Android NetSDK Demo优化

概要设计说明书

|  |  |
| --- | --- |
| 编制 | 杨铮 |
| 审核 | 冯晓晨 |
| 批准 |  |

**目录**

[1. 简介 3](#_Toc12720220)

[1.1 编写目的 3](#_Toc12720221)

[1.2背景 3](#_Toc12720222)

[1.3术语和缩写 3](#_Toc12720223)

[1.4参考资料 3](#_Toc12720224)

[2. 总体架构 3](#_Toc12720225)

[2.1 系统说明 3](#_Toc12720226)

[2.2 运行环境 3](#_Toc12720227)

[2.3 总体结构及模块划分 4](#_Toc12720228)

[2.4 模块划分列表 4](#_Toc12720229)

[2.5 可测试性、维护/升级设计 4](#_Toc12720230)

[2.6 可配置设计 5](#_Toc12720231)

[2.7 尚未解决的问题 5](#_Toc12720232)

[3. 模块说明 5](#_Toc12720233)

[3.1 PD-001 视图模块 5](#_Toc12720234)

[3.1.1 模块功能描述 5](#_Toc12720235)

[3.1.2 子模块划分列表 5](#_Toc12720236)

[3.1.3 子模块设计说明 5](#_Toc12720237)

[3.2 PD-002控制模块 12](#_Toc12720238)

[3.2.1 模块功能描述 12](#_Toc12720239)

[3.2.2 子模块划分列表 12](#_Toc12720240)

[3.2.3 子模块设计说明 12](#_Toc12720241)

[3.3 PD-003数据模块 14](#_Toc12720242)

[3.3.1 模块功能描述 14](#_Toc12720243)

[3.3.2 模块设计说明 14](#_Toc12720244)

[4. 接口说明 15](#_Toc12720245)

[4.1 SDKGuider公共接口 15](#_Toc12720246)

[4.2 DevManageGuider设备管理接口 16](#_Toc12720247)

[4.3 DevConfigGuider远程配置类接口 19](#_Toc12720248)

[5. 修订记录 24](#_Toc12720249)

# 简介

## 1.1 编写目的

描述Android NetSDK demo开发框架及相关接口和对象，为Demo后来维护的人员提供文档参考。

## 1.2背景

随着客户集成Andorid HCNetSDK的需求的不断增长，老版本的Demo由于其简陋的界面和复杂的代码逻辑，不再适用于当下客户的集成需求。为了给客户提供更加清晰的指导代码和直观界面，我们在对老demo功能进行梳理，按照功能划分模块，重构Andorid Demo框架。

## 1.3术语和缩写

| 术语/缩写 | 含 义 |
| --- | --- |
| SDK | Software Development Kit，软件开发套件，本文中“SDK”，“网络SDK”都是指海康威视设备网络SDK |
| SadpSDK | 设备发现SDK |
| Android studio(AS) | Android Studio 是开发 Android 应用程序的官方 IDE，基于 Intellij IDEA框架开发。缩写为AS。 |

## 1.4参考资料

无

# 总体架构

## 2.1 系统说明

新版demo将老版demo进行拆分，按照功能划分为设备管理、预览、回放、远程配置、报警等模块，为客户提供直观且专注的界面。实现控制、界面、数据的完全分离，将SDK接口调用的业务逻辑从界面代码中剥离，为客户提供完整无干扰的SDK示例代码。

## 2.2 运行环境

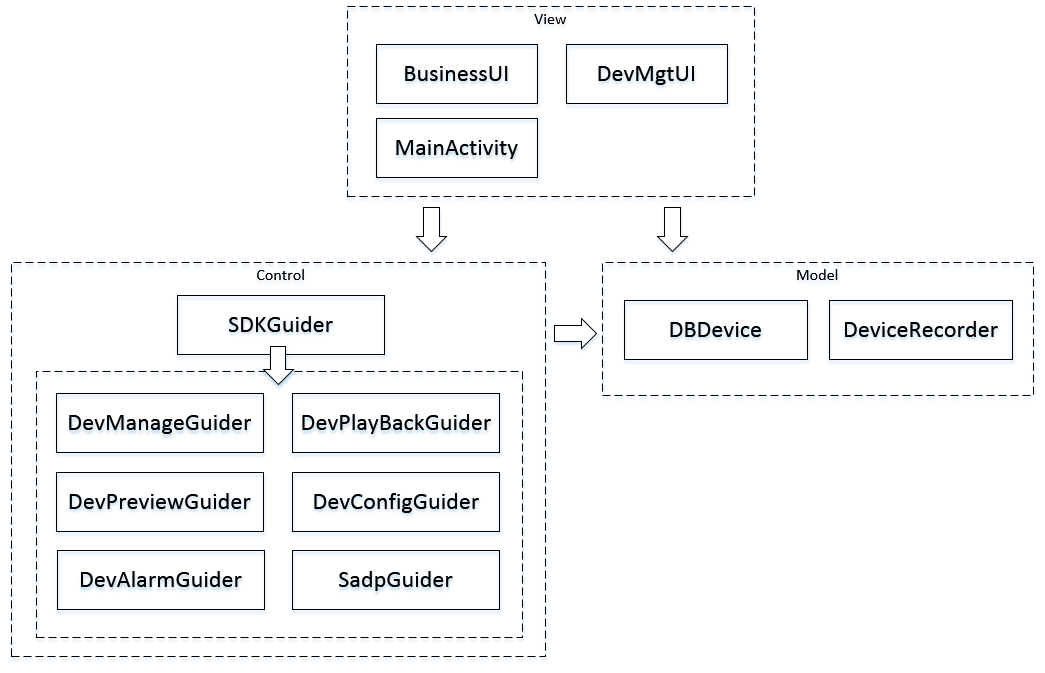
Demo运行于Android系统中，根据HCNetSDK提供的arm64-v8a、armeabi-v7a、armeabi版本，需保证Demo能够在以上CPU架构的平台上正常运行。

