```

response：

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
 "scope":"default",
 "tokenType":"bearer",
 "expiresIn":"*******",
 "accessToken":"*******",
 "refreshToken":"*******"
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100208	AppId or secret is not right. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	appId或secret错误

9.1.1.2 Refresh Token(刷新 token)

接口功能

accessToken快过期时，第三方系统通过调用此接口，重新获取可用token。accessToken有效时间参照2.1.1鉴权接口返回字段expiresIn的值。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/sec/v1.1.0/refreshToken
```

注意事项

Body参数说明

参数	必选/可选	类型	描述
appId	必选	String	用户名，填写应用程序ID
secret	必选	String	登录用户口令，填写应用程序密码
refreshToken	必选	String	刷新令牌，用于获取一个新的accessToken

返回结果

字段	类型	描述
scope	String	范围
tokenType	String	鉴权token类型，默认值bearer
expiresIn	String	平台生成并返回accessToken的有效时间，单位秒
accessToken	String	Oauth 2.0 鉴权参数
refreshToken	String	刷新令牌，用于获取一个新的accessToken

消息事例

Method：

```
POST
```

request：

```
https://server:port/iocm/app/sec/v1.1.0/refreshToken
Content-Type:application/json
Body:
{
 "appId"="******",
 "secret"="******",
 "refreshToken":"******"
}
```

response：

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
 "accessToken":"*******",
 "tokenType":"*******",
 "expiresIn":"*******",
 "scope":" *******" ,
 "refreshToken":"*******"
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效
401	100006	Refresh access token failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	refreshToken无效

Http Status Code	error_code	error_desc	说明
401	100208	AppId or secret is not right. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	appId或secret错误

9.1.1.3 注销

接口功能

NA注销鉴权信息。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/sec/v1.1.0/logout
```

注意事项

无

参数说明

参数	必选/可选	类型	描述
accessToken	必选	String(256)	调用鉴权接口获取到的Oauth 2.0鉴权参数

返回结果

Status Code: 204 No Content

消息事例

Method:

```
POST
```

Request:

```
https://server:port/iocm/app/sec/v1.1.0/logout
Header:
Content-Type:application/json
Body:
{
    "accessToken": "*******"
}
```

Response:

```
Status Code: 204 No Content
Content-Type: application/json
```

异常返回码

Http Status Code	error_code	error_desc	说明
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效

9.1.2 设备管理

设备管理提供了Application（下文简称App）申请设备的增，删，改，查以及修改设备基本信息的接口。

App向IoT Platform申请新的设备, IoT Platform分配对应的设备验证码，待设备携带验证码请求接入平台后，分配其deviceId和 secret，允许其使用。平台提供了增，删，改，查接口，实现对新增设备接入灵活操作。对设备的管理请参考本文档进行开发。

9.1.2.1 注册直连设备

接口功能

应用程序添加设备，获取设备的验证码，并在设备访问南向接口时携带验证码，获取设备唯一标识和密码。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/reg/v1.2.0/devices?appId={appId}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

备注：app_key为参数中的appId，Authorization中{accessToken}的值即为调用Auth接口获取到的accessToken。

参数说明

字段	必选/可选	类型	描述
appId	可选	String	应用唯一标识

字段	必选/可选	类型	描述
verifyCode	可选	String	客户端给出verifyCode则返回的就是这个verifyCode，即使用客户端给出的verifyCode。如果不指定，系统自动生成。
nodeId	必选	String	设备唯一标识,如：MAC或SIM卡号或设备esn号等。与设备对接时，必须与设备上报的nodeId一致。 备注：nodeId和verifyCode需要填写相同的值，如果南向连接SoftRadio的模拟环境，该值可自行定义，格式为TESTS_XXXXXX。如果是现网环境，该值通常为设备的IMEI号。
endUserId	可选	String	终端用户Id，如手机号码，email地址
psk	可选	String	psk码，用于生成设备鉴权参数；如不传入，系统自动生成
timeout	可选	Integer	单位秒，不填使用默认值(180s),填写0则永不过期，非0表示在指定时间内设备进行绑定，超过时间验证码无效

返回结果

字段	类型	描述
deviceId	String	设备唯一标识，1-64个字节
verifyCode	String	申请的临时验证码，设备可以通过验证码获取id和密码
timeout	Integer	验证码有效时间，单位秒，设备需要在有效时间内接入平台
psk	String	psk码，用于生成设备鉴权参数

消息事例

Method：

```
POST
```

request：

```
https: //server:port/iocm/app/reg/v1.2.0/devices?appId={appId}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json
Body:
{
     "verifyCode":"AE10-12424-12414",
     "nodeId":"AE10-12424-12414",
     "timeout":300
}
```

response：

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
     "deviceId":"*******",
     "verifyCode":"*******",
     "timeout":300,
     "psk":"******"
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效
401	100025	AppId for auth not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	获取不到appId鉴权信息
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在

Http Status Code	error_code	error_desc	说明
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未被授权
400	100007	Bad request message. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	参数不合法
200	100412	The amount of device has reached the limit.	当前应用下设备数量达到上限
400	100003	Invalid verify code. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	验证码无效
200	100416	The device has already been binded. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备已经绑定
500	100001	Internal server error. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	服务内部处理错误

Http Status Code	error_code	error_desc	说明
200	103026	The license is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	License不存在
200	103028	The license pool resources. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	License无资源
200	103027	The license sales is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	License的销售项不存在

9.1.2.2 发现非直连设备

接口功能

Application发送DISCOVERY命令给网关下的设备。一般用于通过网关添加传感器，若没有网关，该接口请勿使用。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/signaltrans/v1.1.0/devices/{deviceId}/services/{serviceId}/sendCommand?appId={appId}
```

注意事项

```
http携带头域信息：
Header：
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

URL参数说明：

字段	必选/可选	类型	描述
deviceId	必选	String	网关设备唯一标识，1-64个字节
serviceId	必选	String	取值"Discovery"

http消息体说明：

字段	必选/可选	类型	描述
appId	可选	String	应用唯一标识
message	必选	CommandDTONA2Cloud	CommandDTONA2Cloud消息体说明参见下表

CommandDTONA2Cloud消息体说明：

字段	必选/可选	类型	描述
header	必选	CommandNA2CloudHeader	见附录**9.2.9 CommandNA2CloudHeader结构体说明**字段说明
body	可选	ObjectNode	取值长度无要求，消息的消息体

返回结果：

字段	类型	描述
status	String(128)	命令状态：sent --- 已发送、delivered --- 已送达 executed --- 已执行
timestamp	String(128)	发送命令的时间戳，1-32字节 时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z
requestId	String(128)	1-32个字节，平台分配的序列号，标识一个命令 需要requestId关联对应命令执行结果

消息事例

Method：

```
POST
```

request：

```
https://server:port/iocm/app/signaltrans/v1.1.0/devices/{deviceId}/services/Discovery/sendCommand?appId={appId}
```

```
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;

Body:
{
  "header": {
    "mode": "ACK",
    "from": "************",
    "to": null,
    "toType": null,
    "method": "DISCOVERY",
    "requestId": "****************************************",
    "callbackURL": "https://server:port/na/iocm/message/confirm"
  },
  "body": {
      "from": "************",
      "sessionID": "**********",
      "sdp": "**********"
  }
}
```

response：

```
Status Code: 202 Accepted
Content-Type: application/json
Body:
{
    "requestId":"****************************************",
    "status":"sent",
    "timestamp":"1446202014815"
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效

Http Status Code	error_code	error_desc	说明
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未授权
200	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在
500	100023	The data in dataBase is abnomal. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	数据库中数据异常
400	102203	CommandName is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	命令名称无效

Http Status Code	error_code	error_desc	说明
200	100431	The serviceType is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	服务类型不存在
200	100428	The device is not online. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不在线
200	100432	The device command is muted. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备已被muted

9.1.2.3 查询设备激活状态

接口功能

给定网关设备的设备唯一标识查询网关设备激活状态。

调用方法

```
GET
```

接口路径

```
https://server:port/iocm/app/reg/v1.1.0/devices/{deviceId}?appId={appId}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
appId	可选	String	应用唯一标识

返回结果

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
activated	必选	boolean	是否通过验证码获取密码的状态标识 true:已激活;false:未激活
name	必选	String	设备名称

消息事例

Method：

```
GET
```

request：

```
https://server:port/iocm/app/reg/v1.1.0/devices/{deviceId}?appId={appId}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

response：

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
"deviceId":*******",
"activated":*******,
"name":"***"
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息

Http Status Code	error_code	error_desc	说明
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用没有权限
200	100403	The device is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不存在

9.1.2.4 删除直连设备

接口功能

用来删除直连设备.

调用方法

```
DELETE
```

接口路径

```
https://server:port/iocm/app/dm/v1.1.0/devices/{deviceId}?appId={appId}&cascade={cascade}
```

注意事项

```
携带头域信息:
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

本接口只能用于删除直连设备；

参数说明

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
appId	可选	String	应用唯一标识
cascade	可选	Boolean	仅在设备是网关，且连接了传感器时有效。取值null或者true删除网关及其下属设备；取值false时删除网关，并将下属传感器属性变为网关属性

返回结果

Status Code: 204 No Content.

消息事例

Method:

```
DELETE
```

Request:

```
https://server:port/iocm/app/dm/v1.1.0/devices/{deviceId}?appId={appId}&cascade={cascade}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

Response:

```
Status Code: 204 No Content
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在

Http Status Code	error_code	error_desc	说明
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未授权
200	100403	The device is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不存在
200	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在
500	100023	The data in dataBase is abnomal. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	数据中数据异常
400	102203	CommandName is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	命令名称无效

Http Status Code	error_code	error_desc	说明
200	100431	The serviceType is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	服务类型不存在
200	100428	The device is not online. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不在线
200	100432	The device command is muted. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备命令已经muted

9.1.2.5 删除非直连设备

接口功能

Application发送REMOVE命令给网关下的设备。用于通过网关删除传感器，若没有网关，该接口请勿使用。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/signaltrans/v1.1.0/devices/{deviceId}/services/{serviceId}/sendCommand?appId={appId}
```

注意事项

```
http携带头域信息：
Header：
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

http消息体说明：

字段	必选/可选	类型	描述
appId	可选	String	应用唯一标识
message	必选	CommandDTONA2Cloud	NA发送给设备的命令内容

CommandDTONA2Cloud消息体说明：

字段	必选/可选	类型	描述
header	必选	CommandNA2CloudHeader	见**9.2.9 CommandNA2CloudHeader结构体说明**字段说明
body	可选	ObjectNode	取值长度无要求，消息的消息体

返回结果：

字段	类型	描述
status	String	命令状态：1-32字节 sent --- 已发送 delivered --- 已送达 executed --- 已执行
timestamp	String	发送命令的时间戳，1-32字节 时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z
requestId	String	1-32个字节，平台分配的序列号，标识一个命令 需要requestId关联对应命令执行结果

消息事例

Method：

```
POST
```

request：

```
https://server:port/iocm/app/signaltrans/v1.1.0/devices/{deviceId}/services/Remover/sendCommand?
appId={appId}

Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;

Body:
{
```

```
  "header": {
    "mode": "ACK",
    "from": "************",
    "to": null,
    "toType": null,
    "method": "REMOVE",
    "requestId": "****************************************",
    "callbackURL": "https://server:port/callbackUri"
  },
  "body": {
    "timeout": 90,
    "deviceId": "********-0fda-4a56-bed6-4d9019891493",
    "mode": "ON"
  }
}
```

response：

```
Status Code: 202 Accepted
Content-Type: application/json
Body:
{
    "requestId":"****************************************",
    "status":"sent",
    "timestamp":"1446202014815"
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在

Http Status Code	error_code	error_desc	说明
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未授权
200	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在
500	100023	The data in dataBase is abnomal. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	数据库中数据异常
400	102203	CommandName is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	命令名称无效
200	100431	The serviceType is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	服务类型不存在

Http Status Code	error_code	error_desc	说明
200	100428	The device is not online. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不在线
200	100432	The device command is muted. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备已被muted

9.1.2.6 修改设备信息

接口功能

修改设备信息的接口，目前用于设备改名，预留修改设备信息的能力。

调用方法

```
PUT
```

接口路径

```
https://server:port/iocm/app/dm/v1.2.0/devices/{deviceId}?appId={appId}
```

注意事项

```
携带头域信息：
Header：
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
appId	可选	String	应用唯一标识
request	可选	UpdateDeviceInfoReqDTO	自定义结构体（见下表）

UpdateDeviceInfoReqDTO结构体

字段	必选/可选	类型	描述
name	可选	String	设备名称
endUser	可选	String	终端用户，若设备为网关，则endUser可选；若设备是通过网关接入的，则endUser必须为null。
mute	可选	Enum	表示设备是否处于冻结状态，即是否上报数据（处于冻结状态，则不上报数据），取值有“TRUE”,“FALSE”
manufacturerId	可选	String	厂商ID
manufacturerName	可选	String	厂商名
deviceType	可选	String	设备类型: 大驼峰命名方式 MultiSensor ContactSensor Camera Gateway
model	可选	String	型号 z-wave: ProductType + ProductId 16 进制： XXXX-XXXX 补0对齐 如：001A-0A12 其他协议再定
protocolType	可选	String	设备协议类型
deviceConfig	可选	DeviceConfigDTO	设备配置信息，自定义结构体（见下表）
region	可选	String	设备所在地区
location	可选	String	设备的位置
organization	可选	String	设备所属组织
timezone	可选	String	设备所在时区

DeviceConfigDTO

字段	必选/可选	类型	描述
dataConfig	可选	DataConfigDTO	数据配置信息，自定义结构体（见下表）

DataConfigDTO

字段	必选/可选	类型	描述
dataAgingTime	可选	Integer[0,90]	数据老化时长配置，取值：090，单位：天

返回结果

```
Status Code: 204 No Content
```

消息事例

Method：

```
PUT
```

request：

```
https://server:port/iocm/app/dm/v1.1.0/devices/{deviceId}?appId={appId}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
Body:
{
"name":"****",
"endUser":"****"
}
```

response

```
Status Code: 204 No Content
```

9.1.2.7 刷新设备密钥

接口功能

支持刷新注册到本应用的设备和授予权限的其它应用的设备

密钥刷新后，deviceId不变，nodeId值发生变化，设备需要重新绑定。

调用方法

```
PUT
```

接口路径

```
https://server:port/iocm/app/reg/v1.1.0/devices/{deviceId}?appId={appId}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
appId	可选	String	query	授权应用Id
deviceId	必选	String(256)	path	设备Id
verifyCode	可选	String(256)	body	验证码：null 平台生成验证码，并返回；非null 使用，且返回该验证码。（大小写不敏感）
nodeId	可选	String(256)	body	null:nodeId不变；非null：更新；网关的唯一ID，平台用于判重
timeout	可选	Integer	body	验证码超时时间，单位秒。null：默认值(180s)；0：永不过期；非0：指定时间

返回结果

字段	类型	描述
verifyCode	String(256）	申请的临时验证码，网关可以通过验证码获取id和密码
timeout	Integer	验证码有效时间，单位秒，网关需要在有效时间内接入平台

消息事例

Method:

```
PUT
```

Request:

```
https://server:port/iocm/app/reg/v1.1.0/devices/{deviceId}?appId={appId}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;

Body:
{
    "verifyCode": "AE10-12424-12414",
    "nodeId": "rewqrewqrewqre",
    "timeout": 300
}
```

Response:

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
    "verifyCode": "AE10-12424-12414",
    "timeout": 300
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效
401	100025	AppId for auth not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用ID不存在
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未被授权

Http Status Code	error_code	error_desc	说明
400	100007	Bad request message. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	参数不合法
200	100403	The device is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不存在
200	100610	Device is not active. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备未激活
200	100611	Device is online. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备在线
200	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备信息不存在

Http Status Code	error_code	error_desc	说明
200	100426	The nodeId is duplicated. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	nodeId重复
400	100003	Invalid verify code. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	验证码无效

9.1.2.8 设置加密

接口功能

NA调用此接口将车机的加密信息发送给平台

调用方法

```
PUT
```

接口路径

```
https://server:port/iocm/app/dm/v1.1.0/devices/{deviceId}/services/{serviceType}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
deviceId	必选	String	path	设备id
serviceType	必选	String	path	设备服务类型，可选值carEncrypt，carInfo，vehicleProperty

参数	必选/可选	类型	位置	描述
data	可选	JSONObject	body	服务数据： serviceType取值carEncrypt时，data结构体参考**3.10ApplicationSetEncrtptDTO结构体说明**； serviceType取值carInfo、vehicleProperty时，data结构体参考**9.2.11 PutCarInfoData结构体说明**

返回结果

设置加密成功：Status Code: 200 OK

服务类型找不到：Status Code: 400 Not Found

消息事例

Method:

```
PUT
```

Request:

```
https://server:port/iocm/app/dm/v1.1.0/devices/{deviceId}/services/{serviceType}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;

Body:
{
  "data":{
         *****
  }
}
```

Response:

```
Status Code: 200 OK
Content-Type: application/json
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息

Http Status Code	error_code	error_desc	说明
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
200	100403	The device is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不存在
400	100019	Illegal request. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	非法请求，request为空
400	103008	The encrtpt iv is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	加密iv值不存在

Http Status Code	error_code	error_desc	说明
400	103009	The encrtpt key is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	加密key不存在
400	103007	The encrtpt flag is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	加密flag不存在
200	103004	Add new car info, simNumber or carModel cannot be null. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	新的carInfo, simNumber或者carModel不能为null
200	103001	The simNumber is already exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	simNumber已经存在
200	103002	The plateNumber is already exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	plateNumber已经存在

Http Status Code	error_code	error_desc	说明
200	103010	The vin is already exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	vin已经存在
404	200001	The resource is not found. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	资源不存在

9.1.3 数据采集

平台支持对网关事件的订阅，同时还支持查看传感器上报的历史数据，能按时、天、月等维度查看传感器上报的历史数据。数据采集实现了对网关数据收集。

9.1.3.1 按条件批量查询设备信息列表

接口功能

按条件批量查询设备信息列表，如查询指定网关下的所有设备信息列表。

调用方法

```
GET
```

接口路径

```
https://server:port/iocm/app/dm/v1.3.0/devices?
appId={appId}&gatewayId={gatewayId}&nodeType={nodeType}&deviceType={deviceType}&protocolType={proto
colType}&pageNo={pageNo}&pageSize={pageSize}&startTime={startTime}&endTime={endTime}&status={status
}&sort={sort}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	描述
appId	可选	String	应用唯一标识

字段	必选/可选	类型	描述
gatewayId	条件可选	String	设备唯一标识（含网关的设备唯一标识）。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。 gatewayId和pageNo不能同时为null，二者必选其一。
nodeType	可选	String	指定节点的类：ENDPOINT: 传感器. GATEWAY: 网关.
deviceType	可选	String	设备类型
protocolType	可选	String	协议类型（v1.3.0url携带）
pageNo	条件可选	Integer	分页查询参数， pageNo=null时查询内容不分页；取值大于等于0的整数时分页查询,等于0时查询第一页。 gatewayId和pageNo不能同时为null，二者必选其一。
pageSize	可选	Integer	分页查询参数，取值大于等于1的整数，缺省：1
status	可选	String	设备状态,取值:ONLINE在线;OFFLINE 不在线;ABNORMAL异常状态
startTime	可选	String	查询注册设备信息时间在startTime之后的记录。 时间格式：yyyyMMdd’T’HHmmss’Z’ 如： 20151212T121212Z
endTime	可选	String	查询注册设备信息时间在endTime之前的记录。 时间格式：yyyyMMdd’T’HHmmss’Z’ 如： 20151212T121212Z
sort	可选	String	指定返回记录的排序，可取值ASC按注册设备的时间升序排列； DESC按注册设备的时间降序排列 缺省：DESC

注：gatewayId和pageNo不能同时为空。

结果说明

字段	类型	描述
totalCount	Long	查询的记录数量
pageNo	Long	返回的页码序号(大于等于0的整数,0代表第一页)

字段	类型	描述
pageSize	Long	页码大小,当不分页查询时,取值等于totalCount
devices	List<GetDeviceRspDTO>	设备分页列表信息，自定义结构，**9.2.8 GetDeviceRspDTO 结构体说明**

消息事例

Method：

