**集中控制设备通信API调用说明**

**V1.0**

# 变更记录

2015.3.3：V1.0

新建文档

2015.3.16：V1.1

增加API接口

# 初始化

用于通信模块的运行环境初始化，在使用该模块之前，必须进行下文中的初始操作。

## 配置资源路径

用于配置加密、鉴权的资源文件路径。

代码示例：

StringBuilder confPath = **new** StringBuilder(); confPath.append(Class.**class**.getClass().getResource("/").getPath()); confPath.append("repository");

GlobalConfig.*setRepositoryPath*(confPath.toString());

## 配置网卡

在多网卡的环境下，用于配置设备发现时使用的网卡，传入参数为网卡的IP地址。

代码示例:

System.*setProperty*("NetworkInterface", "192.168.17.122");

# 全局参数设置

## 是否打印设备发现信息

开启后，每次进行设备发现时，将在控制台打印设备发现的结果。默认情况下为关闭。

代码示例：

GlobalConfig.*setIsPrintDiscoverResults*(**true**);

## 设置设备发现频率

用于设置，每次设备发现的时间间隔，单位毫秒。默认情况下为10秒。

代码示例：

GlobalConfig.*setDiscoveryIntervalMllisecond*(1000\*30);

## 设置设备发现超时时间

用于设置，每次发出设备发现数据包后，模块等待设备回发数据包的超时时间，单位毫秒。默认情况下为1秒。

代码示例：

GlobalConfig.*setDiscoveryTimeOutMllisecond*(1000\*3);

## 设置控制命令请求超时时间

用于设置，向设备发出控制请求后，模块等待设备反馈的超时时间，单位毫秒。默认情况下为30秒。

代码示例：

GlobalConfig.*setRequestTimeOutMllisecond*(1000\*10);

## 全局参数定义表

该定义表为GlobalDefinitions类中的字段，用于定义各类状态及消息的字符格式。

设备类型：

**public** **static** **final** String ***DeviceType\_IPC*** = "IPC"; 设备类型，IPC

**public** **static** **final** String ***DeviceType\_HREC***="HREC"; 设备类型，录播设备

**public** **static** **final** String ***DeviceType\_HHTC***="HHTC"; 设备类型，集控设备

设备状态：

**public** **static** **final** String ***OnlineStatus\_Offline***="Offline"; 设备状态，离线

**public** **static** **final** String ***OnlineStatus\_Online***="Online"; 设备状态，在线

**public** **static** **final** String ***OnlineStatus\_WrongAuth***="WrongAuth"; 设备状态，权限错误

消息类型：用于模块对外事件通知的回调

**public** **static** **final** String ***Event\_DeviceOnline***="101"; 消息类型，设备上线

**public** **static** **final** String ***Event\_DeviceOffline***="102"; 消息类型，设备离线

**public** **static** **final** String ***Event\_DeviceWrongAuth***="103"; 消息类型，设备权限发生错误

**public** **static** **final** String ***Event\_DeviceMessage***="104"; 消息类型，设备消息通知

设备协议类型：

**public** **static** **final** String ***ProtocolType\_Onvif***="ONVIF"; 协议类型，ONVIF

**public** **static** **final** String ***ProtocolType\_HHTC***="HHTC"; 协议类型，集控

通用分隔符：

**public** **static** **final** String ***Other\_Separator***=":::::::";

# 设备

## 设备接口

类名：Devcie

接口方法：

1. 获取IP地址：

**public** String getIP()

1. 获取协议类型：

**public** String getProtocolType();

1. 获取设备类型：

**public** String getDeviceType();

1. 获取设备状态：

**public** String getDeviceStatus();

1. 设置设备自定义信息：

**public** **void** SetDeviceExtensionInfo(Hashtable<String, Object> info);

1. 获取设备自定义信息：

**public** Hashtable<String, Object> GetDeviceExtensionInfo();

1. 权限测试：

**public** **boolean** AuthTest();

1. 设备同步请求:

**public** Object Invoke(OperationRequest request); 具体返回参数参见下文命令请求部分。

1. 设备异步请求：

**public** **void** StartInvoke(String UUID,OperationRequest request,DeviceCallBackHandler callBackHandler);

UUID：异步请求的流水号，请求完毕后在回调中会传回该流水号，用于异步命令的身份识别。

request：命令请求。

DeviceCallBackHandler：异步回调句柄，具体调用参数参见下文回调句柄部分。

## 集控设备

类名HHTCDevice，实现上述Device接口。

调用方法：

1. 构造函数： **public** HHTCDevice(String deviceIP, String userID, String password,String deviceType) deviceIP：设备IP地址，必填项，不能为NULL。