App基于Android Studio 3.1.3环境开发，Gradle版本4.4，AndroidSDK版本28，可支持的最低SDK版本为15。其他依赖库，如下：

|  |
| --- |
| com.android.support:appcompat-v7:28.0.0 |
| com.android.support:design:28.0.0 |
| com.android.support.constraint:constraint-layout:1.1.3 |
| com.android.support:support-v4:28.0.0 |

## 2.3 总体结构及模块划分

Demo各模块的依赖关系如下：



其中，设备管理、预览、回放、远程配置、报警、Sadp作为控制模块的子模块，实现对HCNetSDK及SadpSDK接口的调用，并对外提供封装的接口。

## 2.4 模块划分列表

|  |  |  |
| --- | --- | --- |
| 模块编号 | 模块名称 | 模块描述 |
| PD-001 | 视图模块 | demo界面相关逻辑 |
| PD-002 | 控制模块 | SDK接口封装 |
| PD-003 | 数据模块 | 登录设备持久化 |

## 2.5 可测试性、维护/升级设计

无

## 2.6 可配置设计

无。

## 2.7 尚未解决的问题

无。

# 模块说明

Demo使用MVC模式设计，实现控制、视图、数据模块分离，以期望获得更好的可扩展性和代码的可读性。



软件总流程图

## 3.1 PD-001 视图模块

### 3.1.1 模块功能描述

视图模块定义并实现demo的界面相关的功能，为用户操作设备提供直观的交互体验。界面按照功能划分为：主界面、预览界面、回放界面、远程配置界面、报警界面、设备发现界面。

### 3.1.2 子模块划分列表

|  |  |  |
| --- | --- | --- |
| 模块编号 | 模块名称 | 模块来源 |
| PD-001-001 | 主界面模块 | 新建 |
| PD-001-002 | 预览界面模块 | 新建 |
| PD-001-003 | 回放界面模块 | 新建 |
| PD-001-004 | 远程配置界面模块 | 新建 |
| PD-001-005 | 报警界面模块 | 新建 |
| PD-001-006 | 设备发现界面模块 | 新建 |

### 3.1.3 子模块设计说明

3.1.3.1 PD-001-001 主界面模块

功能描述：

主界面模块负责全局设备管理，同时维护各功能界面的入口。

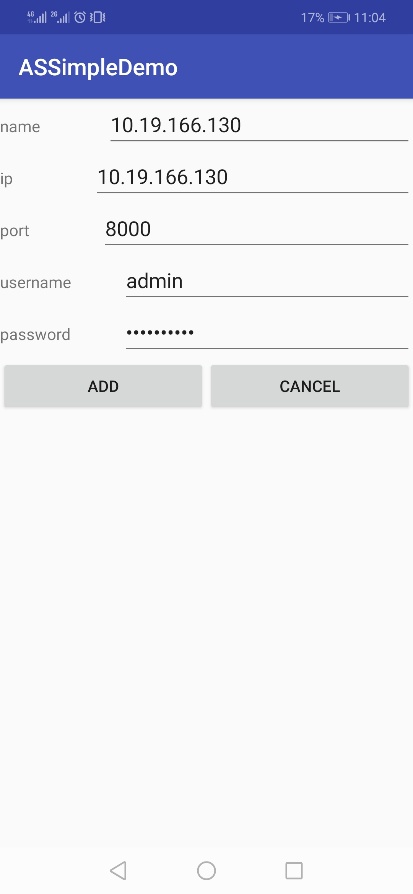
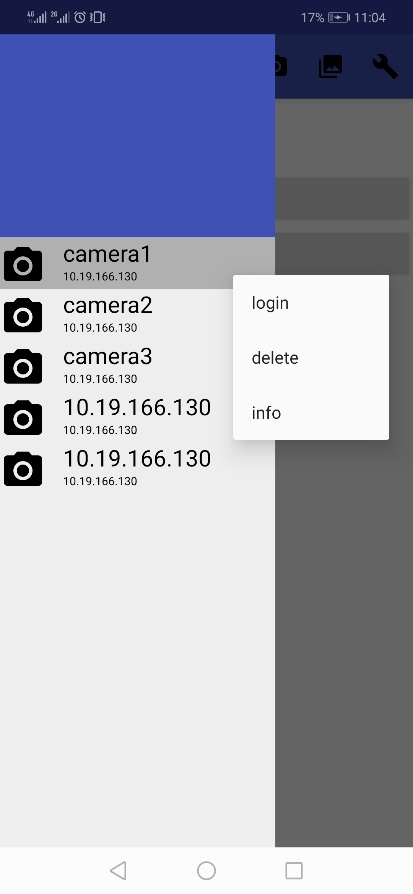