```
GET
```

request：

```
https://server:port/iocm/app/dm/v1.3.0/devices?
appId={appId}&gatewayId={gatewayId}&nodeType={nodeType}&deviceType={deviceType}&protocolType={proto
colType}&pageNo={pageNo}&pageSize={pageSize}&startTime={startTime}&endTime={endTime}&status={status
}&sort={sort}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

response：

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
    "totalCount":1,
    "pageNo":0,
    "pageSize":10,
    "devices":[
        {
            "deviceId":"************************",
            "gatewayId":"************************",
            "nodeType":"GATEWAY",
            "createTime":"20170426T022604Z",
            "lastModifiedTime":"20170426T022604Z",
            "deviceInfo":{
                "nodeId":null,
                "name":null,
                "description":null,
                "manufacturerId":null,
                "manufacturerName":null,
                "mac":null,
                "location":null,
                "deviceType":null,
                "model":null,
                "swVersion":null,
                "fwVersion":null,
                "hwVersion":null,
                "protocolType":null,
                "bridgeId":null,
                "status":"OFFLINE",
                "statusDetail":"NOT_ACTIVE",
                "mute":null,
                "supportedSecurity":null,
                "isSecurity":null,
                "signalStrength":null,
                "sigVersion":null,
                "serialNumber":null,
```

```
                "batteryLevel":null
            },
            "services":null,
            "connectionInfo":{
                "protocolType":null
            },
            "location":null,
            "devGroupIds":[]
        }
    ]
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100216	The application input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用输入无效
200	100218	The gatewayId and pageNo can't be both null. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	网关id和pageNo不能同时为null
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在

Http Status Code	error_code	error_desc	说明
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未被授权

9.1.3.2 查询单个设备信息

接口功能

查询指定条件的设备信息。

调用方法

```
GET
```

接口路径

```
https://server:port/iocm/app/dm/v1.3.0/devices/{deviceId}?appId={appId}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	描述
appId	可选	String	应用唯一标识
deviceId	必选	String	设备唯一标识（含网关的设备唯一标识），1-64个字节

返回结果：9.2.8 GetDeviceRspDTO 结构体说明结构

消息事例

Method：

```
GET
```

request：

```
https://server:port/iocm/app/dm/v1.3.0/devices/{deviceId}?appId={appId}
Header:
"app_key:{appId}"
```

```
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

response：

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
            "deviceId":"************************",
            "gatewayId":"************************",
            "nodeType":"GATEWAY",
            "createTime":"20170426T022604Z",
            "lastModifiedTime":"20170426T022604Z",
            "deviceInfo":{
                "nodeId":null,
                "name":null,
                "description":null,
                "manufacturerId":null,
                "manufacturerName":null,
                "mac":null,
                "location":null,
                "deviceType":null,
                "model":null,
                "swVersion":null,
                "fwVersion":null,
                "hwVersion":null,
                "protocolType":null,
                "bridgeId":null,
                "status":"OFFLINE",
                "statusDetail":"NOT_ACTIVE",
                "mute":null,
                "supportedSecurity":null,
                "isSecurity":null,
                "signalStrength":null,
                "sigVersion":null,
                "serialNumber":null,
                "batteryLevel":null
            },
            "services":null,
            "connectionInfo":{
                "protocolType":null
            },
            "location":null,
            "devGroupIds":[]
 }
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在

Http Status Code	error_code	error_desc	说明
200	100403	The device is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不存在
200	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未被授权

9.1.3.3 Application 订阅平台数据

接口功能

Application订阅设备变更通知，当设备发生变更时平台会推送给Application。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/sub/v1.2.0/subscribe
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，如： 1、deviceAdded（添加新设备） 2、deviceInfoChanged（设备信息变化） 3、deviceDataChanged（设备数据变化） 4、deviceDeleted（删除设备） 5、deviceEvent（设备事件） 6、messageConfirm（消息确认） 7、commandRsp（响应命令） 8、serviceInfoChanged（设备信息） 9、ruleEvent（规则事件） 10、bindDevice（设备绑定激活） 11、deviceDatasChanged（设备数据批量变化）
callbackurl	必选	String	回调的url地址。必须使用HTTPS信道回调地址，同时回调地址中必须指定回调地址的端口。(说明：HTTP信道只可用于调测)

返回结果

```
Status Code: 201 Created
```

消息事例

Method：

```
POST
```

request：

```
https://server:port/iocm/app/sub/v1.2.0/subscribe
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
Body:
{
 "notifyType":"deviceInfoChanged",
 "callbackurl":"https://server:port/callbackUri"
}
```

response：

```
Status Code: 201 Created
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100222	The request callbackurl is illegal. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	回调地址非法
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	鉴权应用不存在
500	101017	Get newCallbackUrl from oss failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	从oss获取新的回调地址失败
500	101016	Get iotws address failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	获取iotws地址失败

9.1.3.4 查询设备历史数据

接口功能

Application查询设备历史数据。

调用方法

```
GET
```

接口路径

```
https://server:port/iocm/app/data/v1.1.0/deviceDataHistory?deviceId={deviceId}&
gatewayId={gatewayId}&serviceId ={serviceId}&pageNo={pageNo}&pageSize
={pageSize}&startTime={startTime}&endTime={endTime}&property={property}&appId={appId}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
gatewayId	必选	String	查询参数，网关的设备唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
serviceId	可选	String	服务Id
property	可选	String	服务属性数据
appId	可选	String	应用唯一标识
pageNo	可选	Integer	分页查询参数， pageNo=null时查询内容不分页；取值大于等于0的整数时分页查询，等于0时查询第一页
pageSize	可选	Integer	分页查询参数，取值大于等于1的整数，缺省值为1 设备历史数据最多保存10万条，这个数值有可能被运营商修改
startTime	可选	String	查询参数，根据时间段查询的起始时间;时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z 设备历史数据最多保存90天，这个数值有可能被运营商修改
endTime	可选	String	查询参数，根据时间段查询的结束时间;时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z

返回结果

字段	必选/可选	类型	描述
totalCount	可选	Long	查询的记录数量
pageNo	必选	Long	返回的页码序号(大于等于0的整数,0代表第一页)
pageSize	必选	Long	页码大小,当不分页查询时,取值等于totalCount
list	必选	List<DeviceDataHistoryDTO>	历史记录列表

DeviceDataHistoryDTO

字段	必选/可选	类型	描述
serviceId	必选	String	服务ID
deviceId	必选	String	设备唯一标识，1-64个字节
gatewayId	必选	String	网关的设备唯一标识，1-64个字节。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
appId	必选	String	应用程序ID
data	必选	JsonObject	设备上报的数据，数据是键值对，取值请参见Profile文件中服务能力表里面的propertyties。
timestamp	必选	String	上报数据的时间戳 时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z

消息事例

Method：

```
GET
```

request：

```
https://server:port/iocm/app/data/v1.1.0/deviceDataHistory?deviceId={deviceId}&
gatewayId={gatewayId}&serviceId ={serviceId}&pageNo={pageNo}&pageSize
={pageSize}&startTime={startTime}&endTime={endTime}&property={property}&appId={appId}
Content-Type:application/json
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

response：

```
Status Code: 200 OK
Content-Type:application/json
Body:
{
    "totalCount":"****",
    "pageNo":"*****",
    "pageSize":"*****",
    "deviceDataHistoryDTOs":["*****"]
}
```

注：实例说明调用查询设备历史数据接口的返回结果，如下：

```
{
    "totalCount": 2,
    "pageNo": 0,
    "pageSize": 10,
    "deviceDataHistoryDTOs": [
        {
            "deviceId": "0d34bb9d-73e1-4780-a12f-9243345a4567",
            "gatewayId": "f6ee5aff-53f1-4459-be2f-4fc05ccd7004",
            "appId": "c8855f40-d7a3-4d51-8b46-9f7747839ee2",
            "serviceId": "Battery",
            "data": {
                "batteryLevel": 100
            },
            "timestamp": "20160708T025828Z"
        },
        {
            "deviceId": "0d34bb9d-73e1-4780-a12f-9243345a4567",
            "gatewayId": "f6ee5aff-53f1-4459-be2f-4fc05ccd7004",
            "appId": "c8855f40-d7a3-4d51-8b46-9f7747839ee2",
            "serviceId": "Temperature",
            "data": {
                "temperature": "33.5"
            },
            "timestamp": "20160708T025827Z"
        }
    ]
}
```

上述实例涉及的Profile文件服务能力表如下：

1. **servicetype-capability.json（Battery）**

```
{
    "services": [
        {
            "serviceType": "Battery",
            "description": "Battery",
            "commands": null,
            "properties": [
                {
                    "propertyName": "batteryLevel",
                    "dataType": "int",
                    "required": true,
                    "min": 0,
                    "max": 100,
                    "step": 1,
                    "maxLength": null,
                    "method": "RE",
                    "unit": null,
                    "enumList": null
                },
            ]
        }
    ]
}
```

2. **servicetype-capability.json（Temperature）**

```
{
    "services": [
```

```
        {
            "serviceType": "Temperature",
            "description": "Temperature",
            "commands": null,
            "properties": [
                {
                    "propertyName": "temperature",
                    "dataType": "decimal",
                    "required": true,
                    "min": 0,
                    "max": 100,
                    "step": null,
                    "maxLength": null,
                    "method": "R",
                    "unit": "C",
                    "enumList": null
                }
            ]
        }
    ]
}
```


异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
400	100419	The deviceId and gatewayId can't be both null. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	deviceId和gatewayId不能同时为空

Http Status Code	error_code	error_desc	说明
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未被授权

9.1.3.5 查询设备能力

接口功能

查询设备能力

调用方法

```
GET
```

接口路径

```
https://server:port/iocm/app/data/v1.1.0/deviceCapabilities?
appId={appId}&gatewayId={gatewayId}&deviceId={deviceId}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	描述
appId	可选	String	应用唯一标识，1-64个字节
deviceId	必选	String	设备唯一标识，1-64个字节
gatewayId	可选	String	设备的网关id。当设备是直连设备时，gatewayId务必与deviceId保持一致。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。

返回结果

字段	类型	描述
deviceCapabilities	List<DeviceCapabilityDTO>	自定义结构体，见下表

DeviceCapabilityDTO

字段	类型	描述
deviceId	String	设备唯一标识，1-64个字节
serviceCapabilities	List<ServiceCapabilityDTO >	设备的服务能力列表

ServiceCapabilityDTO

字段	类型	描述
serviceId	String	服务标识
serviceType	String	服务类型
description	String	描述
commands	List<ServiceCommand >	支持的命令名称列表，ServiceCommand结构体
properties	List<ServiceProperty>	支持的属性名称列表，ServiceProperty结构体

ServiceCommand结构体说明

字段	类型	描述
commandName	String(256)	命令名称
paras	ServiceCommandPara	属性列表
responses	ServiceCommandResponse	响应列表

ServiceCommandPara结构体说明

字段	类型	描述
paraName	String(256)	参数名称
dataType	String(256)	数据类型
required	boolean	是否必选
min	String	最小
max	String	最大

字段	类型	描述
step	Double	步长
maxLength	Int	最大长度
unit	String	单位（符号）
enumList	List<String>	枚举类型列表

ServiceCommandResponse结构体说明

字段	类型	描述
responseName	String(256)	响应名称
paras	List<ServiceCommandPara>	属性列表

ServiceProperty结构体说明

字段	类型	描述
propertyName	String(256)	属性名称
dataType	String(256)	数据类型
required	boolean	是否必选
min	String	最小
max	String	最大
step	double	步长
maxLength	int	最大长度
method	String(256)	访问方法：RWE 可读R，可写W，可观察E
unit	String	单位（符号）
enumList	List<String>	枚举类型列表

消息事例

Method：

```
GET
```

request：

```
https://server:port/iocm/app/data/v1.1.0/deviceCapabilities?
appId={appId}&gatewayId={gatewayId}&deviceId={deviceId}
Header:
```

```
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

response：

```
Status Code: 200 OK
Content-Type: application/json
Body:  {
    "deviceId":*****
    "serviceCapabilities" : [{
      "serviceId" : "",
      "serviceType" : "",
      "option" : "",
      "description" : null,
      "commands" : null,
      "properties" : [{
          "propertyName" : "",
          "desc" : "",
          "dataType" : "",
          "enumList" : ["", ""],
          "min" : 0,
          "max" : 0,
          "step" : 0.0,
          "maxLength" : 0.0,
          "unit" : null,
          "method" : "",
          "required" : true
        }],
      "events" : null
    }
 ]
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效

Http Status Code	error_code	error_desc	说明
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未被授权

9.1.4 信令传送

应用场景：NA给设备发送消息

开发指导：平台分别提供了下发至设备的接口，下发消息的具体格式需要与设备约定。

9.1.4.1 创建设备命令 V4

接口功能

应用创建设备命令。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommands?appId={appId}
```

注意事项

携带头域信息：

```
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明：

字段	必选/可选	类型	位置	描述

body	必选	PostDeviceCommandReq	body	设备命令的内容,详见请求结构体说明
appId	可选	String(1-64)	query	应用ID，该应用ID要与当前访问应用有绑定关系方可访问

PostDeviceCommandReq请求结构体说明

字段	必选/可选	类型	描述
deviceId	必选	String(64)	下发命令的设备ID
command	必选	CommandDTOV4	下发命令的信息
callbackUrl	可选	String(1024)	命令状态变化通知地址，必须使用HTTPS信道回调地址 (说明：HTTP信道只可用于调测，不可用于商用环境)； 当命令状态变化时(执行失败，执行成功，超时，发送，已送达)会通知NA，平台会以POST方式发送HTTP消息给应用，请求Body为json字符串，格式形如：{"deviceId":"XXX","commandId":"XXX","result":{"resultCode":"SENT","resultDetail":{...}}}。当resultCode为SENT或者DELIVERED时，resultDetail为空；resultCode为SUCCESSFUL时，resultDetail的值是编解码插件设置的，请参考《**编解码库开发与升级指南**》中**表3 设备对平台命令的应答**。
expireTime	可选	Integer(>=0)	下发命令有效的超期时间，单位为秒，表示设备命令在创建后expireTime秒内有效，超过这个时间范围后命令将不再下发，如果未设置则默认为48小时

CommandDTOV4 结构体说明

字段	必选/可选	类型	描述

serviceId	必选	String(1-64)	命令对应的服务ID，不能为空字符串
method	必选	String(1-128)	命令服务下具体的命令名称，服务属性等，不能为空字符串
paras	可选	ObjectNode	命令参数的jsonString，具体格式需要应用和设备约定。

返回结果

投递成功：Status Code: 201 Created

返回参数参见附录**9.2.6 DeviceCommandResp 结构体说明** 结构体说明

消息示例

Method：

```
POST
```

request：

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommands?appId={appId}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
Body:
{
       "deviceId": "********",
       "command": {
               "serviceId": "********",
               "method": "********",
               "paras": {
                       "paraName1": "paraValue1",
                       "paraName2": "paraValue2"
        }
    },
       "callbackUrl": "https://server:port/callbackUri"
}
```

response：

```
Status Code: 201 Created
Content-Type:application/json

Body:
{
       "commandId": "********",
       "appId": "********",
       "deviceId": "********",
       "command": {
               "serviceId": "********",
               "method": "********",
               "paras": {
                       "paraName1": "paraValue1",
```

```
                    "paraName2": "paraValue2"
                }
            },
            "callbackUrl": "https://server:port/callbackUri",
                "expireTime": null,
                "status": "PENDDING",
                "result": null,
                "creationTime": "20170222T164000Z",
                "executeTime": null,
                "platformIssuedTime": null,
                "deliveredTime": null,
                "issuedTimes": null
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效
500	100220	Get AppKey from header failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	获取appKey失败
500	100001	Internal server error. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	立即下发命令发送超时
404	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在

404	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在
404	100431	The serviceType is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	服务类型不存在

9.1.4.2 查询设备命令 V4

接口功能

应用查询设备命令。

调用方法

```
GET
```

接口路径

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommands?
pageNo={pageNo}&pageSize={pageSize}&deviceId={deviceId}&startTime={startTime}&endTime={endTime}&app
Id={appId}
```

注意事项

http携带头域信息：

```
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	位置	描述
pageNo	可选	Integer(>=0)	query	查询的页码，大于等于0，默认0
pageSize	可选	Integer(>=1&&<=1000)	query	查询每页的数量，大于等于1，最大值1000，默认1000
deviceId	可选	String(64)	query	指定查询设备命令的设备ID

startTime	可选	String	query	查询开始时间，查询下发命令时间在startTime之后的记录。时间格式：yyyyMMdd'T'HHmmss'Z' 如：20151212T121212Z
endTime	可选	String	query	查询结束时间，查询下发命令时间在endTime之前的记录。时间格式：yyyyMMdd'T'HHmmss'Z' 如：20151212T121212Z
appId	可选	String	query	应用ID，该应用ID要与当前访问应用有绑定关系方可访问

返回结果

查询成功：Status Code: 200 OK

DeviceCommandQueryResp 结构体说明

字段	必选/可选	类型	描述
pagination	必选	Pagination	页码信息
data	可选	List<DeviceCommandResp>	设备命令列表，**9.2.6 DeviceCommandResp 结构体说明**

Pagination 结构体说明

字段	必选/可选	类型	描述
pageNo	必选	long	页码
pageSize	必选	long	分页大小
totalSize	必选	long	记录总数

返回结果

查询成功：Status Code: 200 OK

消息示例

Method：

```
GET
```

request：

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommands?
pageNo=0&pageSize=10&deviceId=******&startTime=20151112T101012Z&endTime=20151212T121212Z
```