userID：用户名，如果没有则填NULL。

password：密码，如果没有则填NULL。

deviceType：设备类型，选填。

1. 设置用户名：

**public** **void** setUserID(String userID)

1. 获取用户名：

**public** String getUserID()

1. 设置密码：

**public** **void** setPassword(String password)

1. 获取密码：

**public** String getPassword()

1. 获取设备型号：

**public** String getLocalModle() {

1. 获取设备班级名称：

**public** String getHostName() {

# 回调函数

## 异步通信

类名：DeviceCallBackHandler，抽象类，使用时需要自己实现抽象方法，用于回调信息的处理。

调用方法：

1. 命令执行完毕： **public** **abstract** **void** OnInvokeSuccess(String strUUID,String strDeviceToken,String strRequestClassName,OperationRequest Request,Object objResult);

strUUID：异步通信执行时传入的命令流水号。

strDeviceToken：执行命令的设备令牌号。

strRequestClassName：暂时无用。

Request：暂时无用。

objResult：命令请求的返回值，具体返回参数参见下文命令请求部分。

1. 命令执行错误： **public** **abstract** **void** OnInvokeError(String strUUID,String strDeviceToken,String strRequestClassName,OperationRequest Request,String strErrorContent);

strUUID：异步通信执行时传入的命令流水号。

strDeviceToken：执行命令的设备令牌号。

strRequestClassName：暂时无用。

Request：暂时无用。

strErrorContent：错误信息。

## 消息通知

类名：DeviceEventHandler，抽象类，使用时需要自己实现抽象方法，用于通知信息的处理。

调用方法：

1. 消息通知： **public** **abstract** **void** OnEvent(String strDeviceToken,String strEventType,String strEventContext);

strDeviceToken：执行命令的设备令牌号。

strEveentType:消息类型。

strEventContext：消息内容。

# 设备管理

类名：DeviceManager，静态类，所有方法为静态方法。

调用方法：

1. 设备同步请求： **public** **static** Object Invoke(String strToken,OperationRequest request)

strToken：需要执行的设备令牌号，令牌号格式为：IP+“-”+协议类型。

request：命令请求。

1. 设备异步请求： **public** **static** **void** StartInvoke(String strToken,String UUID,OperationRequest request,DeviceCallBackHandler callBackHandler)

strToken：需要执行的设备令牌号，令牌号格式为：IP+“-”+协议类型。

UUID：异步请求的流水号，请求完毕后在回调中会传回该流水号，用于异步命令的身份识别。

request：命令请求。

DeviceCallBackHandler：异步回调句柄。

1. 启动设备发现服务： **public** **static** **void** Run()

用于启动设备发现服务，服务启动后，会定时进行设备发现，并实时维护设备状态和设备信息。

1. 关闭设备发现服务： **public** **static** **void** Stop(){
2. 设置消息通知句柄： **public** **static** **void** SetEventHandler(DeviceEventHandler handler)

设置消息通知句柄后，设备管理模块产生的消息会实时回调到该句柄的接口函数上。

1. 获取设备： **public** **static** Device GetDevice(String strDeviceToken) {

根据设备令牌号获取设备实体。获取范围为所有已发现的设备和已注册的设备。

1. 获取已发现设备： **public** **static** Set<String> GetDiscoveredDevicesList()

设备发现服务启动后，调用该方法会返回所有已发现的设备列表，已注册的设备不会在该列表中出现。列表中 为设备的令牌号。

1. 获取已注册设备： **public** **static** Set<String> GetRegistedDevicesList()

获取所有已注册的设备列表，列表中为设备的令牌号。

1. 注册设备： **public** **static** **void** RegisteDevice(Device device)

向设备管理模块注册设备，注册后模块会自动维护设备信息和设备状态，设备产生消息后会实时回调消息通知 接口。

1. 注销设备： **public** **static** **void** RemoveDevice(String strDeviceToken)

向设备管理模块注销设备。

1. 设备开机： **public** **static** **void** WakeUpDevice(String strMAC)

通过网络唤醒的形式，对设备开机

# 命令请求

命令请求的实体，实例化该类并作为Invoke的参数传入，命令请求时需要的参数通过构造函数传参，返回的参数在同步请求的模式下直接通过方法调用的返回值返回，在异步请求的模式下在异步通信的回调函数中返回。请求参数和返回参数的文字内容和格式参见《集控平台和受控端协议定义》

1. 查询音频模式： HHTCOperationRequestGetAudioMode

请求参数：无

返回参数：String AudioMode

1. 设置音频模式： HHTCOperationRequestSetAudioMode

请求参数：String AudioMode

返回参数：无

1. 查询信号源： HHTCOperationRequestGetChannel

请求参数：无

返回参数：String ChannelName

1. 设置信号源： HHTCOperationRequestSetChannel

请求参数：String ChannelName

返回参数：无

1. 查询节能模式： HHTCOperationRequestGetEnergyMode

请求参数：无

返回参数：String EnergyMode

1. 设置节能模式： HHTCOperationRequestSetEnergyMode

请求参数：String EnergyMode

返回参数：无

1. 查询触控模式： HHTCOperationRequestGetTouchMode

请求参数：无

返回参数：String TouchMode

1. 设置触控模式： HHTCOperationRequestSetTouchMode

请求参数：String TouchMode

返回参数：无

1. 设置音量：HHTCOperationRequestSetVolume

请求参数：int volume

返回参数：无

1. 设置关机：HHTCOperationRequestSetPower

请求参数：String power

返回参数：无

1. 设置系统同步时间：HHTCOperationRequestSetSysTime

请求参数：int year,int month,int day,int hour,int minute,int second

返回参数：无

1. 查询所有状态信息：HHTCOperationRequestGetAllState

请求参数：无

返回参数：String channelName，String audioName，String energyName，String volumeNum，String touchName