图 3-1 （左）功能Table页界面，（中）设备管理界面，（右）设备添加界面

其中功能Table页界面，将各功能界面如预览，回放界面等以Table页的形式放在主界面上，方便用户切换；设备管理界面以侧滑窗口的形式呈现，方便用户在任何时候呼出管理设备，设备添加界面通过点击主界面工具栏按钮呼出，实现新设备添加。

模块类图：

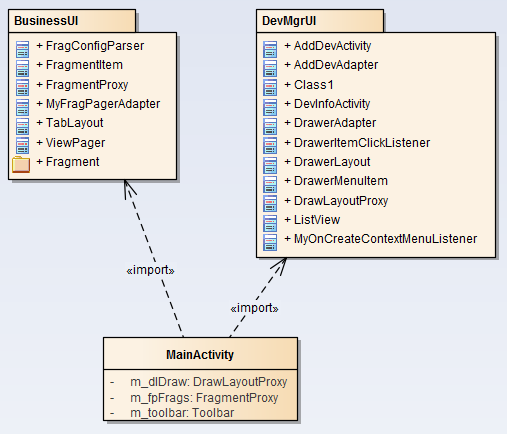


图 3-2 主界面包依赖

其中，BusinessUI包定义并实现了功能业务界面，DevMgrUI包则定义实现设备管理相关界面。

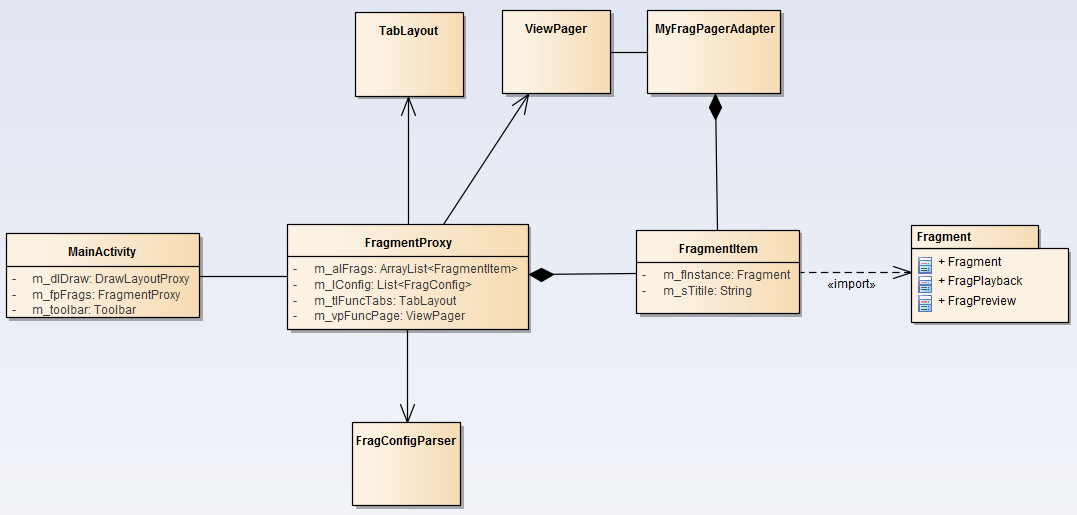


图 3-3 BusinessUI包类图

BusinessUI包定义并实现了功能业务界面相关的类，其中FragmentProxy对象负责根据配置文件创建功能业务界面Fragment窗口。

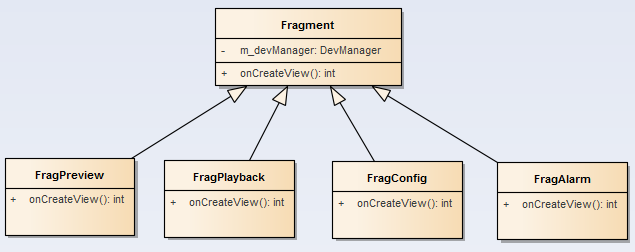


图3-4 Fragment类图

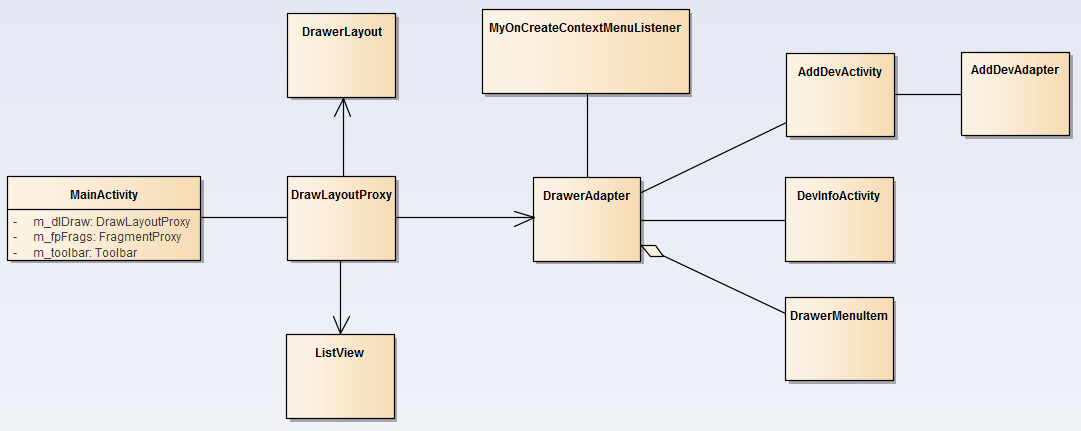
为了统一Fragment页面构造方式，减少界面间耦合度，采用配置文件+反射的方式构造功能业务Fragment页面，用户完成功能界面开发后，只需要在配置文件中注册Fragment入口类路径，demo就能够自动加载窗口。配置文件的解析由FragConfigParser对象完成。配置文件如下：