```
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

response：

```
Status Code: 200 OK
Content-Type:application/json

Body:
{
    "pagination": {
           "pageNo": 0,
           "pageSize": 20,
           "totalSize": 100
    },
       "data": [
               {
               "commandId": "********",
               "appId": "********",
               "deviceId": "********",
               "command": {
                   "serviceId": "********",
                   "method": "********",
                   "paras": {
                               "paraName1": "paraValue1",
                               "paraName2": "paraValue2"
                    }
            },
               "callbackUrl": "https://server:port/callbackUri",
               "expireTime": null,
               "status": "PENDDING",
               "result": null,
               "creationTime": "20170222T164000Z",
               "executeTime": null,
               "platformIssuedTime": null,
               "deliveredTime": null,
               "issuedTimes": null
       },
               {
               "commandId": "********",
               "appId": "********",
               "deviceId": "********",
               "command": {
                   "serviceId": "********",
                   "method": "********",
                   "paras": {
                               "paraName1": "paraValue1",
                               "paraName2": "paraValue2"
                    }
            },
               "callbackUrl": "https://server:port/callbackUri",
               "expireTime": null,
               "status": "PENDDING",
               "result": null,
               "creationTime": "20170222T164000Z",
               "executeTime": null,
               "platformIssuedTime": null,
               "deliveredTime": null,
               "issuedTimes": null
       }
    ]
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	无效的输入
500	100220	Get AppKey from header failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	获取appKey失败
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
200	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在
200	100403	The device is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不存在

200	100434	The device command is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	命令不存在

9.1.4.3 修改设备命令 V4

接口功能

应用修改命令信息，当前仅支持修改设备命令状态为撤销，即为将设备命令撤销。

调用方法

```
PUT
```

接口路径**

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommands/{deviceCommandId}?appId={appId}
```

注意事项

http携带头域信息：

```
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	位置	描述
deviceCommandId	必选	String	path	待修改的设备命令ID
appId	可选	String	query	应用ID，该应用ID要与当前访问应用有绑定关系方可访问
body	必选	UpdateDeviceCommandReq	body	修改设备命令的内容，详见请求结构体说明

UpdateDeviceCommandReq结构体说明

字段	必选/可选	类型	描述

status	必选	String	命令执行结果，可选值：CANCELED

返回结果

```
Status Code: 200 OK
```

返回参数：参见附录**9.2.6 DeviceCommandResp 结构体说明** 结构体说明

消息示例

Method：

```
PUT
```

request：

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommands/{deviceCommandId}?appId={appId}
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
Body:
{
    "status":"CANCELED"
}
```

response：

```
StatusCode: 200 OK
{
       "commandId": "********",
       "appId": "********",
       "deviceId": "********",
       "command": {
           "serviceId": "********",
           "method": "********",
           "paras": {
                       "paraName1": "paraValue1",
                       "paraName2": "paraValue2"
           }
       },
       "callbackUrl": "https://server:port/callbackUri",
       "expireTime": null,
       "status": "PENDDING",
       "result": null,
       "creationTime": "20170222T164000Z",
       "executeTime": null,
       "platformIssuedTime": null,
       "deliveredTime": null,
       "issuedTimes": null
}
```

异常返回码

Http Status Code	error_code	error_desc	说明

400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效
500	100220	Get AppKey from header failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	获取appKey失败
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用没有操作权限
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
200	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在
200	100434	The device command is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	命令不存在
200	100435	The device command already canceled, expired or executed, Cannot cancel. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	命令已撤销、过期或执行，无法撤销

9.1.4.4 创建设备命令撤销任务 V4

接口功能

应用创建设备命令撤销任务，该任务用于撤销指定设备ID下的所有设备命令。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommandCancelTasks?appId={appId}
```

注意事项

http携带头域信息：

```
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	位置	描述
appId	可选	String	query	应用ID，该应用ID要与当前访问应用有绑定关系方可访问
body	必选	CreateDeviceCommandCancelTaskReq	body	设备命令撤销任务参数，详见请求结构体说明

CreateDeviceCommandCancelTaskReq请求结构体说明

字段	必选/可选	类型	描述
deviceId	必选	String(1-64)	待撤销设备命令的设备ID，撤销任务将会撤销所有给该设备下发的设备命令

返回结果

```
Status Code: 201 Created
```

DeviceCommandCancelTaskResp 响应结构体说明

字段	必选/可选	类型	描述

taskId	必选	String(1-64)	设备命令撤销任务的任务ID
appId	必选	String(1-64)	设备命令撤销任务所属的应用ID
deviceId	必选	String(1-64)	设备命令撤销任务指定撤销命令的设备ID
status	必选	String	撤销任务的任务状态，WAITTING表示等待执行中，RUNNING表示撤销任务正在执行，SUCCESS表示撤销任务执行成功，FAILED表示撤销任务执行失败，PART_SUCCESS表示推销任务部分执行成功。
totalCount	必选	Integer	撤销的设备命令总数
deviceCommands	可选	List<DeviceCommandResp>	设备命令撤销任务的任务ID，参见附录**9.2.6 DeviceCommandResp 结构体说明** 结构体说明

消息示例

Method：

```
POST
```

request：

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommandCancelTasks?appId={appId}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
Body:

{
    "deviceId": "********"
}
```

response：

```
Status Code: 201 Created
{
    "taskId": "********",
    "appId": "********",
    "deviceId": "********",
    "status": "WAITTING",
    "totalCount": 1,
    "deviceCommands": [
        {
                "commandId": "********",
                "appId": "********",
                "deviceId": "********",
                "command": {
                    "serviceId": "********",
                    "method": "********",
                    "paras": {
```

```
                                "paraName1": "paraValue1",
                                "paraName2": "paraValue2"
                            }
                        },
                            "callbackUrl": "https://server:port/callbackUri",
                            "expireTime": null,
                            "status": "PENDDING",
                            "result": null,
                            "creationTime": "20170222T164000Z",
                            "executeTime": null,
                            "platformIssuedTime": null,
                            "deliveredTime": null,
                            "issuedTimes": null
                    }
                ]
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	无效的输入
400	100220	Get AppKey from header failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	获取appKey失败
401	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在

404	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在
404	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不存在

9.1.4.5 查询设备命令撤销任务 V4

接口功能

应用查询设备命令撤销任务。

调用方法

```
GET
```

接口路径

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommandCancelTasks?
pageNo={pageNo}&pageSize={pageSize}&taskId={taskId}&deviceId={deviceId}&status={status}&startTime={
startTime}&endTime={endTime}&appId={appId}
```

注意事项

http携带头域信息：

```
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

字段	必选/可选	类型	位置	描述
pageNo	可选	Integer(>=0)	query	查询的页码，大于等于0，默认0
pageSize	可选	Integer(>=1&&<=1000)	query	查询每页的数量，大于等于1，最大值1000，默认1000

taskId	可选	String	query	指定任务ID查询设备命令撤销任务
deviceId	可选	String	query	指定设备ID查询设备命令撤销任务
status	可选	String	query	查询指定任务状态的设备命令撤销任务
startTime	可选	String	query	查询开始时间，查询创建撤销设备命令任务时间在startTime之后的记录。时间格式：yyyyMMdd'T'HHmmss'Z' 如：20151212T121212Z
endTime	可选	String	query	查询结束时间，查询创建撤销设备命令任务时间在endTime之前的记录。时间格式：yyyyMMdd'T'HHmmss'Z' 如：20151212T121212Z
appId	可选	String	query	应用ID，该应用ID要与当前访问应用有绑定关系方可访问

返回结果

查询成功：Status Code: 200 OK

字段	必选/可选	类型	描述
pagination	必选	Pagination	页码信息
data	可选	List<DeviceCommandCancelTaskResp>	设备命令列表

Pagination 响应结构体说明

字段	必选/可选	类型	描述
pageNo	必选	long	页码
pageSize	必选	long	分页大小
totalSize	必选	long	记录总数

DeviceCommandCancelTaskResp 响应结构体说明

字段	必选/可选	类型	描述
taskId	必选	String(1-64)	设备命令撤销任务的任务ID
appId	必选	String(1-64)	设备命令撤销任务所属的应用ID
deviceId	必选	String(1-64)	设备命令撤销任务指定撤销命令的设备ID
status	必选	String	撤销任务的任务状态，WAITTING表示等待执行中，RUNNING表示撤销任务正在执行，SUCCESS表示撤销任务执行成功，FAILED表示撤销任务执行失败，PART_SUCCESS表示推销任务部分执行成功。
totalCount	必选	Integer	撤销的设备命令总数
deviceCommands	可选	List<DeviceCommandResp>	设备命令撤销任务的任务ID，参见附录**9.2.6 DeviceCommandResp 结构体说明** 结构体说明

消息示例

Method：

```
GET
```

request：

```
https://server:port/iocm/app/cmd/v1.4.0/deviceCommandCancelTasks?
pageNo={pageNo}&pageSize={pageSize}&taskId={taskId}&deviceId={deviceId}&status={status}&startTime={
startTime}&endTime={endTime}&appId={appId}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

response：

```
Status Code: 200 OK
Content-Type:application/json

Body:
{
    "pagination": {
           "pageNo": 0,
           "pageSize": 20,
           "totalSize": 100
    },
       "data": [
              {
            "taskId": "********",
            "appId": "********",
            "deviceId": "********",
            "status": "WAITTING",
            "totalCount": 1,
            "deviceCommands": [
                {
                       "commandId": "********",
                       "appId": "********",
                       "deviceId": "********",
                       "command": {
                           "serviceId": "********",
                           "method": "********",
                           "paras": {
                                      "paraName1": "paraValue1",
                                      "paraName2": "paraValue2"
                        }
                    },
                       "callbackUrl": "https://server:port/callbackUri",
                       "expireTime": null,
                       "status": "PENDDING",
                       "result": null,
                       "creationTime": "20170222T164000Z",
                       "executeTime": null,
                       "platformIssuedTime": null,
                       "deliveredTime": null,
                       "issuedTimes": null
                }
            ]
       },
              {
            "taskId": "********",
            "appId": "********",
            "deviceId": "********",
            "status": "WAITTING",
            "totalCount": 1,
            "deviceCommands": [
                {
                       "commandId": "********",
                       "appId": "********",
                       "deviceId": "********",
                       "command": {
                           "serviceId": "********",
                           "method": "********",
                           "paras": {
                                      "paraName1": "paraValue1",
```

```
                                    "paraName2": "paraValue2"
                                }
                            },
                            "callbackUrl": "https://server:port/callbackUri",
                            "expireTime": null,
                            "status": "PENDDING",
                            "result": null,
                            "creationTime": "20170222T164000Z",
                            "executeTime": null,
                            "platformIssuedTime": null,
                            "deliveredTime": null,
                            "issuedTimes": null
                        }
                    ]
                }
            ]
}
```


异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	无效的输入
400	100220	Get AppKey from header failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	获取appKey失败
401	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在

404	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在
404	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不存在

9.1.5 设备服务调用

9.1.5.1 设备服务调用

接口功能

Application给设备发送命令消息，实现对传感器的实时控制。平台分别提供了下发至设备或者具体某传感器的控制命令接口，下发消息的具体格式需要Application与设备自定义，平台在接口中进行封装，透传。

Application发送命令给网关（或者网关下的设备）指定的服务下发命令/事件。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/signaltrans/v1.1.0/devices/{deviceId}/services/{serviceId}/
sendCommand?appId={appId}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
appId	可选	String	query	授权应用Id
deviceId	必选	String	path	设备唯一标识，1-64个字节

参数	必选/可选	类型	位置	描述
serviceId	必选	String	path	服务id，1-64个字节 唯一标识一个服务，设备的服务类型不冲突，服务类型就是服务id。
header	必选	CommandNA2CloudHeader	Body	见下表CommandNA2CloudHeader字段说明
body	必选	JsonObject	Body	命令的具体内容，取值长度无要求。 JsonObject里面是一个个健值对。每个健都是profile中命令的参数名（paraName）。

CommandNA2CloudHeader**字段说明：**

字段	必选/可选	类型	描述
requestId	可选	String(128)	如果填写该requestId，就是用填写的requestId标识一个命令，这里的requestId不能重复，由发起端APP按照下面的规则保证其唯一性：Generate seqNum：APP生成序列号seqNum，在POST请求BODY中携带，序列号规则采用：UUID_XXXX，UUID会根据JAVA自带算法生成，每次APP启动后生成一个新的值，后续使用时不变，直到下次APP重新启动再生成一个新的UUID值，XXXX的取值范围：0001-9999，达到9999后重新从0001开始，如此循环
mode	必选	Enum	是否要确认消息，不需要确认消息：NOACK，需要确认消息：ACK，其它值无效
from	可选	String(128)	表示消息发布者的地址： App发起的请求：/users/{userId} NA发起的请求：/{serviceName} 平台发起的请求：/cloud/{serviceName}
toType	可选	Enum	CLOUD、GATEWAY
to	可选	String(128)	接收者地址
method	必选	String(1-32)	命令名称，profile命令中定义的命令名（commandName），如："DISCOVERY"
callbackURL	可选	String(1024)	回调的url地址。必须使用HTTPS信道回调地址，同时回调地址中必须指定回调地址的端口。(说明：HTTP信道只可用于调测)

返回结果

命令已成功投递: Status Code: 200 OK

命令已接收处理：StatusCode: 202 Accepted

字段	类型	描述
status	String(128)	命令状态： sent --- 已发送 deliveried ---toType 为 CLOUD 的请求才有该返回值 failed--- toType 为 CLOUD的请求才有该返回值
timestamp	String(128)	发送命令的时间戳 时间格式：yyyyMMdd’T’HHmmss’Z’ 如： 20151212T121212Z
requestId	String(128)	toType 为 GATEWAY，如果请求有requestId则同请求的requestId一致，如果请求无requestId 则平台分配一个序列号 toType为CLOUD，则为null;

消息示例

Method:

```
POST
```

Request:

```
https://server:port/iocm/app/signaltrans/v1.1.0/devices/{deviceId}/services/{serviceId}/
sendCommand?appId={appId}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;

Body:
{
    "header": {
        "mode": "ACK",
        "from": "/users/23212121",
        "method": "INVITE-INIT",
        "callbackURL": "https://186.25.1.49:8043/na/iocm/message/confirm"
    },
    "body": {
        "from": "**********",
        "sessionID": "**********",
        "sdp": "**********"
    }
}
```

Response:

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
    "requestId": "**********",
    "status": "sent",
    "timestamp": "**********"
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未授权
200	100418	The deviceData is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备数据不存在

Http Status Code	error_code	error_desc	说明
500	100023	The data in dataBase is abnomal. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	数据库中数据异常
400	102203	CommandName is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	命令名称无效
200	100431	The serviceType is not exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	服务类型不存在
200	100428	The device is not online. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备不在线
200	100432	The device command is muted. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	设备已被muted

9.1.6 批量处理

9.1.6.1 创建批量任务

接口功能

NA创建批量任务。支持给本应用和授予权限的其它应用创建批量任务。

调用方法

```
POST
```

接口路径

```
https://server:port/iocm/app/batchtask/v1.1.0/tasks
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
appId	必选	String(64)	body	应用唯一标识
timeout	必选	Integer	body	任务超时时间
taskName	必选	String	body	任务名称
taskType	必选	String	body	任务类型DeviceCmd
param	必选	ObjectNode	body	根据taskType任务类型的对应不同类型参数，详见下表param参数结构说明

taskType参数取值DeviceCmd（批量下发命令任务），param参数：

字段	必选/可选	类型	描述
type	必选	String	批量命令类型： DeviceList / DeviceType / DeviceArea / GroupList / Broadcast
deviceList	可选	List<String>	deviceId列表，当命令类型为DeviceList时，需要填写
deviceType	可选	String	设备类型，当命令类型为DeviceType时需要填写。

字段	必选/可选	类型	描述
manufacturerId	可选	String	厂商ID，当命令类型为DeviceType时可填写
model	可选	String	设备型号，当命令类型为DeviceType时可填写。
deviceLocation	可选	String	设备位置，当命令类型为DeviceArea时，需要填写
groupList	可选	List<String>	组名字列表，当命令类型为GroupList时需要填写。
command	必选	CommandDTOV1	命令，查看CommandDTOV1结构定义

CommandDTOV1结构体说明

字段	必选/可选	类型	描述
serviceId	必选	String(64)	服务id
method	必选	String(32)	命令名称，服务属性等
paras	可选	ObjectNode	命令参数的jsonString

返回结果

Status Code: 200 OK

字段	类型	描述
taskID	String	任务ID

消息示例

Method: POST

Request:

```
https://server:port/iocm/app/batchtask/v1.1.0/tasks
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
Body:
{
 "appId" : "******",
 "timeout" : 1000,
 "taskName" : "******",
    "taskType" : "DeviceReg",
```

```
    "param" : {
      "fileId" : "******"
     }
}
```

Response:

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
    "taskID":"******"
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
401	100002	Invalid access token. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的token信息
200	105001	The batchTask count has reached the limit. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	未完成的任务数大于等于10个时，返回任务达到数量限制
200	105002	The batchTask name has exist. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	任务名字已存在
400	100007	Bad request message. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	错误的请求消息

Http Status Code	error_code	error_desc	说明
401	100028	The user has no right. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	用户没有操作权限

9.1.6.2 查询单个任务信息

接口功能

NA查询单个任务信息。支持查询本应用和授予权限的其它应用创建的批量任务信息。

调用方法

```
GET
```

接口路径

```
https://server:port/iocm/app/batchtask/v1.1.0/tasks/{taskId}?appId={appId}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
taskId	必选	String	path	任务Id
appId	可选	String	query	授权应用Id

返回结果

Status Code: 200 OK

字段	类型	描述
taskId	String	任务ID
taskName	String	任务名字
appId	String	任务的归属appId

字段	类型	描述
operator	String	下发任务的操作员
taskFrom	String	Portal/Northbound
taskType	String	任务类型 DeviceCmd
status	String	任务状态 Pending/Running/Complete/Timeout
startTime	String	任务的创建时间
timeout	Int	任务的超时时间
progress	Int	任务进度，千分比0-1000，向下取整
totalCnt	Int	任务详情总数
successCnt	Int	成功的任务详情数
failCnt	Int	失败的任务详情数
timeoutCnt	Int	超时的任务详情数
expiredCnt	Int	未执行的失效任务详情数
completeCnt	Int	完成的任务详情数，完成=成功+失败+超时
successRate	Int	任务成功率，千分比0-1000，向下取整
param	ObjectNode	不同的任务类型的具体参数

param参数结构说明：

taskType参数取值DeviceCmd（批量下发命令任务），param参数：

字段	必选/可选	类型	描述
commandType	必选	String	批量命令类型： DeviceList / DeviceType / DeviceArea / GroupList / Broadcast
deviceType	可选	String	设备类型
manufacturerId	可选	String	厂商Id
model	可选	String	设备类型
serviceId	必选	String	设备中的服务标识
method	必选	String	命令名称，如："DISCOVERY"

消息示例

Method: GET

Request:

```
https://server:port/iocm/app/batchtask/v1.1.0/tasks/{taskId}?appId={appId}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

Response:

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
    "taskId" : "********",
    "taskName" : "********",
    "appId" : "********",
    "operator" : "********",
    "taskFrom" : "********",
    "taskType" : "DeviceCmd",
    "status" : "********",
    "startTime" : "********",
    "timeout" : 1000,
    "progress" : 100,
    "totalCnt" : 100,
    "successCnt" : 70,
    "failCnt" : 10,
    "timeoutCnt" : 10,
    "expiredCnt" : 10,
    "completeCnt" : 100,
    "successRate" : 70,
    "param" : {
       "fileName" : "******"
     }
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在