1. 获取OPS截屏数据：HHTCOperationRequestScreenShot

请求参数：无

返回参数：String base64Str

1. 信息发布消息：HHTCOperationRequestNewsManager

请求参数：Map<String,String> newsMap

返回参数：无

1. 更新数据库：HHTCOperationRequestUpdateDB

请求参数：String SQL

返回参数：String updateFlag

1. 事件订阅：HHTCOperationRequestEventSubscription

请求参数：String strAddress

返回参数：无

# Demo

//设置资源路径

StringBuilder confPath = **new** StringBuilder();

confPath.append(Class.**class**.getClass().getResource("/").getPath());

confPath.append("repository");

GlobalConfig.*setRepositoryPath*(confPath.toString());

//设置网卡

System.*setProperty*("NetworkInterface", "192.168.17.135");

//设置设备发现间隔时间

GlobalConfig.*setDiscoveryIntervalMllisecond*(1000\*30);

//打印设备发现信息

GlobalConfig.*setIsPrintDiscoverResults*(**true**);

//设置消息通知回调

DeviceEventHandler eventHandler=**new** DeviceEventHandler() {

@Override

**public** **void** OnEvent(String strDeviceToken, String strEventType,

String strEventContext) {

// **TODO** Auto-generated method stub

System.***out***.println("消息通知：");

System.***out***.println(" 设备令牌号："+strDeviceToken);

System.***out***.println(" 消息类型"+strEventType);

System.***out***.println(" 消息内容"+strEventContext);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

};

//设置异步请求回调

DeviceCallBackHandler callBackHandler=**new** DeviceCallBackHandler() {

@Override

**public** **void** OnInvokeSuccess(String strUUID, String strDeviceToken,

String strRequestClassName, OperationRequest Request,

Object objResult) {

System.***out***.println("异步请求成功:");

System.***out***.println(" 命令流水号："+strUUID);

System.***out***.println(" 设备令牌号："+strDeviceToken);

**if**(objResult!=**null**)

System.***out***.println(" 返回结果："+objResult);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

@Override

**public** **void** OnInvokeError(String strUUID, String strDeviceToken,

String strRequestClassName, OperationRequest Request,

String strErrorContent) {

// **TODO** Auto-generated method stub

System.***out***.println("异步请求失败：");

System.***out***.println(" 命令流水号："+strUUID);

System.***out***.println(" 设备令牌号："+strDeviceToken);

System.***out***.println(" 异常信息"+ strErrorContent);

}

};

//启动设备发现服务

DeviceManager.*Run*();

//延迟5秒，打印已发现的设备

Thread.*sleep*(1000\*10);

System.***out***.println("已发现设备：");

System.***out***.println(DeviceManager.*GetDiscoveredDevicesList*());

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//向设备管理模块获取设备，读取设备信息

Device device=DeviceManager.*GetDevice*("192.168.17.201-ONVIF");

**if** (device!=**null**) {

System.***out***.println("读取设备信息");

System.***out***.println(" IP:"+device.getIP());

System.***out***.println(" 协议类型："+device.getProtocolType());

System.***out***.println(" 设备类型："+device.getDeviceType());

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

//遍历已发现的集控设备，并注册设备

**for** (String token : DeviceManager.*GetDiscoveredDevicesList*()) {

**if** (token.split("-")[1].equals(GlobalDefinitions.***ProtocolType\_HHTC***)) {

HHTCDevice hhtcDevice=(HHTCDevice)DeviceManager.*GetDevice*(token);

DeviceManager.*RegisteDevice*(hhtcDevice);

}

}

System.***out***.println("读取已注册设备：");

System.***out***.println(DeviceManager.*GetRegistedDevicesList*());

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//遍历已注册的集控设备，并同步发送控制命令，查询音频模式

System.***out***.println("同步遍历音频模式：");

**for** (String token : DeviceManager.*GetRegistedDevicesList*()) {

System.***out***.println(" Token:"+token);

String audiomode=(String)DeviceManager.*Invoke*(token, **new** HHTCOperationRequestGetAudioMode());

System.***out***.println(" AudioMode:"+audiomode);

}

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//便利以注册的集控蛇设备，并异步发送控制命令，查询音频模式

System.***out***.println("异步遍历音频模式：");

**for** (String token : DeviceManager.*GetRegistedDevicesList*()) {

String strUUID=UUID.*randomUUID*().toString();

System.***out***.println(" 发出异步命令，Token:"+token+" UUID:"+strUUID);

DeviceManager.*StartInvoke*(token, strUUID, **new** HHTCOperationRequestGetAudioMode(), callBackHandler);

}

System.***out***.println(" 异步命令发送完毕。");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");