|  |
| --- |
| <fragconfig>  <fragitem className=".View.BusinessUI.Fragment.PlayBack.FragPlayBack"  titile="PlayBack"  args="@null"/>  <fragitem  className=".View.BusinessUI.Fragment.Alarm.FragAlarm"  titile="Alarm"  args="@null"/>  <fragitem  className=".View.BusinessUI.Fragment.Configure.FragConfig"  titile="Config"  args="@null"/></fragconfig> |

Fragment加载流程



设备管理界面类图：



设备管理界面以列表视图的形式管理设备，界面初始化时会读取用户设备配置文件和SQLite设备数据库中的设备项，并加入到设备列表中。流程如下：

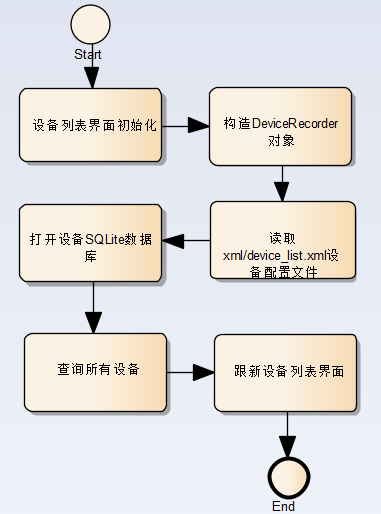


图3-5设备配置加载流程

3.1.3.2 PD-001-002 预览界面模块

由陈佳工完善。

3.1.3.3 PD-001-003 回放界面模块

由胡炳伦工完善。

3.1.3.4 PD-001-004 远程配置界面模块

界面设计如下：



网络参数、报警输入配置参数、通道配置等为各个功能模块的配置。每个功能的配置由若干个子功能的测试函数组成。点击大模块按钮，在界面下方的listView呈现该大功能模块的若干个子模块的测试函数；点击各个测试函数，会在界面以toast方式呈现该测试函数调用接口的状态，成功或者失败。若成功，显示接口调用成功，若失败，呈现接口调用失败的错误码。

3.1.3.5 PD-001-005 报警界面模块



本报警模块界面主要是布防方式的报警。点击布防按钮，若设备上传报警信息，在界面上方的listView中显示。

3.1.3.6 PD-001-006 设备发现界面模块

由杨晓峰工完善。

## 3.2 PD-002控制模块

### 3.2.1 模块功能描述

控制模块主要负责调用SDK接口，实现控制控制。控制模块对外提供SDK接口封装，实现SDK业务逻辑与界面逻辑的分离，降低模块间的耦合度。同时，控制模块向SDK的集成开发人员提供无干扰示例代码，便于开发人员理解SDK集成方式。

### 3.2.2 子模块划分列表

|  |  |  |
| --- | --- | --- |
| 模块编号 | 模块名称 | 模块来源 |
| PD-002-001 | 设备管理模块 | 新建 |
| PD-002-002 | 预览模块 | 新建 |
| PD-002-003 | 回放模块 | 新建 |
| PD-002-004 | 远程配置模块 | 新建 |
| PD-002-005 | 报警模块 | 新建 |
| PD-002-006 | 设备发现模块 | 新建 |