Http Status Code	error_code	error_desc	说明
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未被授权

9.1.6.3 查询任务详情信息

接口功能

NA查询单个任务的任务详情信息。支持查询本应用和授予权限的其它应用创建的批量任务信息。

调用方法

```
GET
```

接口路径

```
https://server:port/iocm/app/batchtask/v1.1.0/taskDetails?
taskId={taskId}&appId={appId}&status={status}&index={index}&nodeId={nodeId}&deviceId={deviceId}&com
mandId={commandId}&pageNo={pageNo}&pageSize={pageSize}
```

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
appId	可选	String	query	授权应用Id
taskId	必选	String	query	任务Id
status	可选	String	query	任务详情状态 Pending/WaitResult/Success/Fail/Timeout
pageNo	可选	Integer	query	分页查询参数，pageNo=null时查询内容不分页；取值大于等于0的整数时分页查询,等于0时查询第一页
pageSize	可选	Integer	query	分页查询参数，取值大于等于1的整数,缺省：1

参数	必选/可选	类型	位置	描述
index	可选	Integer	query	文件里第几行,查询批量注册设备使用
nodeId	可选	String	query	设备nodeId，查询批量注册设备使用
deviceId	可选	String	query	设备Id，查询批量命令使用
commandId	可选	String	query	命令Id，查询批量命令使用

返回结果

Status Code: 200 OK

字段	类型	描述
pageNo	Integer	页码
pageSize	Integer	每页数据记录数量
totalCount	Integer	总数据记录数量
taskDetails	LIST<QueryTaskDetailDTOCloud2NA>	设备详情信息列表

QueryTaskDetailDTOCloud2NA结构

字段	类型	描述
status	String	Pending/WaitResult/Success/Fail/Timeout，任务详情状态
output	ObjectNode	批量注册设备或批量下发命令的输出，详见下表
error	String	错误原因，格式为： {"error_code": "", "error_desc": ""}
param	ObjectNode	不同的任务类型的具体参数，详见下表

批量下发命令任务，output为：

字段	类型	描述
requestId	String(64)	请求id
commandResult	CommandResult	命令的响应

批量下发命令任务，param为：

字段	类型	描述
deviceId	String	设备Id
commandId	String	命令Id

消息示例

Method: GET

Request:

```
https://server:port/iocm/app/batchtask/v1.1.0/taskDetails?
taskId={taskId}&appId={appId}&status={status}&index={index}&nodeId={nodeId}&deviceId={deviceId}&com
mandId={commandId}&pageNo={pageNo}&pageSize={pageSize}
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

Response:

```
Status Code: 200 OK
Content-Type: application/json
Body:
{
    "totalCount": 1,
    "pageNo": 0,
    "pageSize": 1,
    "taskDetails": [
        {
            "status": "Fail",
            "output": null,
            "error": "{\"error_code\":100007,\"error_desc\":\"Badrequest message.\"}",
            "param": {
                "index": 0,
                "nodeId": "1666"
            }
        }
    ]
}
```

异常返回码

Http Status Code	error_code	error_desc	说明
400	100022	The input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	输入参数无效

Http Status Code	error_code	error_desc	说明
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用未被授权

9.1.7 规则

Application能够根据具体的业务需要，制定规则（条件）和动作，当网关的状态达到指定规则后，以邮件或者短信的方式对用户进行实时提醒。根据业务需要，编排规则，指定场景触发。

9.1.7.1 创建规则

接口功能

创建规则

请求方法

POST

请求路径

https://server:port/iocm/app/rule/v1.2.0/rules

注意事项

携带头域信息

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述

app_key	必选	String	header	鉴权用户名，填写创建应用时返回的id。
Authorization	必选	String	header	鉴权信息，填写鉴权接口返回的accessToken。

消息体参数定义查看RuleDTO1.2定义说明。

RuleDTO1.2 结构体说明

参数	必选/可选	类型	位置	描述
app_key	必选	String	header	鉴权用户名，填写创建应用时返回的id。
Authorization	必选	String	header	鉴权信息，填写鉴权接口返回的accessToken。
ruleId	可选	String	body	规则实例的id，仅在规则更新时有效，其余情况应为空
appKey	可选	String	body	应用实例的key值(1-128)
name	必选	String	body	规则名（并且接受1~128个字符）
description	可选	String	body	规则描述(1-256)
author	必选	String	body	创建此规则的用户的id（1-128）（不能为空）
conditions	可选	List < Json >	body	条件列表，自定义结构体（见下表）, 与groupExpress二选一必填
logic	可选	String	body	多条件之间的逻辑关系，支持and和or，默认为and
timeRange	可选	TimeRange	body	条件场景的时间段。TimeRange为自定义结构体（见下表Condition-CycleTimer）
actions	定时场景必选	List < Json >	body	规则的动作。Action为自定义结构体。（见下面Action-DeviceCMD、Action -SMS、Action -email、Action-DeviceAlarm）
matchNow	可选	String	body	表示是否立即触发，即是否理解进行条件判断，条件符合的话执行动作，取值有“yes”,“no”，默认为“no”
status	可选	String	body	规则的状态，“active”代表激活状态，“inactive”代表未激活，默认为“active”状态

groupExpress	可选	GroupExpress	body	复杂多条件表达式 。为自定义结构体GroupExpress，与conidtions二选一必填
timezoneID	可选	String	body	国际时区，如Asia/Shanghai、UTC
triggerSources	可选	List <Json	body	触发源列表，自定义结构体TriggerSource，只与GroupExpress联用构造复杂多条件规则，目前有DEVICE类型和TIMER类型

Device-TriggerSource 结构体说明

字段	必选/可选	字段类型	描述
type	必选	Enum	设备触发源类型 DEVICE
deviceId	必选	String	设备Id(1-36)
serviceId	必选	String	服务id(1-128)

Timer-TriggerSource 结构体说明

字段	必选/可选	字段类型	描述
type	必选	Enum	设备触发源类型 TIMER

GroupExpress结构体说明：

字段	必选/可选	字段类型	描述
operator	可选	Enum	操作符and/or，默认为and
groupElements	必选	List<GroupElement>	最外层操作符对应的元素，自定义结构体GroupElement

GroupElement结构体说明：

字段	必选/可选	字段类型	描述
operator	可选	Enum	操作符and/or，默认为and
elements	必选	List<Json>	内层表达式运算的元素,一个element表示一个condition,自定义结构conditions

Condition-DeviceData结构说明：

字段	必选/可选	字段类型	描述
type	必选	String	条件触发condition类型，取值为"DEVICE_DATA"
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
deviceInfo	必选	DeviceInfo	设备数据的定位信息，自定义结构，根据定位信息读取的数据作为比较表达式的左值
operator	必选	String	数据比较的操作符数据比较的操作符(目前支持的操作符有：>,<,>=,<=,=,between,geo.circle.in, geo.circle.out，具体使用见举例说明)
value	可选	String	比较表达式的右值(与between操作符联用时，右值表示最小值和最大值用逗号隔开如“20,30”表示大于等于20 小于30，应用场景:温度在20到30之间触发规则)
transInfo	可选	Json	不需要平台理解的信息
duration	可选	Integer	规则执行延时时间，单位为分钟，默认为0不延时
strategy	必选	Strategy	配置条件处理策略，自定义结构体Strategy

Strategy结构体说明：

字段	必选/可选	字段类型	描述
trigger	可选	String	pulse：上报的数据满足条件则触发，不判断上一次的数据 reverse：上一次数据不满足条件，本次数据满足条件则触发 默认为pulse
eValidTime	可选	Integer (>=-1)	事件时效性，单位为s，表示收到的传感器数据多久有效，（以eventTime为基准

DeviceInfo结构体说明：

字段	必选/可选	字段类型	描述

deviceId	必选	String	设备id
path	必选	String	以"/"分割的信息路径，按照当前的数据模型，路径为ServiceId/DataProperty，例如门磁状态为"DoorWindow/status"

Condition-DeviceGroupData结构说明：

字段	必选/可选	字段类型	描述
type	必选	String	条件触发condition类型，取值为"DEVICE_DATA_GROUP"
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
deviceGroupInfo	必选	DeviceGroupInfo	组设备数据的定位信息，自定义结构，根据定位信息读取的数据作为比较表达式的左值
operator	必选	String	数据比较的操作符数据比较的操作符(目前支持的操作符有：>,<,>=,<=,=,between,geo.circle.in,geo.circle.out，具体使用见举例说明)
value	可选	String	比较表达式的右值(与between操作符联用时，右值表示最小值和最大值用逗号隔开如“20,30”表示大于等于20 小于30，应用场景:温度在20到30之间触发规则)
transInfo	可选	Json	不需要平台理解的信息
duration	可选	Integer	规则执行延时时间，单位为分钟，默认为0不延时
strategy	必选	Strategy	配置条件处理策略，自定义结构体Strategy

DeviceGroupInfo结构体说明：

字段	必选/可选	字段类型	描述
groupId	必选	String	设备组id

path	必选	String	以"/"分割的信息路径，按照当前的数据模型，路径为ServiceId/DataProperty，例如门磁状态为"DoorWindow/status",属性支持多个,比如OBD需要两个属性,经度和纬度”, 多个属性中间以逗号分隔

Condition-DeviceTypeData结构说明：

字段	必选/可选	字段类型	描述
type	必选	String	条件触发condition类型，取值为"DEVICE_DATA_TYPE"
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
deviceTypeInfo	必选	DeviceTypeInfo	组设备数据的定位信息，自定义结构，根据定位信息读取的数据作为比较表达式的左值
operator	必选	String	数据比较的操作符数据比较的操作符(目前支持的操作符有：>,<,>=,<=,=,between,geo.circle.in,geo.circle.out，具体使用见举例说明)
value	可选	String	比较表达式的右值(与between操作符联用时，右值表示最小值和最大值用逗号隔开如“20,30”表示大于等于20 小于30，应用场景:温度在20到30之间触发规则)
transInfo	可选	Json	不需要平台理解的信息
duration	可选	Integer	规则执行延时时间，单位为分钟，默认为0不延时
strategy	必选	Strategy	配置条件处理策略，自定义结构体Strategy

DeviceTypeInfo结构体说明：

字段	必选/可选	字段类型	描述

manufacturerId	必选	String	生产商Id
model	必选	String	设备型号
deviceType	必选	String	设备类型
protocolType	必选	String	协议类型
path	必选	String	以"/"分割的信息路径，按照当前的数据模型，路径为ServiceId/DataProperty，例如门磁状态为"DoorWindow/status",属性支持多个,比如OBD需要两个属性,经度和纬度”，多个属性中间以逗号分隔

Condition-DailyTimer结构说明：

字段	必选/可选	字段类型	描述
type	必选	String	定点触发condition类型，“DAILY_TIMER”
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
time	必选	String	指定执行时间点，格式为"HH:MM"
daysOfWeek	可选	String	星期列表，以逗号分隔

Condition-CycleTimer结构说明：

字段	必选/可选	字段类型	描述
type	必选	String	周期性触发condition类型，取值为“CYCLE_TIMER”
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
timeRange	必选	timeRange	限定的时间范围
interval	必选	Integer	周期，单位为分钟

Action-DeviceCMD结构说明：

字段	必选/可选	字段类型	描述

type	必选	String	DEVICE_CMD
id	可选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
appKey	可选	String(64)	应用实例的key值
deviceId	可选	String(64)	设备id(不填时，命令下发给触发规则的设备)
cmd	必选	CMD	命令内容
cmdVersion	可选	String	命令下发的版本
cmdMetaData	可选	ObjectNode	命令下发的参数
transInfo	可选	Json	不需要平台理解的信息

CMD 结构说明：

字段	必选/可选	字段类型	描述
messageType	必选	String(128)	命令名字
messageBody	可选	Json	下发命令的消息内容
serviceId	必选	String(64)	服务id

当cmdVersion="V1.3.0"时，OjbectNode(cmdMetaData)结构说明：

字段	必选/可选	字段类型	描述
requestId	可选	String(64)	命令请求Id
callbackUrl	可选	String(1024)	命令状态变化通知地址，当命令状态变化时(执行失败，执行成功，超时，发送，已送达)会通知NA
expireTime	可选	Integer	已废弃参数！异步命令1.3版本创建规则Demo

Action -SMS结构说明：

字段	必选/可选	字段类型	描述
type	必选	String	发送短信action类型，取值为“SMS”

id	可选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
msisdn	必选	String	短信通知地址
content	可选	String	通知内容通知内容（短信content为空时，短信邮件由北向应用发送）
subject	可选	String	通知主题
title	可选	String	短信别名
accountId	可选	String	账户Id

Action -email结构说明：

字段	必选/必选	字段类型	描述
type	必选	String	发送邮件action类型，取值为“EMAIL”
id	必选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
email	必选	String	邮箱地址
content	必选	String	通知内容通知内容（邮件content或者subject为空时，短信邮件由北向应用发送）
subject	必选	String	通知主题
title	必选	String	邮件别名
accountId	必选	String	账户Id

Action-DeviceAlarm结构说明：

字段	必选/可选	字段类型	描述
type	必选	String	发送邮件action类型，取值为“DEVICE_ALARM”
id	可选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
name	必选	String	告警名
status	必选	String	告警状态 fault/recovery
severity	必选	String	告警级别 warning/minor/major/critical
description	可选	String	告警描述

成功结果

状态码StatusCode：200 OK

内容类型Content-Type：application/json 返回结构：

字段	字段类型	描述
ruleId	String	规则实例id

异常返回码

HttpStatus	error_code	error_desc	说明
401	100002	Invalid access token.	错误的token信息
401	101409	Authentication failure. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中没有携带appId
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中携带appId不存在
200	101403	The rule is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	规则不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用没有权限

200	101402	The rule name already existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	规则名称已存在
500	100023	The data in dataBase is abnormal. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	获取配置参数数据失败
200	101404	The rule count of each app has reached the upper limit, the current up limit is 30 entries. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用已有规则数量查过规定的最大值
400	100216	The application input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	规则属性传参不合法
500	101424	Internal server error happened when create rule. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	创建规则异常
200	101401	The rule input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	接口传参不合法

400	101416	The action of rule is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	Action参数不合法
400	101418	Action id duplicated in same rule. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	Action id重复
400	101425	Rule api parameter is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	接口传参不合法
500	50204	Convert json to class failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	数据格式转换失败

消息示例

```
Curl
curl -X POST
--header "Content-Type: application/json"
--header "Accept: application/json"
--header "app_key: string"
--header "Authorization: Bearer {accessToken}"
"https://server:port/iocm/app/rule/v1.2.0/rules"
-d "{
  "name": "rule1",
  "author": "wudashan",
  "description": "this is a test rule 2015.12.23",
  "status": "active",
  "logic": "and",
  "conditions": [
    {
      "type": "DEVICE_DATA",
      "deviceInfo": {
        "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
        "path": "DoorWindow/status"
      },
      "operator": "=",
      "value": "OPEN",
      "strategy" :{
          "trigger": "pulse",
```

```
          "eValidTime": "-1"
      }
    }
  ],
  "actions": [
    {
      "type": "DEVICE_CMD",
      "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
      "cmd": {
        "serviceId": "DoorWindow",
        "messageType": "MUTE",
        "messageBody": {
          "force": "true"
        }
      }
    }
  ]
}"
```

异步命令1.3版本创建规则Demo

```
curl -X POST
--header "Content-Type: application/json"
--header "Accept: application/json"
--header "app_key: string"
--header "Authorization: Bearer {accessToken}"
"https://server:port/iocm/app/rule/v1.2.0/rules"
-d "{
  "name": "rule1",
  "author": "wudashan",
  "description": "this is a test rule 2015.12.23",
  "status": "active",
  "logic": "and",
  "conditions": [
    {
      "type": "DEVICE_DATA",
      "deviceInfo": {
        "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
        "path": "DoorWindow/status"
      },
      "operator": "=",
      "value": "OPEN",
      "strategy" :{
          "trigger": "pulse",
          "eValidTime": "-1"
      }
    }
  ],
  "actions": [
    {
      "type": "DEVICE_CMD",
      "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
      "cmd": {
        "serviceId": "DoorWindow",
        "messageType": "MUTE",
        "messageBody": {
          "force": "true"
        }
      },
      "cmdVersion": "V1.3.0"
    }
  ]
}"
```

异步命令1.3版本创建between表达式规则Demo

```
curl -X POST
--header "Content-Type: application/json"
--header "Accept: application/json"
--header "app_key: string"
```

```
--header "Authorization: Bearer {accessToken}"
"https://server:port/iocm/app/rule/v1.2.0/rules"
-d "{
  "name": "rule1",
  "author": "wudashan",
  "description": "this is a test rule 2015.12.23",
  "status": "active",
  "logic": "and",
  "conditions": [
    {
      "type": "DEVICE_DATA",
      "deviceInfo": {
        "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
        "path": "Temperature/temperature"
      },
      "operator": "between",
      "value": "20,30",
       "strategy":{
        "trigger":"pulse",
        "eValidTime":"-1"
      }
    }
  ],
  "actions": [
    {
      "type": "DEVICE_CMD",
      "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
      "cmd": {
        "serviceId": "DoorWindow",
        "messageType": "MUTE",
        "messageBody": {
          "force": "true"
        }
      },
      "cmdVersion": "V1.3.0"
    }
  ]
}"
```

异步命令1.3版本创建geo.circle.in/out表达式规则Demo

```
curl -X POST
--header "Content-Type: application/json"
--header "Accept: application/json"
--header "app_key: string"
--header "Authorization: Bearer {accessToken}"
"https://server:port/iocm/app/rule/v1.2.0/rules"
-d "{
  "name": "rule1",
  "author": "wudashan",
  "description": "this is a test rule 2015.12.23",
  "status": "active",
  "logic": "and",
  "conditions": [
    {
      "type": "DEVICE_DATA",
      "deviceInfo": {
        "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
        "path": "Location/longitude,latitude"
      },
      "operator": "geo.circle.in",
      "value": "114074792,22641851,100",
       "strategy":{
        "trigger":"pulse",
        "eValidTime":"-1"
      }
    }
  ],
  "actions": [
    {
```

```
      "type": "DEVICE_CMD",
      "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
      "cmd": {
        "serviceId": "DoorWindow",
        "messageType": "MUTE",
        "messageBody": {
          "force": "true"
        }
      },
      "cmdVersion": "V1.3.0"
    }
  ]
}"
```

返回结果