### 3.2.3 子模块设计说明

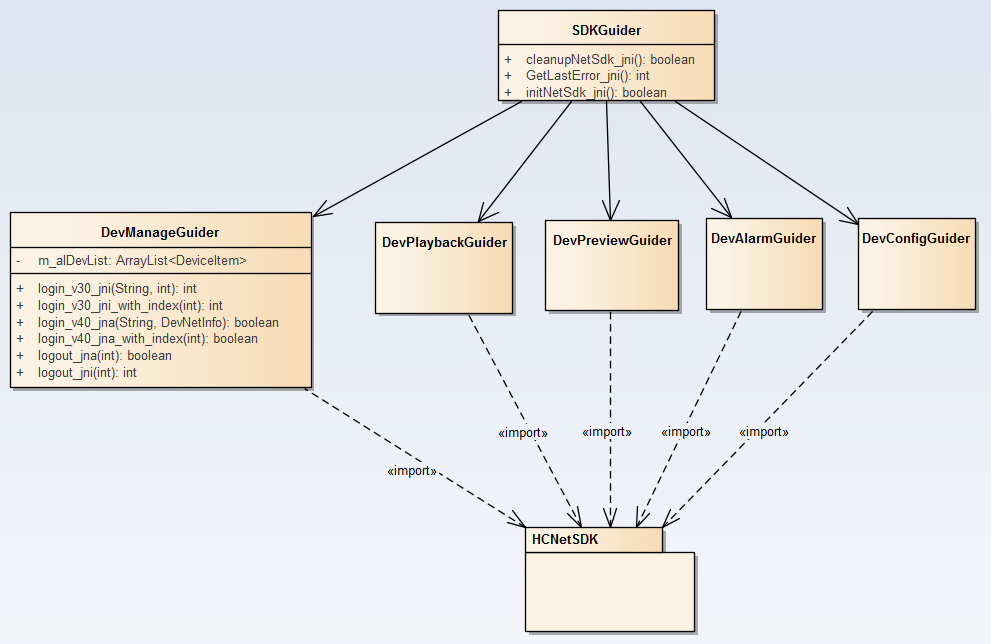


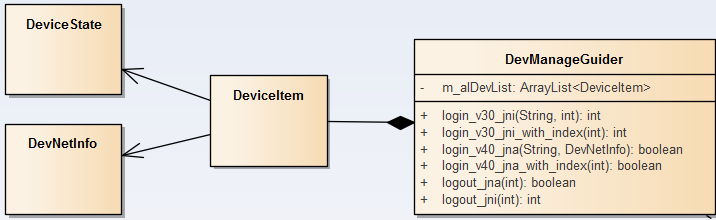
图3-6 控制模块类图

3.2.3.1 PD-002-001 设备管理模块

功能描述：

设备管理子模块负责全局设备管理，维护设备的登录信息，状态等。并对外提供设备登陆、注销、获取设备列表、查找设备等接口

类图：



其中DeviceItem是设备列表中的设备项，包含一个设备的所有信息。

3.2.3.2 PD-002-002 预览模块

由陈佳工完善。

3.2.3.3 PD-002-003 回放模块

由胡炳伦工完善。

3.2.3.4 PD-002-004 远程配置模块

配置类功能比较多，需要分门别类管理。首先将接口分为几个大类，然后在大类中分为若干个配置子类。几个大类如下图所示：

目前ASSimpleDemo暂时列出这几个类，后续可以根据客户需求自行增加。

每个大类由若干个testFunction组成，每个testFunction调用各个配置接口。

3.2.3.5 PD-002-005 报警模块

由焦媛媛工完善。

3.2.3.6 PD-002-00 设备发现模块

由杨晓峰工完善。

## 3.3 PD-003数据模块

### 3.3.1 模块功能描述

数据模块主要负责数据的持久化操作，从用户的配置文件中读取设备配置，或将成功登陆的设备保存到SQLite数据库。

### 3.3.2 模块设计说明

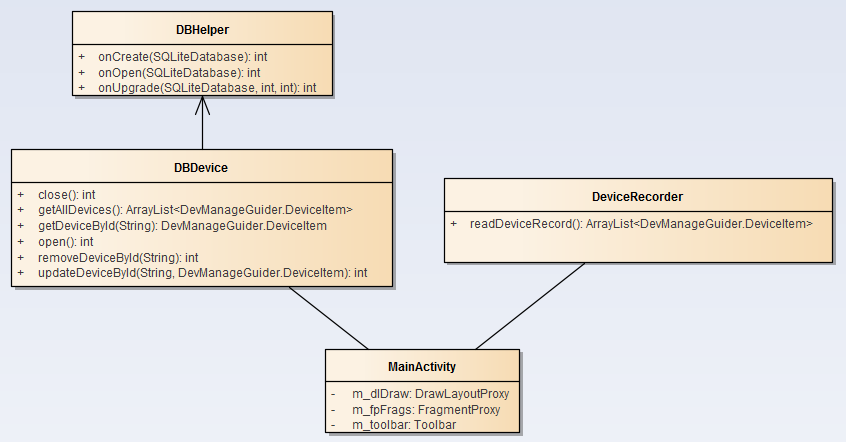


图3-7 模块类图

其中，DBHelper派生于SQLiteOpenHelper，DBDevice对象负责操作DB，实现设备项的增删改查。DeviceRecorder对象负责从用户的配置文件中读取设备项并添加到设备列表界面上，方便用户批量添加设备。

# 接口说明

## 4.1 SDKGuider公共接口

/\*\*

\* @fn GetLastError\_jni

\* @return 返回错误码

\* @brief 获取SDK错误码

\*/

int GetLastError\_jni()

/\*\*

\* @fn initNetSdk\_jni

\* @return 成功初始化NetSDK,返回True,否则False

\* @brief NetSDK初始化.

\*/

boolean initNetSdk\_jni()

/\*\*

\* @fn cleanupNetSdk\_jni

\* @return 成功返回True,否则False

\* @brief NetSDK反初始化.

\*/

boolean cleanupNetSdk\_jni()

## 4.2 DevManageGuider设备管理接口

/\*\*

\* @fn setCurrSelectDevIndex

\* @param [int] iCurrSelectDevIndex

\* @brief 设置当前选中设备的列表序号，当设备列表界面中选中设备时调用。

\*/

void setCurrSelectDevIndex(int iCurrSelectDevIndex)

/\*\*

\* @fn getCurrSelectDevIndex

\* @return [int] 设备索引号

\* @brief 获取当前选中的设备在设备列表中的索引号.