```
{
  "ruleId":"*******"
}
```

9.1.7.2 更新规则

接口功能

更新规则

请求方法

PUT

请求路径

https://server:port/iocm/app/rule/v1.2.0/rules

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
app_key	必选	String	header	鉴权用户名，填写创建应用时返回的id。
Authorization	必选	String	header	鉴权信息，填写鉴权接口返回的accessToken。

消息体参数定义查看RuleDTO1.2定义说明。

成功结果

状态码StatusCode：200 OK

内容类型Content-Type：application/json

返回结构

字段	字段类型	描述
ruleId	String	规则实例id

异常返回码

HttpStatus	error_code	error_desc	说明
401	100002	Invalid access token.	错误的token信息
400	101419	RuleId can not be null or empty when update rule. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	ruleId不能为空
401	101409	Authentication failure. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中没有携带appId
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中携带appId不存在
200	101403	The rule is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	规则不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用没有权限

200	101402	The rule name already existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	规则名称已存在
500	100023	The data in dataBase is abnormal. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	获取配置参数数据失败
200	101404	The rule count of each app has reached the upper limit, the current up limit is 30 entries. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用已有规则数量查过规定的最大值
400	100216	The application input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	规则属性传参不合法
500	101424	Internal server error happened when create rule. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	创建规则异常
200	101401	The rule input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	接口传参不合法
400	101416	The action of rule is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	Action参数不合法

400	101418	Action id duplicated in same rule. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	Action id重复
400	101425	Rule api parameter is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	接口传参不合法
500	50204	Convert json to class failed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	数据格式转换失败

消息示例

```
Curl
curl -X PUT
--header "Content-Type: application/json"
--header "Accept: application/json"
--header "app_key: string"
--header "Authorization: Bearer {accessToken}"
"https://server:port/iocm/app/rule/v1.2.0/rules"
-d "{
  "ruleId":"rule1"
  "name": "rule1",
  "author": "wudashan",
  "description": "this is a test rule 2015.12.23",
  "status": "active",
  "logic": "and",
  "conditions": [
    {
      "type": "DEVICE_DATA",
      "deviceInfo": {
        "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
        "path": "DoorWindow/status"
      },
      "operator": "=",
      "value": "OPEN"
    }
  ],
  "actions": [
    {
      "type": "DEVICE_CMD",
      "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
      "cmd": {
        "serviceId": "DoorWindow",
        "messageType": "MUTE",
        "messageBody": {
          "force": "true"
        }
      }
    }
  ]
```

```
}"
```

异步命令1.3版本更新规则Demo

```
curl -X POST
--header "Content-Type: application/json"
--header "Accept: application/json"
--header "app_key: string"
--header "Authorization: Bearer {accessToken}"
"https://server:port/iocm/app/rule/v1.2.0/rules"
-d "{
  "ruleId":"rule1"
  "name": "rule1",
  "author": "wudashan",
  "description": "this is a test rule 2015.12.23",
  "status": "active",
  "logic": "and",
  "conditions": [
    {
      "type": "DEVICE_DATA",
      "deviceInfo": {
        "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
        "path": "DoorWindow/status"
      },
      "operator": "=",
      "value": "OPEN"
    }
  ],
  "actions": [
    {
      "type": "DEVICE_CMD",
      "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
      "cmd": {
        "serviceId": "DoorWindow",
        "messageType": "MUTE",
        "messageBody": {
          "force": "true"
        }
      },
      "cmdVersion": "V1.3.0"
    }
  ]
}"
```

返回结果

```
{
  "ruleId": "********"
}
```

9.1.7.3 修改规则状态

接口功能

修改规则状态

请求方法

PUT

请求路径

https://server:port/iocm/app/rule/v1.2.0/rules/{ruleId}/status/{status}

注意事项

```
携带头域信息
Header：
```

```
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
app_key	必选	String	header	鉴权用户名，填写创建应用时返回的id
Authorization	必选	String	header	鉴权信息，填写鉴权接口返回的accessToken
ruleId	必选	String	path	规则的ID
status	必选	String	path	规则的状态，“active”代表激活状态，“inactive”代表未激活

成功结果

状态码StatusCode：200 OK

异常返回码

HttpStatus	error_code	error_desc	说明
401	100002	Invalid access token.	错误的token信息
401	101409	Authentication failure. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中没有携带appId
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中携带appId不存在
200	101403	The rule is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	规则不存在

200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用没有权限
200	101401	The rule input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	接口传参不合法

消息示例

```
Curl
curl -X POST
--header "Content-Type: application/json"
--header "Accept: application/json"
--header "app_key: string"
--header "Authorization: Bearer {accessToken}"
-d "{
  "status": "******"
}"
"http://server:port/iocm/app/rule/v1.1.0/rules/string"
```

返回结果

NA

9.1.7.4 删除规则

接口功能

删除规则

请求方法

DELETE

请求路径

https://server:port/iocm/app/rule/v1.2.0/rules/{ruleId}

注意事项

```
携带头域信息
Header:
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
app_key	必选	String	header	鉴权用户名，填写创建应用时返回的id
Authorization	必选	String	header	鉴权信息，填写鉴权接口返回的accessToken
ruleId	必选	String	path	规则实例的id

成功结果

状态码StatusCode：204 No Content

异常返回码

HttpStatus	error_code	error_desc	说明
401	100002	Invalid access token.	错误的token信息
401	101409	Authentication failure. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中没有携带appId
200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中携带appId不存在
200	101403	The rule is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	规则不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用没有权限

消息示例

```
Curl
curl -X DELETE
--header "Accept: application/json"
```

```
--header "app_key: string"
--header "Authorization: Bearer {accessToken}"
"http://server:port/iocm/app/rule/v1.1.0/rules/{ruleId}"
```

返回接口

NA

9.1.7.5 查找规则

接口功能

查找规则

请求方法

GET

请求路径

https://server:port/iocm/app/rule/v1.2.0/rules?name={name}&author={author}

注意事项

```
携带头域信息
Header：
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
app_key	必选	String	header	鉴权用户名，填写创建应用时返回的id
Authorization	必选	String	header	鉴权信息，填写鉴权接口返回的accessToken
author	必选	String	query	用户id
name	可选	String	query	规则名称

成功结果

状态码StatusCode：200 OK

返回结果：Rule列表，消息体参数定义查看RuleDTO1.2定义说明。

异常返回码

HttpStatus	error_code	error_desc	说明
401	100002	Invalid access token.	错误的token信息

200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用没有权限

消息示例

```
Curl
curl -X GET
--header "Accept: application/json"
--header "app_key: string"
--header "Authorization: Bearer {accessToken}"
"http://server:port/iocm/app/rule/v1.2.0/rules?name={name}&author={author}
```

消返回结果

```
{
  "name": "****",
  "author": "*****",
  "description": "this is a test rule 2015.12.23",
  "type": "noType",
  "status": "active",
  "logic": "and",
  "conditions": [
    {
      "type": "DeviceData",
      "id": 0,
      "deviceInfo": {
        "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
        "path": "DoorWindow/status"
      },
      "operator": "=",
      "value": "OPEN"
    }
  ],
  "actions": [
    {
      "type": "DeviceCMD",
      "id": 0,
      "appId": "pnS6yhwLszJ8kZF7HhuH7c0eoGQa",
      "deviceId": "0285f8e6-f0f1-441a-bc27-30f02545658c",
      "cmd": {
        "serviceId": "DoorWindow",
        "messageType": "MUTE",
        "messageBody": {
          "force": "true"
        }
      }
    }
  ]
}
```

9.1.7.6 批量修改规则状态

接口功能

批量修改规则状态

请求方法

PUT

请求路径

https://server:port/iocm/app/rule/v1.2.0/rules/status

注意事项

```
携带头域信息
Header：
"app_key:{appId}"
"Authorization:Bearer {accessToken}"
Content-Type:application/json;
```

参数说明

参数	必选/可选	类型	位置	描述
app_key	必选	String	header	鉴权用户名，填写创建应用时返回的id
Authorization	必选	String	header	鉴权信息，填写鉴权接口返回的accessToken
requests	必选	List<RuleStatusUpdateReqDTO>	body	请求结构体列表，RuleUpdateRequestDTO为自定义结构体，见下表

RuleStatusUpdateReqDTO 结构体说明：

参数	必选/可选	类型	描述
ruleId	必选	String	规则Id
status	必选	String	状态，取值可为“active”和“inactive”。

成功结果

状态码

StatusCode：200 OK

内容类型

Content-Type：application/json

返回结果：

参数	类型	描述
result	RuleResultBodyDTO	总体执行结果，自定义结构体，具体结构见下表
responses	List<RuleResponseBodyDTO>	每个条目的执行结果，自定义结构体，具体结构见下表

RuleResultBodyDTO 结构体说明：

参数	类型	描述
result_code	String	总体执行结果代码，若执行成功则为“OK”，否则为“Partial OK”
result_desc	String	总体执行结果描述，若全部执行成功则为“Operation successful.”，否则为“Operation partial ok.”

RuleResponseBodyDTO结构体说明：

参数	类型	描述
ruleId	String	规则Id
response	ExceptionMsg	执行结果，如果成功则不带，否则包含异常信息

异常返回码

HttpStatus	error_code	error_desc	说明
401	100002	Invalid access token.	错误的token信息
401	101409	Authentication failure. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中没有携带appId

200	100203	The application is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	请求头域中携带appId不存在
200	101403	The rule is not existed. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	规则不存在
200	100217	The application hasn't been authorized. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	应用没有权限
200	101401	The rule input is invalid. For more information,please visit the following URL: http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=2161&extra=	接口传参不合法

消息示例

```
Curl
curl -X GET
--header "Accept: application/json"
--header "app_key: string"
--header "Authorization: Bearer {accessToken}"
"https://server:port/iocm/app/rule/v1.2.0/rules/status"
-d "{
    "requests":[{
 "ruleId":"******ruleid_1",
 "status":"active"
 },{
 "ruleId":"******ruleid_2",
 "status":"inactive"
 }
 ]
}"
```

返回结果

```
{
  "result": {
 "result_code":"Partial OK",
 "result_desc":"Operation partial ok"
```

```
  },
  "responses": [
    {
     "ruleId":"******ruleid_1",
      "response": {
        "error_code": "*********",
        "error_desc": "*********"
      }
    },{
     "ruleId":"******ruleid_2",
      "response": {
        "error_code": "*********",
        "error_desc": "*********"
      }
    }
  ]
}
```

9.1.8 消息推送

Application向平台订阅设备变更通知（**9.1.3.3 Application 订阅平台数据**），当设备发生变更时，平台向Application推送此消息（Application启动初始化订阅），Application根据通知类型进行对消息分派对应的服务模块进行处理。

Application订阅接口：参考**9.1.3.3 Application 订阅平台数据**。

9.1.8.1 注册直连设备的通知

接口功能

NA调用IoT平台的接口添加新设备，获取设备验证码（参见**9.1.2.1 注册直连设备**），IoT平台使用该接口功能将该信息通知给NA，平台通知给Application。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“deviceAdded”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

注意事项

参数说明

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型：deviceAdded
deviceId	必选	string	设备唯一标识
resultCode	必选	string	取值：expired过期，不成功； succeeded成功
deviceInfo	可选	DeviceInfo	参见**9.2.1 DeviceInfo结构体说明**

消息示例

Body

{

"deviceId":"74febc35-805f-4c43-9189-649d5d7fff77",

"resultCode":"succeeded",

"deviceInfo":{

"name":"Sensor_12",

"manufacturer":"wulian",

"deviceType":90",

"model":"90",

"mac":"C7EA1904004B1204",

"swVersion":"…",

"fwVersion":"…",

"hwVersion":"…",

"protocolType":"zigbee",

"description":"smock detector",

"nodeType":"GATEWAY",

}

}

9.1.8.2 发现非直连设备的通知

接口功能

IoT平台接收到网关添加新设备，IoT平台使用该接口功能将该信息通知给Application。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“deviceAdded”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明：

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，deviceAdded（添加设备）

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识
gatewayId	必选	String	网关唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
nodeType	必选	String	指定节点的类型 ENDPOINT: 传感器. GATEWAY: 网关.
deviceInfo	必选	DeviceInfo	设备信息，设备信息变化、添加、删除设备时会带此参数，定义参见**9.2.1 DeviceInfo结构体说明**

添加新设备：

Body

{

"notifyType ": "deviceAdded",

"deviceId":"7b3979fc-b072-433b-b3f6-673072e1bc04",

"gatewayId":"**********",

"deviceInfo":{

"name":"Sensor_12",

"manufacturer":"wulian",

"type":90",

"model":"90",

"mac":"C7EA1904004B1204",

"swVersion":"…",

"fwVersion":"…",

"hwVersion":"…",

"protocolType":"zigbee",

"description":"smock detector"

}

}

9.1.8.3 设备信息变化的通知

接口功能

IoT平台接收到设备信息变化（静态信息，如设备名称、厂商id），IoT平台使用该接口功能将该信息通知给Application。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“deviceInfoChanged”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明：

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型， deviceInfoChanged（设备信息变化）
deviceId	必选	String	设备唯一标识
gatewayId	必选	String	网关唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
deviceInfo	必选	DeviceInfo	设备信息，设备信息变化、添加、删除设备时会带此参数，定义参见**9.2.1 DeviceInfo结构体说明**

设备信息变化：

Body

{

"notifyType ": "deviceInfoChanged",

"deviceId":"7b3979fc-b072-433b-b3f6-673072e1bc04",

"gatewayId":"**********",

"deviceInfo":{

}

}

9.1.8.4 设备数据变化的通知

接口功能

IoT平台接收到设备数据变化（动态变化，如，服务属性值的变化），IoT平台使用该接口功能将该信息通知给Application。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“deviceDataChanged”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明：

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，deviceDataChanged（设备数据变化）
requestId	可选	String	关联消息的Id
deviceId	必选	String	设备唯一标识
gatewayId	可选	String	网关唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
service	必选	DeviceServiceData	设备的service数据，只有设备数据变化时会带此参数，定义参见下表

DeviceServiceData

字段	必选/可选	类型	描述
serviceId	必选	String	服务标识
serviceType	可选	String	服务类型
data	可选	ObjectNode	服务属性信息
eventTime	可选	String	事件产生时间,时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z

消息事例

Body

{

" notifyType ": "deviceDataChanged",

"requestId": "{requestId}",

"timestamp":"{timestamp}",

"deviceId":"7b3979fc-b072-433b-b3f6-673072e1bc04",

"gatewayId":"**********",

"service": {

"serviceType": "",

"serviceId": "",

"data"： {"battery_low"：1},

"eventTime":"{eventTime}"

}

}

9.1.8.5 删除非直连设备的通知

接口功能

IoT平台接收到删除非直连设备，IoT平台使用该接口功能将该信息通知给Application。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“deviceDeleted”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明：

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，deviceDeleted（删除设备）
deviceId	必选	String	设备唯一标识
gatewayId	必选	String	网关唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。

删除设备：

Body

{

"notifyType ": "deviceDeleted",

"deviceId":"7b3979fc-b072-433b-b3f6-673072e1bc04",

"gatewayId":"**********",

}

9.1.8.6 消息确认通知

接口功能

IoT平台接收到网关的确认消息（如下发命令给网关，网关确认接收到该消息），将该消息通知给Application。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“messageConfirm”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，messageConfirm (消息确认)
header	必选	Header	见header字段说明
body	必选	ObjectNode	根据业务具体定义，确认消息可以携带状态变化等消息

Header**字段说明：**

字段	必选/可选	类型	描述
requestId	必选	String	消息的序列号，唯一标识该消息，1-128字节

字段	必选/可选	类型	描述
from	必选	String	表示消息发布者的地址： 网关或设备发起的请求：/devices/{deviceId} 设备服务发起的请求：/devices/{deviceId}/services/{serviceId} 1-128字节
to	必选	String	表示消息接收者的地址，1-128字节 To就是请求中的From，如APP的userId
status	必选	String	命令状态：1-32字节 sent --- 已发送 delivered --- 已送达 executed --- 已执行
timestamp	必选	String	时间格式：yyyyMMdd’T’HHmmss’Z’ 如： 20151212T121212Z 1-32字节

网关确认消息通知

Body

{

"notifyType"： "messageConfirm",

"header":

{

"requestId":"{ requestId }",

"from": "{from}",

"to": "{to}",

"status":"delivered",

"timestamp":"{timestamp}"

}

"body":

{

JSONBody

}

}

9.1.8.7 设备响应命令通知

接口功能

IoT平台接收到设备（包括网关设备和普通设备）的响应命令（如下发命令给设备，设备执行命令后，发送响应命令给平台，如视频呼叫、视频录制，抓图等），将该消息通知给Application。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“commandRsp”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

注意事项

commandRsp对应的command是指使用设备服务接口下发的命令。请参考**9.1.5 设备服务调用**。

参数说明

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，commandRsp (响应命令)
header	必选	Header	见Header字段说明
body	必选	ObjectNode	响应命令的消息体Body中的内容

Header

字段	必选/可选	类型	描述
requestId	必选	String	消息的序列号，唯一标识该消息，1-128个字节
from	必选	String	1-128个字节，表示消息发布者的地址： 网关或设备发起的请求：/devices/{deviceId} 设备服务发起的请求：/devices/{deviceId}/services/{serviceId}

字段	必选/可选	类型	描述
to	必选	String	表示消息接收者的地址，1-128个字节 To就是请求中的From，如APP的userId
timestamp	可选	String	时间格式：yyyyMMdd’T’HHmmss’Z’如：20151212T121212Z，1-32个字节
deviceId	必选	String	设备唯一标识（命令所属设备的ID）
serviceType	必选	String	命令所属服务类型
method	必选	String	(1-32个字节)命令名称，存在于设备服务能力下的命令名称字段，如："DISCOVERY" "REMOVE" 。详情参考profile全量模板

设备命令响应通知

Body

```
{
"notifyType":"commandRsp",
"header":{
"requestId":"{requestId}",
"from":"/devices/{deviceId}/services/{serviceId}",
"to":"/apps/{appId}/users/{userId}",
"deviceId":"{deviceId}",
"serviceType":"{serviceType}",
"method":"{commandRsp}",
},
"body":
{
...
}
}
```

9.1.8.8 设备事件通知

接口功能

IoT平台接收到设备发出的事件（如设备UDS存储空间不足事件），将该消息通知给Application。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“deviceEvent”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，deviceEvent (设备事件)
header	必选	Header	见Header字段说明
body	必选	ObjectNode	响应命令的消息体Body中的内容

Header

字段	必选/可选	类型	描述
eventType	必选	String	事件类型，1-32个字节
from	必选	String	1-128个字节，表示消息发布者的地址，设备服务发起的请求：/devices/{deviceId}/services/{serviceId}
to	可选	String	1-128个字节，当catogery为“event”时请求地址： 上报设备服务事件：/event/v1.1.0/devices/{deviceId}/services/{serviceId}
timestamp	必选	String	时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z，1-32个字节
eventTime	必选	String	时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z，1-32个字节
deviceId	必选	String	设备唯一标识（命令所属设备的ID）
serviceType	必选	String	命令所属服务类型

消息事例

Body

{

"notifyType":"deviceEvent",

"header":{

"eventType":"{eventType}",

"from":"/devices/{deviceId}/services/{serviceId}",

"to":"/event/v1.1.0/devices/{deviceId}/services/{serviceId}",

"deviceId":"{deviceId}",

"serviceType":"{serviceType}",

"timestamp":"{timestamp}",

eventTime":"{eventTime}",

},

"body":

{

usedPercent":80

}

}

9.1.8.9 设备服务上报通知

接口功能

IoT平台接收到设备服务信息变化，IoT平台使用该接口功能将该信息通知给Application。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“serviceInfoChanged”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明:

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，serviceInfoChanged（设备信息变化）

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识
gatewayId	必选	String	网关唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
serviceId	必选	String	设备服务标识
serviceType	必选	String	设备服务类型
serviceInfo	必选	ServiceInfo	设备服务信息（增量上报），见下表

ServiceInfo

字段	必选/可选	类型	描述
muteCmds	必选	List<String>	隐私模式的命令："VIDEO_RECORD", "INVITE-INIT", "INVITE", "IMAGE_FIXED_CAPTURE"

消息事例

Body

{

"notifyType":"serviceInfoChanged",

"deviceId":"7b3979fc-b072-433b-b3f6-673072e1bc04",

"gatewayId":"**********",

"serviceType":"{ serviceType}",

"serviceId": "{serviceId}",

"serviceInfo":

{

"muteCmds":"VIDEO_RECORD"

}

}

9.1.8.10 规则事件上报通知

接口功能

IoT平台规则触发后生成对应的规则事件通知NA。

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“ruleEvent”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明:

字段	必选/可选	类型	描述
notifyType	必选	String	"ruleEvent"
author	必选	String	同规则接口中的author
ruleId	必选	String	规则id
ruleName	必选	String	规则名称
logic	可选	String	多条件逻辑关系
reasons	必选	List<Reason>	触发原因，对应conditions
triggerTime	必选	String	规则触发时间
actionsResults	必选	List<ActionResult>	动作执行结果

Reason 字段说明：

字段	必选/可选	类型	描述
satisfactionTime	必选	String	条件满足时间
type	必选	String	条件类型，包括DEVICE_DATA，DailyTimer，CycleTimer三种
id	可选	String	对应condition的id
info	可选	Json	不同条件类型携带不同信息
transInfo	可选	Json	事件推送时的回填信息，对应规则的condition中的transInfo

DEVICE_DATA类型的条件携带的info：

字段	必选/可选	类型	描述
realValue	可选	String	设备数据的实际值

ActionResult 字段说明：

字段	必选/可选	类型	描述
type	必选	String	action类型
id	可选	String	对应action的id
exception	与result二选一	String	规则引擎构造action对应的请求过程中遇到异常时携带的异常信息
result	与exception二选一	Json	动作的执行结果，对于DEVICE_CMD/SMS/EMAIL类型的动作，内容为statusCode +body，详细信息参考接口文档中对应接口的说明
info	可选	Json	不同动作类型携带不同信息
transInfo	可选	Json	事件推送时的回填信息，对应规则的action中的transInfo

DEVICE_CMD类型的动作携带的info

字段	必选/可选	类型	描述
requestId	必选	String	下发异步命令时，为请求消息分配的UUID

消息事例

Body

{ "notifyType": "ruleEvent",

"author": "*****",

"ruleId": "**********",

"ruleName": "name",

"reasons": [{

"satisfactionTime": " yyyyMMddTHHmmssZ ",

"type": "******"

}

],

"triggerTime": "yyyyMMddTHHmmssZ",

"actionsResults": [{

"type": "******",

"exception": "******"

}

]

}

9.1.8.11 绑定设备通知

接口功能

即设备初始登陆通知，网关设备在平台注册绑定（网关设备在平台创建对应设备的信息并获取密码的过程），平台通知给NA。

调用方法

POST

接口路径

{callbackUrl}

此处的接口路径是应用向平台订阅数据时注册的对应notifyType为“bindDevice”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，取值：bindDevice
deviceId	必选	string	设备Id
resultCode	必选	string	绑定结果，取值：expired/succeeded
deviceInfo	可选	DeviceInfo	设备信息，参见**9.2.1 DeviceInfo结构体说明**

返回结果

无

消息示例

Method：POST

request：

Body

{
"notifyType": "bindDevice",
"deviceId": "74febc35-805f-4c43-9189-649d5d7fff77",
"resultCode": "succeeded",
"deviceInfo": {
"name": "Sensor_12",
"manufacturer": "wulian",
"deviceType": 90,
"model": "90",
"mac": "C7EA1904004B1204",
"swVersion": "…",
"fwVersion": "…",
"hwVersion": "…",
"protocolType": "zigbee",
"description": "smockdetector",
"nodeType": "GATEWAY"
}
}

9.1.8.12 批量设备数据变化上报通知

接口功能

批量设备数据变化后消息通知

调用方法

POST

接口路径

{callbackUrl}

此处的**接口路径**是应用向平台订阅数据时注册的对应notifyType为“deviceDatasChanged”的回调通知地址，请参考**9.1.3.3 Application 订阅平台数据**。

参数说明

字段	必选/可选	类型	描述
notifyType	必选	String	deviceDatasChanged
requestId	必选	String	标识命令消息
deviceId	必选	String	设备唯一标识
gatewayId	必选	String	网关唯一标识
services	必选	List<DeviceServiceData>	服务

DeviceServiceData结构：

字段	必选/可选	类型	描述
serviceId	必选	String	服务ID
serviceType	必选	String	服务类型
data	必选	ObjectNode	数据
eventTime	必选	String	时间格式 yyyymmddThhmissZ , 例如, 20151212T121212Z.

返回结果

无

消息示例

Body

{

"notifyType": "deviceDatasChanged",

"requestId": "**********",

"deviceId": "7b3979fc-b072-433b-b3f6-673072e1bc04",

"gatewayId": "**********",

"service": {

"serviceId": "Battery",
"serviceType": "Battery",
"data": {},
"eventTime": "20170101120000"
}
}

9.2 常用数据结构体定义

9.2.1 DeviceInfo 结构体说明

字段	必选/可选	类型	描述
nodeId	必选	String	设备唯一标识
name	可选	String	设备名称
description	可选	String	设备描述
manufactureId	必选	String	供应商标识
manufacturerName	可选	String	供应商名字
mac	可选	String	物理地址
location	可选	String	设备位置
deviceType	必选	String	设备种类： MultiSensor ContactSensor Camera Gateway
model	必选	String	设备型号. 如果是Z-Wave协议，型号就是ProductType + ProductId（用零填充，如果需要） 格式的十六进制数,例如 001A-0A12. 如果是其他协议，型号需要另一种格式表示
swVersion	可选	String	使用Z-Wave协议时的软件版本号。 其格式是 major version number.minor version number, 例如, 1.1.
fwVersion	可选	String	固件版本号
hwVersion	可选	String	硬件版本号
protocolType	必选	String	协议类型 (Z-Wave).

字段	必选/可选	类型	描述
signalStrength	可选	String	信号强度
bridgeId	可选	String	网桥标识
supportedSecurity	可选	String	是否支持安全模式，其取值为TRUE或FALSE
isSecurity	可选	String	设备是否安全，在安全模式为TRUE的情况下有效，其取值为TRUE或FALSE
sigVersion	可选	String	信令版本
runningStatus	可选	String	设备运行状态： NORMAL：正常 ABNORMAL：异常 FAULT：故障
status	可选	String	设备状态. ONLINE: 在线 OFFLINE: 下线 INBOX: 停用.
statusDetail	可选	String	设备状态的详细情况
mute	可选	String	设备是否进行屏蔽 TRUE: 被屏蔽. FALSE: 没被屏蔽.

9.2.2 DeviceService 结构体说明

字段	必选/可选	类型	描述
serviceType	必选	String	服务类型
serviceId	必选	String	服务ID标识
data	必选	JSON String	指定一个属性值对(AVP).
eventTime	必选	DateTime	时间格式 yyyymmddThhmissZ , 例如, 20151212T121212Z.
serviceInfo	可选	ServiceInfo	业务信息，此字段用于内部控制功能，不用关心此字段

9.2.3 QueryDeviceDTOCloudNA 结构体说明

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识
gatewayId	必选	String	网关唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
nodeType	必选	String	指定节点的类型 ENDPOINT: 传感器. GATEWAY: 网关.
createTime	必选	String	设备注册时间
lastModifiedTime	必选	String	设备信息最后修改时间
deviceInfo	必选	DeviceInfo	详见"9.2.1 DeviceInfo结构体说明"
services	可选	List<DeviceService>	详见 "9.2.2 DeviceService结构体说明"

9.2.4 ServiceCommand 结构体说明

字段	必选/可选	类型	描述
commandName	可选	String	命令
paras	可选	List<ServiceCommandPara>	属性列表，参考下表

ServiceCommandPara结构体

字段	必选/可选	类型	描述
paraName	可选	String	参数名
dataType	可选	String	数据类型
required	可选	boolean	是否必选
min	可选	int	最小
max	可选	int	最大

字段	必选/可选	类型	描述
step	可选	Double	步长
maxLength	可选	int	最大长度
unit	可选	String	单位（符号）
enumList	可选	List<String>	枚举类型列表

9.2.5 ServiceProperty 结构体说明

字段	必选/可选	类型	描述
propertyName	可选	String	属性名
dataType	可选	String	数据类型
required	可选	boolean	是否必选
min	可选	int	最小
max	可选	int	最大
step	可选	double	步长
maxLength	可选	int	最大长度
method	可选	String	访问方法：RW 可读R，可写W
unit	可选	String	单位（符号）
enumList	可选	List<String>	枚举类型列表

9.2.6 DeviceCommandResp 结构体说明

字段	必选/可选	类型	描述
commandId	必选	String(1-64)	设备命令ID
appId	必选	String(1-64)	设备命令撤销任务所属的应用ID
deviceId	必选	String(1-64)	设备命令指定的设备ID
command	必选	CommandDTOV4	下发命令的信息，定义参见附录**9.2.7 CommandDTOV4 结构体说明**

callbackUrl	可选	String(1024)	命令状态变化通知地址，必须使用HTTPS信道回调地址 (说明：HTTP信道只可用于调测，不能用于商用环境)； 当命令状态变化时(执行失败，执行成功，超时，发送，已送达)会通知NA，平台会以POST方式发送HTTP消息给应用，请求Body为json字符串，格式形如： {"deviceId":"deviceId","commandId":"commandId","result": {"status":"SUCCESS","result":{…}}}
expireTime	可选	Integer(>=0)	下发命令有效的超期时间，单位为秒，表示设备命令在创建后expireTime秒内有效，超过这个时间范围后命令将不再下发，如果未设置则默认为48小时
status	必选	String	设备命令的状态，DEFAULT表示未下发，EXPIRED表示命令已经过期，SUCCESSFUL表示命令已经成功执行，FAILED表示命令执行失败，TIMEOUT表示命令下发执行超时，CANCELED表示命令已经被撤销执行
result	可选	ObjectNode	设备命令执行的详细结果，格式 为json字符串
creationTime	必选	String(20)	设备命令的创建时间
executeTime	可选	String(20)	设备命令执行的时间
platformIssuedTime	可选	String(20)	平台发送设备命令的时间
deliveredTime	可选	String(20)	平台将设备命令送达到设备的时间
issuedTimes	可选	Integer(>=0)	平台发送设备命令的次数

9.2.7 CommandDTOV4 结构体说明

字段	必选/可选	类型	描述
serviceId	必选	String(1-64)	命令对应的服务ID，不能为空字符串
method	必选	String(1-128)	命令服务下具体的命令名称，服务属性等，不能为空字符串

paras	可选	ObjectNode	命令参数的jsonString，具体格式需要应用和设备约定。

9.2.8 GetDeviceRspDTO 结构体说明

字段	类型	描述
deviceId	String	设备唯一标识，1-64个字节
gatewayId	String	设备唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
nodeType	String	指定节点的类型 ENDPOINT: 传感器. GATEWAY: 网关.
createTime	String	设备注册时间；时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z
lastModifiedTime	String	最新修改时间；时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z
deviceInfo	DeviceInfo	DeviceInfo的定义参见**9.2.1 DeviceInfo结构体说明**
services	List<DeviceService>	DeviceService的定义参见**9.2.2 DeviceService结构体说明**
connectionInfo	ConnectionInfo	连接信息
location	Location	位置信息
devGroupIds	List<String>	设备组Id标签列表

9.2.9 CommandNA2CloudHeader 结构体说明

字段	必选/可选	类型	描述
requestId	可选	String	如果填写该requestId，用填写的requestId标识一个命令，如果不填写就由平台生成。1-32个字节
mode	必选	String	是否要确认消息，不需要确认消息：“NOACK”，需要确认消息：“ACK”，其它值无效。

字段	必选/可选	类型	描述
from	可选	String	(1-128个字节)表示消息发布者的地址： Application发起的请求：/{serviceName}
to	可选	String	接收者地址
toType	可选	String	取值范围：CLOUD|GATEWAY
method	必选	String	发现命令名称："DISCOVERY" 删除命令名称："REMOVE"
callbackURL	可选	String	回调的url地址。必须使用HTTPS信道回调地址，同时回调地址中必须指定回调地址的端口。(说明：HTTP信道只可用于调测)

9.2.10 ApplicationSetEncrtptDTO 结构体说明

字段	必选/可选	类型	描述
flag	必选	Integer	加密标志，取值0表示不加密；取值2表示AES加密，key和iv必填
iv	可选	String(16)	iv 和 key 共同决定加密方式
key	必选	String(16)	iv 和 key 共同决定加密方式

9.2.11 PutCarInfoData 结构体说明

字段	必选/可选	类型	描述
engineModel	可选	String(256)	发动机型号
vin	可选	String(256)	VIN
plateNumber	可选	String(256)	车牌号
simNumber	可选	String(256)	SIM卡号(注：首次创建车辆信息时必须携带simNumber；修改车辆信息时可选)
carModel	可选	String(256)	车辆型号(注：首次创建车辆信息时必须携带carModel，修改车辆信息时可选)
terminalID	可选	String(256)	终端ID
iccid	可选	String(256)	ICCID
imei	可选	String(256)	IMEI

字段	必选/可选	类型	描述
encryptFlag	可选	Integer	加密标志，取值0表示不加密，取值2表示加密
encryptIv	可选	String(256)	encryptIv 和 encryptKey 共同决定加密方式
encryptKey	可选	String(256)	encryptIv 和 encryptKey 共同决定加密方式

10 Platform SDK API 参考(北向)

10.1 接口列表

注意:北向提供的SDK API 是平台与Application之间的http接口的封装。

调用过程中，都要先初始化上下文调用信息：

```
String APP_ID = "ZpM****kca";
String APP_SECRET = "1Om****g0a";
ConfigService.init("https", "*.*.*.*", "8743"); //server info
// init ssl info        UrlConfiguration.initSSL(SSLUtil.SSL_TWO_AUTH_WITH_NO_DNS_CHECK,
                "/***", "***", "/***", "***");
// init login
UrlConfiguration.setIotIp(ConfigService.getIotServerIp());
                        UrlConfiguration.setIotPort(ConfigService.getIotServerPort());
UrlConfiguration.setProtocolType(ConfigService.getProtocolType());
        AuthHttpClient.getInstance().setKey(APP_ID);
AuthHttpClient.getInstance().setSecret(APP_SECRET);
```

其中，初始化SSL信息中（若不需要使用制定的证书认证，不需调用SSL初始化），参数说明如下：

参数	必选/可选	类型	描述
sslType	必选	int	SSL检查的类型，当前有两个值： SSLUtil.SSL_TWO_AUTH_WITH_DNS_CHECK 双向认证检查DNS认证 SSLUtil.SSL_TWO_AUTH_WITH_NO_DNS_CHECK 双向认证不检查DNS认证

参数	必选/可选	类型	描述
caServerFilePath	必选	String	服务器jks证书地址（相对于当前项目CLASSPATH的地址），需包含在开发的项目中
caServerFilePWD	必选	String	服务器jks证书密码
caClientFilePath	必选	String	客户端pkcs12证书地址（相对于当前项目CLASSPATH的地址），需包含在开发的项目中
caClientFilePWD	必选	String	客户端pkcs12证书密码

应用开发语言若是JAVA请使用JDK1.8。

应用和IoT平台之间请使用安全传输协议TLS 1.1/1.2。

10.1.1 应用安全接入

Application携带在IoT平台产生的app ID 和 密码过来，调用鉴权接口，获取鉴权token。请参考本文档 2.1.1鉴权章节进行开发。

SDK中封装类：com.huawei.iom.sdk.api.service.token.AccessTokenService

10.1.1.1 Auth(鉴权)

接口功能

实现第三方系统在访问开放API之前的认证。

调用方法

public static AppLoginDTOCloud2NA loginFromNA(AppLoginDTONA2Cloud appLoginDTONA2Cloud) throws BaseException

注意事项

鉴权接口是调用其他API的前提，北向接口除了鉴权接口（Auth），其他接口调用都需要在request header中携带参数app_key和Authorization:Bearer {accessToken}。app_key为参数中的appId，Authorization中{accessToken}的值即为调用Auth接口获取到的accessToken。

参数说明

参数	必选/可选	类型	描述
appLoginDTONA2Cloud	必选	AppLoginDTONA2Cloud	包含北向登录信息的对象

AppLoginDTONA2Cloud对象：

参数	必选/可选	类型	描述
appId	必选	String	用户名，填写应用程序ID
secret	必选	String	登录用户口令，填写应用程序密码

返回结果

字段	必选/可选	类型	描述
AppLoginDTOCloud2NA	必选	AppLoginDTOCloud2NA	返回北向的认证后对象

AppLoginDTOCloud2NA对象：

字段	必选/可选	类型	描述
scope	必选	String	范围，默认值default
tokenType	必选	String	鉴权token类型，默认值bearer
expiresIn	必选	integer	平台生成并返回accessToken的有效时间，单位秒
accessToken	必选	String	Oauth 2.0 鉴权参数
refreshToken	必选	String	刷新Token，用于刷新已有token

调用示例

```
String APP_ID = "ZpM****kca";
String APP_SECRET = "1Om****g0a";
ConfigService.init("https", "*.*.*.*", "8743");
AppLoginDTONA2Cloud appLoginDTONA2Cloud = new AppLoginDTONA2Cloud();
appLoginDTONA2Cloud.setAppId(APP_ID);
appLoginDTONA2Cloud.setSecret(APP_SECRET);
// login
AppLoginDTOCloud2NA appLoginDTOCloud2NA = AccessTokenService.loginFromNA(appLoginDTONA2Cloud);
System.out.println("login response: " + appLoginDTOCloud2NA);
```

输出：

```
login response: AppLoginDTOCloud2NA [accessToken:7d8****fec, tokenType:bearer, scope:[read,
write], expiresIn:43199, refreshToken:a3e****b28]
```

10.1.1.2 Refresh Token(刷新 token)

接口功能

accessToken快过期时，第三方系统通过调用此接口，重新获取可用token。

调用方法

public static AppLoginDTOCloud2NA refreshTokenFromNA(AppRefreshTokenDTONA2Cloud appRefreshTokenDTONA2Cloud) throws BaseException

注意事项

参数说明

参数	必选/可选	类型	描述
appRefreshTokenDTONA2Cloud	必选	AppRefreshTokenDTONA2Cloud	刷新token的请求对象

AppRefreshTokenDTONA2Cloud对象：

参数	必选/可选	类型	描述
appId	必选	String	用户名，填写应用程序ID
secret	必选	String	登录用户口令，填写应用程序密码
refreshToken	必选	String	鉴权时获取到的refreshToken