\*/

int getCurrSelectDevIndex()

/\*\*

\* @brief 设备网络参数对象

\*/

public class DevNetInfo implements Serializable {

public String m\_szIp; //ip地址

public String m\_szPort; //端口

public String m\_szUserName; //登陆用户名

public String m\_szPassword; //登陆密码

/\*\*

\* 判断IP格式和范围

\*/

public boolean checkIp(){

if(m\_szIp.length() < 7 || m\_szIp.length() > 15 || "".equals(m\_szIp))

{

return false;

}

if (m\_szIp != null && !m\_szIp.isEmpty()) {

// 定义正则表达式，ip

String regex = "^(1\\d{2}|2[0-4]\\d|25[0-5]|[1-9]\\d|[1-9])\\.(1\\d{2}|2[0-4]\\d|25[0-5]|[1-9]\\d|\\d)\\.(1\\d{2}|2[0-4]\\d|25[0-5]|[1-9]\\d|\\d)\\.(1\\d{2}|2[0-4]\\d|25[0-5]|[1-9]\\d|\\d)$";

// 判断ip地址是否与正则表达式匹配

return m\_szIp.matches(regex);

}

return false;

}

public boolean checkPort() {

Pattern pattern = Pattern.compile("[1-9][0-9]\*");

return pattern.matcher(m\_szPort).matches();

}

public boolean checkNetInfo(){

return checkIp()&&checkPort()

&&!m\_szUserName.isEmpty()&&!m\_szPassword.isEmpty();

}

}

/\*\*

\* @brief 设备状态对象

\*/

public class DeviceState implements Serializable {

public int m\_iLogState = 0; // 0-offline, 1-online, 2-dropoff

public int m\_iAlarmState = 0; // 0-alarmclosed, 1-alarmopen

public void reset(){

m\_iLogState = 0;

m\_iAlarmState = 0;

}

}

/\*\*

\* @brief 设备信息对象

\*/

public class DeviceItem implements Serializable {

public String m\_szUUID; //登录设备的全局唯一ID，设备数据库表主键

public String m\_szDevName; //设备名

public int m\_lUserID = -1; //设备登陆ID

public byte m\_byLoginFlag = -1; // 设备登录方式，0-jni, 1-jna

public DeviceState m\_struDevState = new DeviceState(); //设备状态

public DevNetInfo m\_struNetInfo; // 设备网络参数

public HCNetSDKByJNA.NET\_DVR\_DEVICEINFO\_V40 m\_struDeviceInfoV40\_jna; //V40登陆返回信息结构体

public NET\_DVR\_DEVICEINFO\_V30 m\_struDeviceInfoV30\_jni; //V30登陆返回信息结构体

public DeviceItem() {

m\_szUUID = UUID.randomUUID().toString();

}

public DeviceItem(String szUUID) {

m\_szUUID = szUUID;

}

}

/\*\*

\* @fn getCurrSelectDev

\* @return [DeviceItem] 设备信息对象

\* @brief 获取当前选中设备的信息对象

\*/

DeviceItem getCurrSelectDev()

/\*\*

\* @fn getDevByUserID

\* @param [in] lUserID 设备登陆ID

\* @return设备信息对象

\* @brief 使用登陆id查找设备信息对象

\*/

DeviceItem getDevByUserID(int lUserID)

/\*\*

\* @fn getDevList

\* @return设备列表

\* @brief 获取设备列表

\*/

ArrayList<DeviceItem> getDevList()

/\*\*

\* @fn login\_v40\_jna

\* @param [in] szDevName 设备名称

\* @param [in] struDevNetInfo 设备登陆的网络参数

\* @return 登陆成功返回true,否则false

\* @brief登陆设备

\*/

boolean login\_v40\_jni(String szDevName, DevNetInfo struDevNetInfo)

/\*\*

\* @fn login\_v40\_jna\_with\_index

\* @param [in] iDevIndex 设备列表中的索引号

\* @return 登陆成功返回true,否则false

\* @brief 当设备已经添加到设备列表之后，可以通过在列表中的索引号进行登陆

\*/

boolean login\_v40\_jni\_with\_index(int iDevIndex)

/\*\*

\* @fn logout\_jna

\* @param [in] iDevIndex 设备索引号

\* @return 注销成功true,否则false

\* @brief 注销设备

\*/

boolean logout\_jni(int iDevIndex)