返回结果

字段	必选/可选	类型	描述
AppLoginDTOCloud2NA	必选	AppLoginDTOCloud2NA	返回北向的认证后对象

AppLoginDTOCloud2NA对象：

字段	必选/可选	类型	描述
scope	必选	String	范围，默认值default
tokenType	必选	String	鉴权token类型，默认值bearer
expiresIn	必选	integer	平台生成并返回accessToken的有效时间，单位秒

字段	必选/可选	类型	描述
accessToken	必选	String	Oauth 2.0 鉴权参数
refreshToken	必选	String	刷新Token，会生成个新的

调用示例

```
String APP_ID = "ZpM****kca";
String APP_SECRET = "1Om****g0a";
ConfigService.init("https", "*.*.*.*", "8743");
        AppLoginDTONA2Cloud appLoginDTONA2Cloud = new AppLoginDTONA2Cloud();
        appLoginDTONA2Cloud.setAppId(APP_ID);
        appLoginDTONA2Cloud.setSecret(APP_SECRET);
        System.out.println("login request: " + appLoginDTONA2Cloud);
        // login
        AppLoginDTOCloud2NA appLoginDTOCloud2NA =
AccessTokenService.loginFromNA(appLoginDTONA2Cloud);
        // refresh token
        AppRefreshTokenDTONA2Cloud appRefreshTokenDTONA2Cloud = new AppRefreshTokenDTONA2Cloud();
        appRefreshTokenDTONA2Cloud.setAppId(APP_ID);
        appRefreshTokenDTONA2Cloud.setSecret(APP_SECRET);
        appRefreshTokenDTONA2Cloud.setRefreshToken(appLoginDTOCloud2NA.getRefreshToken());
        appLoginDTOCloud2NA = AccessTokenService.refreshTokenFromNA(appRefreshTokenDTONA2Cloud);
        System.out.println("refresh token response: " + appLoginDTOCloud2NA);
```

输出：

```
refresh token response: AppLoginDTOCloud2NA [accessToken:dab****20b, tokenType:bearer, scope:
[read, write], expiresIn:43199, refreshToken:c50****b9d]
```

10.1.2 设备管理

设备管理提供了Application申请设备的增，删，改，查接口。

提供修改设备或者设备所属传感器基本信息的功能。

app向IoT平台申请新的设备,IoT平台分配对应的设备验证码，待设备携带验证码请求接入平台后，分配其id,密码，允许其使用。平台提供了增，删，改，查接口，实现对新增设备接入灵活操作。对设备的管理请参考本文档进行开发。

SDK中封装类：com.huawei.iom.sdk.api.service.device.DeviceService

10.1.2.1 注册直连设备

接口功能

应用程序添加设备，获取设备的验证码，并在设备访问南向接口时携带验证码，获取设备唯一标识和密码。

调用方法

public static AddDeviceDTOCloud2NA registerDevice(String verifyCode, String nodeId)

```
        throws IOException, HttpResponseException, BaseException
```

参数说明

字段	必选/可选	类型	描述
verifyCode	可选	String	客户端给出verifyCode则返回的就是这个verifyCode，即使用客户端给出的verifyCode。如果不指定，系统自动生成。
nodeId	必选	String	设备唯一标识, 如：MAC或SIM卡号或设备esn号等。与设备对接时，必须与设备上报的nodeId一致。

返回结果

字段	必选/可选	类型	描述
AddDeviceDTOCloud2NA	必选	AddDeviceDTOCloud2NA	注册设备时的返回对象，内部属性值为null则注册失败

AddDeviceDTOCloud2NA对象：

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
verifyCode	必选	String	申请的临时验证码，设备可以通过验证码获取id和密码
timeout	必选	Integer	验证码有效时间，单位秒，设备需要在有效时间内接入平台
psk	必选	String	psk码，用于生成设备鉴权参数

调用示例

```
// set nodeId
                        String nodeId = "****"
                        // register device
                        AddDeviceDTOCloud2NA addDeviceDTOCloud2NA =
DeviceService.registerDevice(nodeId, nodeId);
```

10.1.2.2 发现非直连设备

接口功能

Application发送DISCOVERY命令给网关下的设备，实质是设备服务调用。

此接口应用场景一般用于通过网关添加传感器，若没有网关，该接口无效，请勿使用。

调用方法

```
public static CommandDTODevice2CloudReply sendDeviceCommand(String deviceId, String serviceId,
            CommandDTONA2Cloud request)
            throws IOException, HttpRequestException, HttpResponseException, BaseException
```

URL参数说明：

字段	必选/可选	类型	描述
deviceId	必选	String	网关设备唯一标识，1-64个字节
serviceId	必选	String	取值"Discovery"
request	必选	CommandDTONA2Cloud	请求的命令信息

CommandDTONA2Cloud对象：

字段	必选/可选	类型	描述
header	必选	CommandNA2CloudHeader	见CommandNA2CloudHeader对象说明
body	必选	Map	消息的消息体

CommandNA2CloudHeader对象：

字段	必选/可选	类型	描述
requestId	可选	String	如果填写该requestId，就是用填写的requestId标识一个命令，如果不填写就由平台生成。1-32个字节
mode	必选	String	是否要确认消息，不需要确认消息：”NOACK”，需要确认消息：”ACK”，其它值无效。
from	必选	String	(1-128个字节)表示消息发布者的地址： Application发起的请求：/{serviceName}
method	必选	String	发现命令名称： "DISCOVERY"
callbackURL	可选	String	命令返回的回调地址。回调地址中必须指定回调地址的端口。

返回结果：

字段	必选/可选	类型	描述
CommandDTODevice2CloudReply	必选	CommandDTODevice2CloudReply	返回对象

CommandDTODevice2CloudReply对象：

字段	必选/可选	类型	描述
status	必选	String	命令状态：1-32字节 sent --- 已发送 delivered --- 已送达 executed --- 已执行
timestamp	必选	String	发送命令的时间戳，1-32字节 时间格式： yyyyMMdd’T’HHmmss’Z’ 如： 20151212T121212Z
requestId	必选	String	1-32个字节，平台分配的序列号，标识一个命令 需要requestId关联对应命令执行结果

调用示例

```
        // info
        String deviceId = "5df****afc";
        String serviceId = "RawData";
        CommandDTONA2Cloud commandDTONA2Cloud = new CommandDTONA2Cloud();
        CommandNA2CloudHeader header = new CommandNA2CloudHeader();
        header.setMode("ACK");
        header.setFrom("/users/***");
        header.setMethod("rawData");
        header.setCallbackURL("****");
        Map<String, String> body = new HashMap<String, String>();
        body.put("**", "***");
        commandDTONA2Cloud.setHeader(header);
        commandDTONA2Cloud.setBody(body);
        // send command
        CommandDTODevice2CloudReply reply = DeviceService.sendDeviceCommand(deviceId, serviceId,
commandDTONA2Cloud);
```

10.1.2.3 查询设备激活状态

接口功能

给定网关设备的设备唯一标识查询网关设备激活状态。

调用方法

public static QueryDeviceRegStatusDTOCloud2NA queryDeviceActiveStatus(String deviceId)

```
            throws IOException, HttpRequestException, HttpResponseException, URISyntaxException,
BaseException
```

参数说明

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节

返回结果

字段	必选/可选	类型	描述
QueryDeviceRegStatusDTOCloud2NA	必选	QueryDeviceRegStatusDTOCloud2NA	返回对象

QueryDeviceRegStatusDTOCloud2NA对象：

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
activated	必选	boolean	网关是否通过验证码获取密码的状态标识
name	必选	String	设备名称

调用示例

```
// device info
String deviceId = "7f7d****e67";
// query device status info
QueryDeviceRegStatusDTOCloud2NA queryDeviceRegStatusDTOCloud2NA =
DeviceService.queryDeviceActiveStatus(deviceId);
```

10.1.2.4 删除设备

接口功能

用来删除设备.

调用方法

```
public static void deleteDevice(String deviceId)
            throws IOException, HttpRequestException, HttpResponseException, URISyntaxException,
BaseException
```

参数说明

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节

返回结果

无，失败时抛出异常.

调用示例

```
// device info
String deviceId = "7f7d****e67";
// delete device info
DeviceService.deleteDevice(deviceId);
```

10.1.2.5 修改设备信息

接口功能

修改设备信息的接口，目前用于设备改名，预留修改设备信息的能力。

调用方法

```
public static void renameDevice(String deviceId, UpdateDeviceInfoDTONA2Cloud request)
            throws IOException, HttpRequestException, HttpResponseException, URISyntaxException,
BaseException
```

参数说明

字段	必选/可选	类型	描述
deviceId	可选	String	设备唯一标识，1-64个字节
request	可选	UpdateDeviceInfoDTONA2Cloud	自定义结构体（见下表）

UpdateDeviceInfoDTONA2Cloud对象：

字段	必选/可选	类型	描述
name	可选	String	设备名称
endUser	可选	String	拥有者标识
mute	可选	Enum	表示设备是否处于冻结状态，即是否上报数据（处于冻结状态，则不上报数据），取值有”TRUE”,”FALSE”
manufacturerId	可选	String	厂商ID

字段	必选/可选	类型	描述
manufacturerName	可选	String	厂商名
location	可选	String	设备的位置
deviceType	可选	String	设备类型: 大驼峰命名方式 MultiSensor ContactSensor Camera Gateway
model	可选	String	型号 z-wave: ProductType + ProductId 16 进制： XXXX-XXXX 补0对齐 如：001A-0A12 其他协议再定
protocolType	可选	String	设备协议类型

返回结果

无，失败时抛出异常

调用示例

```
// device info
String deviceId = "7f7d****e67";
UpdateDeviceInfoDTONA2Cloud updateDeviceInfoDTONA2Cloud = new UpdateDeviceInfoDTONA2Cloud();
updateDeviceInfoDTONA2Cloud.setName("***");
updateDeviceInfoDTONA2Cloud.setEndUser("***");
// update device info
DeviceService.renameDevice(deviceId, updateDeviceInfoDTONA2Cloud);
```

10.1.3 数据采集

平台支持对网关事件的订阅，同时还支持查看传感器上报的历史数据，能按时、天、月等维度查看传感器上报的历史数据。数据采集实现了对网关数据收集。

SDK中封装类：com.huawei.iom.sdk.api.service.data.DataService

10.1.3.1 按条件批量查询设备信息列表

接口功能

按条件批量查询设备信息列表，如查询指定网关下的所有设备信息列表。

调用方法

```
public static QueryDevicesDTOCloud2NA queryDevicesInfo(String appId, String gatewayId, String
status, String nodeType,
            Integer pageNo, Integer pageSize, String startTime, String endTime,  String sort)
```

```
            throws IOException, HttpRequestException, URISyntaxException, ServiceException,
HttpResponseException
```

参数说明

字段	必选/可选	类型	描述
appId	可选	String	应用唯一标识
gatewayId	可选	String	设备唯一标识（含网关的设备唯一标识）。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。 gatewayId和pageNo不能同时为null，二者必选其一。
nodeType	可选	String	指定节点的类：ENDPOINT: 传感器. GATEWAY: 网关.
pageNo	可选	Integer	分页查询参数， pageNo=null时查询内容不分页；取值大于等于0的整数时分页查询,等于0时查询第一页。 gatewayId和pageNo不能同时为null，二者必选其一。
pageSize	可选	Integer	分页查询参数，取值大于等于1的整数，缺省：1
status	可选	String	设备状态,取值:ONLINE在线;OFFLINE 不在线;ABNORMAL异常状态
startTime	可选	String	查询注册设备信息时间在startTime之后的记录。时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z
endTime	可选	String	查询注册设备信息时间在endTime之前的记录。时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z
sort	可选	String	指定返回记录的排序，可取值ASC按注册设备的时间升序排列； DESC按注册设备的时间降序排列 缺省：DESC

注：gatewayId和pageNo不能同时为空。

结果QueryDevicesDTOCloud2NA对象说明

字段	必选/可选	类型	描述
totalCount	可选	Long	查询的记录数量
pageNo	必选	Long	返回的页码序号(大于等于0的整数,0代表第一页)
pageSize	必选	Long	页码大小,当不分页查询时,取值等于totalCount
devices	必选	List<QueryDeviceDTOCloud2NA>	自定义结构，见下表

QueryDeviceDTOCloud2NA结构

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
gatewayId	可选	String	设备唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
nodeType	必选	String	指定节点的类型 ENDPOINT: 传感器. GATEWAY: 网关.
creationTime	必选	String	设备注册时间；时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z
lastModifiedTime	必选	String	最新修改时间；时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z
deviceInfo	必选	DeviceInfo	DeviceInfo的定义参见3.1
services	可选	List<DeviceService>	DeviceService的定义参见3.2

调用示例

```
        // query parameter deviceId没给值时，返回对象的属性值都为null(deviceId = null;)
        String deviceId = "7f7****e67";//必选
        String gatewayId = deviceId;//必选
        String serviceId = null;
        int pageNo = 0;
        int pageSize = 1;
        String startTime = null;
        String endTime = null;
        // query
        QueryDeviceDataHistoryDTOCloud2NA queryDeviceDataHistoryDTOCloud2NA =
DataService.getDeviceHistory(deviceId, gatewayId, serviceId, pageNo, pageSize, startTime, endTime);
```

10.1.3.2 查询单个设备信息

接口功能

查询指定条件的设备信息。

调用方法

```
public static QueryDeviceDTOCloud2NA queryOneDeviceInfo(String deviceId, String appId)
            throws IOException, HttpRequestException, HttpResponseException, URISyntaxException,
ServiceException
```

参数说明

字段	必选/可选	类型	描述
appId	可选	String	应用唯一标识
deviceId	必选	String	设备唯一标识（含网关的设备唯一标识），1-64个字节

返回结果QueryDeviceDTOCloud2NA对象说明

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
gatewayId	可选	String	网关唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
nodeType	必选	String	指定节点的类型ENDPOINT: 传感器，GATEWAY: 网关。
creationTime	必选	String	设备注册时间 时间格式：yyyyMMdd’T’HHmmss’Z’ 如： 20151212T121212Z
lastModifiedTime	必选	String	最新修改时间 时间格式：yyyyMMdd’T’HHmmss’Z’ 如： 20151212T121212Z

字段	必选/可选	类型	描述
deviceInfo	必选	DeviceInfo	DeviceInfo的定义参见3.1章节**10.2.1 DeviceInfo结构体**
services	可选	List<DeviceService>	Service的定义参见3.2章节**10.2.2 DeviceService结构体**

调用示例

```
        // query parameter deviceId没给值时，返回对象的属性值都为null(deviceId = null;)
        String deviceId = "7f7d48ac-f7a1-4b36-b2df-a5d56633ee67";
        // query
        QueryDeviceDTOCloud2NA queryDeviceDTOCloud2NA = DataService.queryOneDeviceInfo(deviceId,
null);
```

10.1.3.3 查询设备历史数据

接口功能

Application查询设备历史数据。

调用方法

```
public static QueryDeviceDataHistoryDTOCloud2NA getDeviceHistory(String deviceId,String gatewayId,
String serviceId, Integer pageNo, Integer pageSize,
            String startTime, String endTime)
            throws IOException, HttpRequestException, HttpResponseException, URISyntaxException,
ServiceException
```

参数说明

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
gatewayId	必选	String	查询参数，网关的设备唯一标识。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
serviceId	可选	String	服务Id
pageNo	可选	Integer	分页查询参数， pageNo=null时查询内容不分页；取值大于等于0的整数时分页查询,等于0时查询第一页
pageSize	可选	Integer	分页查询参数，取值大于等于1的整数，缺省值为1 设备历史数据最多保存10万条，这个数值有可能被运营商修改

字段	必选/可选	类型	描述
startTime	可选	String	查询参数，根据时间段查询的起始时间; 时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z 设备历史数据最多保存90天，这个数值有可能被运营商修改
endTime	可选	String	查询参数，根据时间段查询的结束时间; 时间格式：yyyyMMdd’T’HHmmss’Z’ 如：20151212T121212Z

返回结果QueryDeviceDataHistoryDTOCloud2NA对象：

字段	必选/可选	类型	描述
totalCount	可选	Long	查询的记录数量
pageNo	必选	Long	返回的页码序号(大于等于0的整数,0代表第一页)
pageSize	必选	Long	页码大小,当不分页查询时,取值等于totalCount
list	必选	List<DeviceDataHistoryDTO>	历史记录列表

DeviceDataHistoryDTO对象：

字段	必选/可选	类型	描述
serviceId	必选	String	服务ID
deviceId	必选	String	设备唯一标识，1-64个字节
gatewayId	必选	String	网关的设备唯一标识，1-64个字节。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。
appId	必选	String	应用程序ID
data	必选	JsonObject	设备上报的数据，数据是键值对，取值请参见Profile文件中服务能力表里面的propertyties。

字段	必选/可选	类型	描述
timestamp	必选	String	上报数据的时间戳 时间格式： yyyyMMdd’T’HHmmss’Z’ 如： 20151212T121212Z

调用示例

```
// query parameter
        String deviceId = "7f7d48ac-f7a1-4b36-b2df-a5d56633ee67";//必选
        String gatewayId = deviceId;//必选
        String serviceId = null;
        int pageNo = 0;
        int pageSize = 1;
        String startTime = null;
        String endTime = null;
        // query
        QueryDeviceDataHistoryDTOCloud2NA queryDeviceDataHistoryDTOCloud2NA =
DataService.getDeviceHistory(deviceId, gatewayId, serviceId, pageNo, pageSize, startTime, endTime);
```

10.1.3.4 查询设备能力

接口功能

查询设备能力

调用方法

```
public static
QueryDeviceCapabilitiesDTOCloud2NA queryDeviceServiceCapabilities(String deviceId,
String gatewayId)
            throws IOException,
HttpRequestException, HttpResponseException, URISyntaxException, ServiceException
```

参数说明

字段	必选/可选	类型	描述
deviceId	条件可选	String	设备唯一标识，1-64个字节，deviceId与gatewayId必须有一个存在
gatewayId	条件可选	String	网关标识，1-64个字节，deviceId与gatewayId必须有一个存在。当设备是直连设备时，gatewayId为设备本身的deviceId。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。

返回QueryDeviceCapabilitiesDTOCloud2NA对象，结构如下：

字段	必选/可选	类型	描述
deviceCapabilitys	必选	List<DeviceCapabilityDTO>	自定义结构体，见下表

DeviceCapabilityDTO

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识，1-64个字节
serviceCapabilities	必选	List<DeviceServiceCapabilityDTO>	设备的服务能力列表

DeviceServiceCapabilityDTO

字段	必选/可选	类型	描述
serviceId	必选	String	服务标识
serviceType	必选	String	服务类型
description	可选	String	描述
commands	必选	List<ServiceCommand>	支持的命令名称列表,参见结构体3.4
properties	必选	List<ServiceProperty>	支持的属性名称列表，参见结构体3.5

调用示例

```
        // query parameter
        String deviceId = "7f7****e67";
        String gatewayId = null;
        // query
        QueryDeviceCapabilitiesDTOCloud2NA queryDeviceCapabilitiesDTOCloud2NA =
DataService.queryDeviceServiceCapabilities(deviceId, gatewayId);
```

10.1.4 设备服务调用

Application给设备发送命令消息，例如实现对传感器的实时控制。平台分别提供了下发至设备或者具体某传感器的控制命令接口，下发消息的具体格式需要Application与设备自定义，平台在接口中进行封装，透传。

具体调用与“2.2.2发现非直连设备”一致，请参考该章节。

10.1.5 规则

Application能够根据具体的业务需要，制定规则（条件）和动作，当网关的状态达到指定规则后，以邮件或者短信的方式对用户进行实时提醒。根据业务需要，编排规则，指定场景触发。

SDK中封装类：com.huawei.iom.sdk.api.service.rule.RuleService

10.1.5.1 创建规则

接口功能

创建规则。

调用方法

```
public static RuleCreationResultDTO createRule(RuleDTO ruleInfo)
            throws BaseException
```

参数RuleDTO对象结构说明

字段	必选/可选	类型	描述
ruleId	可选	String	规则实例的id，仅在规则更新时有效，此处不带。
appKey	可选	String	应用实例的key值(0-128)
name	必选	String	规则名（并且接受1~128个字符）
description	可选	String	规则描述(0-256)
author	必选	String	创建此规则的用户的id（0-128）
conditions	必选	List<ObjectNode>	条件列表，自定义结构体（见下表Condition-ObjectNode结构说明）
logic	可选	String	多条件之间的逻辑关系，支持and和or，默认为and
timeRange	可选	TimeRange	条件场景的时间段。TimeRange为自定义结构体（见下表）
actions	定时场景必选	List<ObjectNode >	规则的动作。Action为自定义结构体。（见下表Action-DeviceCMD结构说明）
matchNow	可选	String	表示是否立即触发，即是否理解进行条件判断，条件符合的话执行动作，取值有”yes”,”no”,默认为”no”.
status	可选	String	规则的状态，“active”代表激活状态，“inactive”代表未激活，默认为”active”状态

TimeRange结构体说明：

字段	必选/可选	类型	描述
startTime	必选	String	开始时间，格式为 ”HH:mm”。
endTime	必选	String	结束时间，格式为 ”HH:mm”，开始时间与结束时间一致，则认为是全天。
daysOfWeek	可选	String	星期列表，以逗号分隔。1代表周日，2代表周一，以此类推，默认每天。

Condition-ObjectNode结构说明：

字段	必选/可选	类型	描述
type	必选	String	条件触发condition类型，取值为"DEVICE_DATA"
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由IoT自动编号。
deviceInfo	必选	DeviceInfo	设备数据的定位信息，自定义结构，根据定位信息读取的数据作为比较表达式的左值
operator	必选	String	数据比较的操作符
value	必选	String	比较表达式的右值
transInfo	可选	Json	不需要平台理解的信息
duration	可选	Integer	规则执行延时时间，单位为分钟，默认为0不延时

DeviceInfo结构体说明：

字段	必选/可选	类型	描述
deviceId	必选	String	设备id
path	必选	String	以"/"分割的信息路径，按照当前的数据模型，路径为ServiceId/DataProperty，例如门磁状态为"DoorWindow/status"

Condition-DailyTimer结构说明：

字段	必选/可选	类型	描述
type	必选	String	定点触发condition类型，“DAILY_TIMER”
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
time	必选	String	指定执行时间点，格式为"HH:mm"
daysOfWeek	可选	String	星期列表，以逗号分隔，默认每天

Condition-CycleTimer结构说明：

字段	必选/可选	类型	描述
type	必选	String	周期性触发condition类型，取值为“CYCLE_TIMER”
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
timeRange	必选	timeRange	限定的时间范围
interval	必选	Integer	周期，单位为分钟

Action-DeviceCMD结构说明：

字段	必选/可选	类型	描述
type	必选	String	下发控制命令action类型，取值为“DEVICE_CMD”
id	可选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
appKey	可选	String	应用实例的key值
deviceId	必选	String	设备id
cmd	必选	CMD	命令内容
transInfo	可选	Json	不需要平台理解的信息

CMD结构说明：

字段	必选/可选	类型	描述
messageType	必选	String	命令名字
messageBody	必选	JSON	下发命令的消息内容
serviceId	必选	String	服务id

Action-SMS结构说明：

字段	必选/可选	类型	描述
type	必选	String	发送短信action类型，取值为”SMS”
id	可选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
msisdn	必选	String	短信通知地址
content	必选	String	通知内容
subject	必选	String	通知主题

Action-email结构说明：

字段	必选/可选	类型	描述
type	必选	String	发送邮件action类型，取值为“EMAIL”
id	可选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
email	必选	String	邮箱地址
content	必选	String	通知内容
subject	必选	String	通知主题

返回结果RuleCreationResultDTO对象

字段	必选/可选	类型	描述
ruleId	必选	String	规则实例id

调用示例

```
String deviceId = "7f7***e67";
// set create info
RuleDTO ruleInfo = new RuleDTO();
ruleInfo.setName("***");
ruleInfo.setAuthor("***");
List<ObjectNode> conditions = new ArrayList<ObjectNode>();
DeviceInfo deviceInfo = new DeviceInfo();
deviceInfo.setDeviceId(deviceId);
deviceInfo.setPath("DoorWindow/status");
ObjectNode condition1 = JsonUtil.createObjectNode();
condition1.put("type", "DEVICE_DATA");
condition1.put("deviceInfo",JsonUtil.jsonObj2Sting(deviceInfo));
condition1.put("operator", "=");
condition1.put("value", "OPEN");
conditions.add(condition1);
ruleInfo.setConditions(conditions);
List<ObjectNode> actions = new ArrayList<ObjectNode>();
ObjectNode action1 = JsonUtil.createObjectNode();
action1.put("type", "DEVICE_CMD");
action1.put("deviceId", deviceId);
CMD cmd1 = new CMD();
cmd1.setServiceId("RawData");
cmd1.setMessageType("rawData");
ObjectNode msgBody1 = JsonUtil.createObjectNode();
msgBody1.put("rawData", "MLAZ");
cmd1.setMessageBody(msgBody1);
action1.put("cmd", JsonUtil.jsonObj2Sting(cmd1));
actions.add(action1);
ruleInfo.setActions(actions);
// create
RuleCreationResultDTO response = RuleService.createRule(ruleInfo);
```

10.1.5.2 更新规则

接口功能

更新规则。实际过程是删除原有规则，创建新的规则，会产生新的规则ID。

调用方法

```
public static RuleCreationResultDTO updateRule(RuleDTO ruleInfo)
        throws BaseException
```

参数RuleDTO对象说明

字段	必选/可选	类型	描述
ruleId	必选	String	需要更新的规则实例的id（创建规则成功后平台返回的ruleId）
appKey	必选	String	应用实例的key值(0-128)
name	必选	String	规则名（并且接受1-128个字符）
description	可选	String	规则描述(0-256)
author	必选	String	创建此规则的用户的id（0-128）
conditions	必选	List<ObjectNode >	条件列表，自定义结构体（见下表）
logic	可选	String	多条件之间的逻辑关系，支持and和or，默认为and

字段	必选/可选	类型	描述
timeRange	可选	TimeRange	条件场景的时间段。TimeRange为自定义结构体（见下表）
actions	定时场景必选	List<ObjectNode >	规则的动作。Action为自定义结构体。（见下表）
matchNow	可选	String	表示是否立即触发，即是否理解进行条件判断，条件符合的话执行动作，取值有”yes”,”no”，默认为”no”
status	可选	String	规则的状态，“active”代表激活状态，“inactive”代表未激活，默认为”active”状态

TimeRange结构体说明：

字段	必选/可选	类型	描述
startTime	必选	String	开始时间，格式为 ”HH:mm”。
endTime	必选	String	结束时间，格式为 ”HH:mm”，开始时间与结束时间一致，则认为是全天。
daysOfWeek	可选	String	星期列表，以逗号分隔。1代表周日，2代表周一，以此类推，week周列表需与开始时间保持一致，默认每天。

Condition- ObjectNode结构说明：

字段	必选/可选	类型	描述
type	必选	String	条件触发condition类型，取值为"DEVICE_DATA"
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
deviceInfo	必选	DeviceInfo	设备数据的定位信息，自定义结构，根据定位信息读取的数据作为比较表达式的左值
operator	必选	String	数据比较的操作符
value	必选	String	比较表达式的右值

字段	必选/可选	类型	描述
transInfo	必选	Json	不需要平台理解的信息，规则触发之后通过规则事件返回给NA
duration	可选	Integer	规则执行延时时间，单位为分钟，默认为0不延时

DeviceInfo结构体说明：

字段	必选/可选	类型	描述
deviceId	必选	String	设备id
path	必选	String	以"/"分割的信息路径，按照当前的数据模型，路径为ServiceId/DataProperty，例如门磁状态为"DoorWindow/status"

Condition-DailyTimer结构说明：

字段	必选/可选	类型	描述
type	必选	String	定点触发condition类型，“DAILY_TIMER”
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
time	必选	String	指定执行时间点，格式为"HH:MM"
daysOfWeek	必选	String	星期列表，以逗号分隔， 默认每天

Condition-CycleTimer结构说明：

字段	必选/可选	类型	描述
type	必选	String	周期性触发condition类型，取值为“CYCLE_TIMER”
id	可选	String	对condition的编号，在一条规则范围内唯一，不指定时由系统自动编号
timeRange	必选	timeRange	限定的时间范围

字段	必选/可选	类型	描述
interval	必选	Integer	周期，单位为分钟

Action-DeviceCMD结构说明：

字段	必选/可选	类型	描述
type	必选	String	下发控制命令action类型，取值为“DEVICE_CMD”
id	可选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
appKey	必选	String	应用实例的key值
deviceId	必选	String	设备id
cmd	必选	CMD	命令内容
transInfo	必选	Json	不需要平台理解的信息，规则触发之后通过规则事件返回给NA

CMD结构说明：

字段	必选/可选	类型	描述
messageType	必选	String	命令名字
messageBody	必选	Map<String, String>	命令参数列表
serviceId	必选	String	服务id

Action-SMS结构说明：

字段	必选/可选	类型	描述
type	必选	String	发送短信action类型，取值为”SMS”
id	可选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
msisdn	必选	String	短信通知地址
content	必选	String	通知内容

字段	必选/可选	类型	描述
subject	必选	String	通知主题

Action-email结构说明：

字段	必选/可选	类型	描述
type	必选	String	发送邮件action类型，取值为“EMAIL”
id	可选	String	对action的编号，在一条规则范围内唯一，不指定时由系统自动编号
email	必选	String	邮箱地址
content	必选	String	通知内容
subject	必选	String	通知主题

返回结果RuleCreationResultDTO对象：

字段	必选/可选	类型	描述
ruleId	必选	String	规则实例id

消息示例

```
Method:
PUT
request:
https://server:port/iocm/app/rule/v1.2.0/rules
Content-Type:application/json
Header:
"app_key: aOpVJb06fLzRShwOMhOywFvGsIga "
"Authorization:Bearer f9a2cbbdabefae2b91b21c4172c0d534"
Body:
{
    "ruleId": "*****",
    "appKey": "aOpVJb06fLzRShwOMhOywFvGsIga",
    "name": "scene condx",
    "description": "scene condx",
    "author": "23558",
    "conditions": [
        {
            "duration": 0,
            "transInfo": {
                "rightValue": "OPEN"
            },
            "id": "0",
            "type": "DEVICE_DATA",
            "value": "OPEN",
            "deviceInfo": {
                "path": "DoorWindow/status",
```

```
                "deviceId": "2770f874-dec4-468e-bcb1-66425aa9d546"
            },
            "operator": "="
        }
    ],
    "logic": "and",
    "timeRange": null,
    "actions": [
        {
            "transInfo": {},
            "appKey": "aOpVJb06fLzRShwOMh0ywFvGsIga",
            "cmd": {
                "messageBody": {
                    "mode": "ON",
                    "format": "AVI",
                    "time": 60
                },
                "messageType": "VIDEO_RECORD",
                "serviceId": "Camera"
            },
            "id": "8",
            "type": "DEVICE_CMD",
            "deviceId": "17c0d895-370f-4a97-8989-1ff07af9ec15"
        }
    ],
    "matchNow": "no",
    "status": "active"
}
response:
Status Code: 200 OK
Content-Type:application/json
Body:
{
  "ruleId":"576e51aa45706d1584f7c06f"
}
```

10.1.5.3 修改单个规则状态

接口功能

修改单个规则状态。

调用方法

```
public static int changeStatus(String ruleId, String status)
            throws BaseException
```

参数说明

字段	必选/可选	类型	描述
ruleId	必选	String	需要修改的规则的id
status	必选	String	规则状态，active代表激活，inactive代表未激活

返回结果

返回执行状态，成功时返回200

调用示例

```
        // init
```

```
        String ruleId = "583****50c";
        String status = "active"; //inactive active
        // change status
        int responseStatus = RuleService.changeStatus(ruleId, status);
```

10.1.5.4 删除规则

接口功能

删除规则。

调用方法

```
public static int deleteRule(String ruleId)
            throws BaseException, URISyntaxException
```

参数说明

字段	必选/可选	类型	描述
ruleId	必选	String	需要删除的规则的id

返回结果

返回执行状态，成功时返回204

调用示例

```
        // init
        String ruleId = "583****531";
        // delete
        int responseStatus = RuleService.deleteRule(ruleId);
```

10.1.5.5 查找规则

接口功能

查找规则。

接口路径

```
public static List<RuleDTO> findRuleByNameAndAuthor(String ruleName, String author)
            throws BaseException, URISyntaxException
```

参数说明

字段	必选/可选	类型	描述
author	必选	String	用户id
name	可选	String	规则名称

返回结果

Rule列表，同创建/更新规则结构体

消息示例

```
// init
String ruleName = null;//
String author = "***";
// query
List<RuleDTO> ruleDTOList = RuleService.findRuleByNameAndAuthor(ruleName, author);
```

10.1.6 订阅平台数据

北向应用向平台订阅设备变更信息，一个APP一个事件对应一个订阅回调地址。

SDK中封装类：com.huawei.iom.sdk.api.service.subscribe.SubscribeService

备注说明：调用后返回的数据格式参见“华为IoT_Platform_API参考(北向)”文档。

10.1.6.1 北向应用订阅平台数据

接口功能

北向应用订阅设备变更通知，当设备发生变更时平台会推送给北向应用。

调用方法

```
public static int subscribeDeviceEvent(SubscribeDTONA2Cloud request)
            throws BaseException
```

参数说明

字段	必选/可选	类型	描述
notifyType	必选	String	通知类型，如： 1、bindDevice（设备激活通知） 2、deviceAdded（添加新设备） 3、deviceInfoChanged（设备信息变化） 4、deviceDataChanged（设备数据变化） 5、deviceDeleted（删除设备） 6、deviceEvent（设备事件） 7、messageConfirm（消息确认） 8、commandRsp（响应命令） 9、serviceInfoChanged（服务信息） 10、ruleEvent（规则事件）
callbackurl	必选	String	回调的url地址。回调地址中必须指定回调地址的端口。

返回结果

返回执行状态，成功时返回201

调用示例

```
// set parameter
SubscribeDTONA2Cloud request = new SubscribeDTONA2Cloud();
request.setNotifyType("bindDevice");
request.setCallbackurl("http://185.12.26.50:9999");
// subscribe event
int responseStatus = SubscribeService.subscribeDeviceEvent(request);
```

10.2 常用数据结构体定义

10.2.1 DeviceInfo 结构体

字段	必选/可选	类型	描述
nodeId	必选	String	设备唯一标识
name	可选	String	设备名称
description	可选	String	设备描述
manufactureId	必选	String	供应商标识
manufacturerName	可选	String	供应商名字
mac	可选	String	物理地址
location	可选	String	设备位置
deviceType	必选	String	设备种类： MultiSensor ContactSensor Camera Gateway
model	必选	String	设备型号. 如果是Z-Wave协议，型号就是ProductType + ProductId（用零填充，如果需要） 格式的十六进制数,例如 001A-0A12. 如果是其他协议，型号需要另一种格式表示
swVersion	可选	String	使用Z-Wave协议时的软件版本号。 其格式是 major version number.minor version number, 例如, 1.1.
fwVersion	可选	String	固件版本号
hwVersion	可选	String	硬件版本号
protocolType	必选	String	协议类型 (Z-Wave).
signalStrength	可选	String	信号强度

字段	必选/可选	类型	描述
bridgeId	可选	String	网桥标识
supportedSecurity	可选	String	是否支持安全模式，其取值为TRUE或FALSE
isSecurity	可选	String	设备是否安全，在安全模式为TRUE的情况下有效，其取值为TRUE或FALSE
sigVersion	可选	String	信令版本
runningStatus	可选	String	设备运行状态： NORMAL：正常 ABNORMAL：异常 FAULT：故障
status	可选	String	设备状态. ONLINE: 在线 OFFLINE: 下线 INBOX: 停用.
statusDetail	可选	String	设备状态的详细情况
mute	可选	String	设备是否进行屏蔽 TRUE: 被屏蔽. FALSE: 没被屏蔽.

10.2.2 DeviceService 结构体

字段	必选/可选	类型	描述
serviceType	必选	String	服务类型
serviceId	必选	String	服务ID标识
data	必选	JSON String	指定一个属性值对(AVP).
eventTime	必选	DateTime	时间格式 yyyymmddThhmissZ , 例如, 20151212T121212Z.
serviceInfo	可选	ServiceInfo	业务信息，此字段用于内部控制功能，不用关心此字段

10.2.3 QueryDeviceDTOCloud2NA 结构体

字段	必选/可选	类型	描述
deviceId	必选	String	设备唯一标识
gatewayId	必选	String	网关唯一标识。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。当设备是直连设备时，gatewayId为设备本身的deviceId。
nodeType	必选	String	指定节点的类型 ENDPOINT: 传感器. GATEWAY: 网关.
createTime	必选	String	设备注册时间
lastModifiedTime	必选	String	设备信息最后修改时间
deviceInfo	必选	DeviceInfo	详见"**10.2.1 DeviceInfo结构体**"
services	可选	List<DeviceService>	详见 "**10.2.2 DeviceService结构体**"

10.2.4 ServiceCommand 结构体

字段	必选/可选	类型	描述
commandName	可选	String	命令
paras	可选	List<ServiceCommandPara>	属性列表，参考下表

ServiceCommandPara结构体

字段	必选/可选	类型	描述
paraName	可选	String	参数名
dataType	可选	String	数据类型
required	可选	boolean	是否必选
min	可选	int	最小
max	可选	int	最大

字段	必选/可选	类型	描述
step	可选	Double	步长
maxLength	可选	int	最大长度
unit	可选	String	单位（符号）
enumList	可选	List<String>	枚举类型列表

10.2.5 ServiceProperty 结构体

字段	必选/可选	类型	描述
propertyName	可选	String	属性名
dataType	可选	String	数据类型
required	可选	boolean	是否必选
min	可选	int	最小
max	可选	int	最大
step	可选	double	步长
maxLength	可选	int	最大长度
method	可选	String	访问方法：RW 可读R，可写W
unit	可选	String	单位（符号）
enumList	可选	List<String>	枚举类型列表