## 4.3 DevConfigGuider远程配置类接口

由于远程配置类接口比较多，每个功能模块只列出一个接口进行说明。

**ScreenTest：**

**public** CGReturn test\_ControlScreen\_jni(**int** iUserID)  
{  
 CGReturn ret=**new** CGReturn();  
 NET\_DVR\_SCREEN\_CONTROL\_V41 struCtrlControl\_V41 = **new** NET\_DVR\_SCREEN\_CONTROL\_V41();  
 struCtrlControl\_V41.**byProtocol** = 1;  
 struCtrlControl\_V41.**bySerialNo** = 1;  
 struCtrlControl\_V41.**byWallNo** = 1;  
 struCtrlControl\_V41.**dwCommand** = 1;  
 struCtrlControl\_V41.**struControlParam**.**byInputSourceType** = 0;  
 struCtrlControl\_V41.**struControlParam**.**byColorType** = 1;  
 struCtrlControl\_V41.**struControlParam**.**byColorScale** = 1;  
 struCtrlControl\_V41.**struControlParam**.**byPosition** = 1;  
 struCtrlControl\_V41.**struControlParam**.**byPositionScale** = 1;  
 struCtrlControl\_V41.**struRect**.**dwHeight** = 1920;  
 struCtrlControl\_V41.**struRect**.**dwWidth** = 100;  
 struCtrlControl\_V41.**struRect**.**dwXCoordinate** =0;  
 struCtrlControl\_V41.**struRect**.**dwYCoordinate** = 0;  
 **if**(HCNetSDK.*getInstance*().NET\_DVR\_RemoteControl(iUserID,HCNetSDK.*getInstance*().***NET\_DVR\_CONTROL\_SCREEN***,struCtrlControl\_V41))  
 {  
 ret.**status\_1**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 Log.*i*(**"[NetSDKSimpleDemo]"**,**"NET\_DVR\_CONTROL\_SCREEN success!"**);  
 }  
 **else** {  
 ret.**status\_1**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 Log.*e*(**"[NetSDKSimpleDemo]"**,**"NET\_DVR\_CONTROL\_SCREEN fail,error code = "**+HCNetSDK.*getInstance*().NET\_DVR\_GetLastError());  
 }  
 **return** ret;  
}

**ConfigTest：**

**public static void** Test\_Time(**int** iUserID)  
{  
 NET\_DVR\_TIME struTimeCfg = **new** NET\_DVR\_TIME();  
 **if**(!HCNetSDK.*getInstance*().NET\_DVR\_GetDVRConfig(iUserID, HCNetSDK.*getInstance*().***NET\_DVR\_GET\_TIMECFG***, 0, struTimeCfg))  
 {  
 System.***out***.println(**"NET\_DVR\_GET\_TIMECFG faild!"** + **" err: "** + HCNetSDK.*getInstance*().NET\_DVR\_GetErrorMsg(***ptrINT***));  
 }  
 **else** {  
 System.***out***.println(**"NET\_DVR\_GET\_TIMECFG succ:"** + struTimeCfg.ToString());  
 }  
 **if**(!HCNetSDK.*getInstance*().NET\_DVR\_SetDVRConfig(iUserID, HCNetSDK.*getInstance*().***NET\_DVR\_SET\_TIMECFG***, 0, struTimeCfg))  
 {  
 System.***out***.println(**"NET\_DVR\_SET\_TIMECFG faild!"** + **" err: "** + HCNetSDK.*getInstance*().NET\_DVR\_GetErrorMsg(***ptrINT***));  
 }  
 **else** {  
 System.***out***.println(**"NET\_DVR\_SET\_TIMECFG succ:"** + struTimeCfg.ToString());  
 }  
}

**ManageTest：**

**public** CGReturn Test\_SearchLog\_jni(**int** iUserID)  
 {  
 CGReturn ret=**new** CGReturn();  
 NET\_DVR\_TIME struStartTime = **new** NET\_DVR\_TIME();  
 NET\_DVR\_TIME struEndTime = **new** NET\_DVR\_TIME();  
 struStartTime.**dwYear** = 2014;  
 struStartTime.**dwMonth** = 11;  
 struStartTime.**dwDay** = 26;  
 struEndTime.**dwYear** = 2014;  
 struEndTime.**dwMonth** = 11;  
 struEndTime.**dwDay** = 27;  
 **int** nSearchHandle = (**int**)HCNetSDK.*getInstance*().NET\_DVR\_FindDVRLog\_V30(iUserID, 0, 0, 0, struStartTime, struEndTime, **false**);  
 **if**(nSearchHandle >= 0)  
 {  
 NET\_DVR\_LOG\_V30 struLog = **new** NET\_DVR\_LOG\_V30();  
 **int** nState = -1;  
 **int** nCount = 0;  
 **while**(**true**)  
 {  
 nState = (**int**)HCNetSDK.*getInstance*().NET\_DVR\_FindNextLog\_V30((**long**)nSearchHandle, struLog);  
 **if**(nState != HCNetSDK.***NET\_DVR\_FILE\_SUCCESS*** && nState != HCNetSDK.***NET\_DVR\_ISFINDING***)  
 {  
 ret.**status\_1**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 **break**;  
 }  
 **else if**(nState == HCNetSDK.***NET\_DVR\_FILE\_SUCCESS***)  
 {  
 ret.**status\_1**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 nCount++;  
 System.***out***.println(**"find log time:"** + struLog.**strLogTime**.**dwHour** + **"-"** + struLog.**strLogTime**.**dwMinute** + **"-"** + struLog.**strLogTime**.**dwSecond**);  
 }  
 **try** {  
 Thread.*sleep*(100);  
 } **catch** (InterruptedException e)  
 {  
 e.printStackTrace();  
 }  
 }  
 HCNetSDK.*getInstance*().NET\_DVR\_FindLogClose\_V30(nSearchHandle);  
 }  
 **return** ret;  
 }

**PictureTest:**

**public** CGReturn PicUpload(**int** iUserID)  
{  
 CGReturn ret=**new** CGReturn();  
 NET\_DVR\_PICTURECFG struPic= **new** NET\_DVR\_PICTURECFG();  
 struPic.**byUseType** = 1;  
 struPic.**bySequence** = 1;  
 struPic.**sPicName**[0]=**'a'**;  
 struPic.**sPicName**[1]=**'b'**;  
 struPic.**dwVideoWallNo** = 0x0100000;  
 struPic.**struBasemapCfg**.**byScreenIndex** = 1;  
 struPic.**struBasemapCfg**.**byMapNum** = 1;  
 struPic.**struBasemapCfg**.**wSourHeight** = 1;  
 struPic.**struBasemapCfg**.**wSourWidth** = 1;  
 String sFileName = **"/mnt/sdcard/Pictures/Screenshots/Screenshot\_2015-10-12-21-27-05.png"**;  
  
 File f = **new** File(sFileName);  
 **if** (f.exists()) {  
 Log.*i*(**"[NetSDKSimpleDemo]"**, **"exist "**+sFileName);  
  
 } **else** {  
 Log.*i*(**"[NetSDKSimpleDemo]"**, **"not exist "**+sFileName);  
 }  
  
 *m\_iHandle* = HCNetSDK.*getInstance*().NET\_DVR\_PicUpload(iUserID, sFileName, struPic);  
 **if**(*m\_iHandle* == -1)  
 {  
 ret.**status\_1**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 Log.*e*(**"[NetSDKSimpleDemo]"**,**"NET\_DVR\_PicUpload fail,error coed = "**+HCNetSDK.*getInstance*().NET\_DVR\_GetLastError());  
 **return** ret;  
 }  
 **else** {  
 ret.**status\_1**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 Log.*i*(**"[NetSDKSimpleDemo]"**,**"NET\_DVR\_PicUpload success"**);  
 }  
  
 **for**(**int** i=0;i<100;i++)  
 {  
 **int** iprogress = HCNetSDK.*getInstance*().NET\_DVR\_GetPicUploadProgress(*m\_iHandle*);  
 Log.*i*(**"[NetSDKSimpleDemo]"**,**"i="**+i+**",---iprogress="**+iprogress);  
 *//Thread.sleep(10);* }  
  
 **if**(HCNetSDK.*getInstance*().NET\_DVR\_CloseUploadHandle(*m\_iHandle*))  
 {  
 Log.*i*(**"[NetSDKSimpleDemo]"**,**"NET\_DVR\_CloseUploadHandle success"**);  
 }  
 **else** {  
 Log.*e*(**"[NetSDKSimpleDemo]"**,**"NET\_DVR\_CloseUploadHandle fail,error coed = "**+HCNetSDK.*getInstance*().NET\_DVR\_GetLastError());  
 }  
 **return** ret;  
}

**PTZTest:**

**public** CGReturn Test\_PTZControl(**int** iPreviewID)  
{  
 CGReturn ret=**new** CGReturn();  
 **if**(!HCNetSDK.*getInstance*().NET\_DVR\_PTZControl(iPreviewID, PTZCommand.***PAN\_LEFT***, 0))  
 {  
 ret.**status\_1**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 System.***out***.println(**"PTZControl PAN\_LEFT 0 faild!"** + **" err: "** + HCNetSDK.*getInstance*().NET\_DVR\_GetLastError());  
 }  
 **else** {  
 ret.**status\_1**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 System.***out***.println(**"PTZControl PAN\_LEFT 0 succ"**);  
 }  
 **try** {  
 Thread.*sleep*(2000);  
 } **catch** (InterruptedException e) {  
 *//* ***TODO Auto-generated catch block*** e.printStackTrace();  
 }  
 **if**(!HCNetSDK.*getInstance*().NET\_DVR\_PTZControl(iPreviewID, PTZCommand.***PAN\_LEFT***, 1))  
 {  
 ret.**status\_2**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 System.***out***.println(**"PTZControl PAN\_LEFT 1 faild!"** + **" err: "** + HCNetSDK.*getInstance*().NET\_DVR\_GetLastError());  
 }  
 **else** {  
 ret.**status\_2**=HCNetSDK.*getInstance*().NET\_DVR\_GetLastError();  
 System.***out***.println(**"PTZControl PAN\_LEFT 1 succ"**);  
 }  
 **return** ret;  
}

# 错误处理说明

视图模块的个关键代码处，如调用业务逻辑接口后，都会以日志的形式打印返回结果，通过日志可以排查程序执行正确性。

业务模块，在调用SDK接口后，判断接口调用返回，并在错误时获取错误码，并以日志的方式打印错误，便于问题排查。

数据模块需要操作文件及数据库，在文件操作代码用try{}catch{}圈住，以及时捕获处理异常，避免程序崩溃。

# 修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **变更时间** | **版本** | **变更人** | **变更说明** |
| 1 | 2019.06.13 | 1.0 | 杨铮 | 新建及评审版本的编写 |
| 2 | 2019.06.26 | 1.1 | 焦媛媛 | 增加远程配置 |