Ford Motor (China) Company

HMI SDK 接口说明书

Version <1.0.0>

免责申明

本文档中的内容仅供参考。福特汽车中国对本服务内容的错误或遗漏概不负责。在任何情况下，福特汽车中国均不对因使用本文档而产生或与之相关的任何特殊，直接，间接，间接或偶然的损害赔偿或任何损害负责，无论是在合同，疏忽，其他侵权行为中服务或服务的内容。福特汽车中国保留随时对本文档内容进行补充，删除或修改的权利，恕不另行通知。

Disclaimer

The contents contained in this document are for general information purposes only. Ford Motor China assumes no responsibility for errors or omissions in the contents on the Service.

In no event shall Ford Motor China be liable for any special, direct, indirect, consequential, or incidental damages or any damages whatsoever, whether in an action of contract, negligence or other tort, arising out of or in connection with the use of the Document or the contents of the Document.

Ford Motor China reserves the right to make additions, deletions, or modification to the contents on the Service at any time without prior notice.

编制	Beyondsoft	日期	2017-11-24	版权	署名-相同方式共享 4.0 国际
审核	Ford	日期	2017-11-24	管理	Ford

修改历史

版本	日期	说明
1.0.0	2017-11-24	初版作成

目录

1 引言

1.1 背景

考虑到 HMI 通信层、Json 解析层以及 App 管理部分变化不大，客户需要定制和修改的主要是 UI 显示部分，每次开发新的 HMI 需要重复拷贝除 UI 部分的代码，容易产生错误并且需要了解该部分涉及到的相关知识和调用流程，为了减少重复部分的开发工作，特提出将前面提及到不变的部分封装为库的形式，并提供接口及接口说明文档供 UI 开发人员使用和参考。

1.2 内容

本文档描述的主要内容包括，HMISDK 主要的接口关系图、调用流程（时序图）、各类接口的说明以及接口方法的详细描述，包括参数、返回值、注意事项等。

1.3 适用范围

本文档主要面向开发人员，指导 UI 界面开发人员如何去调用 HMISDK，协助开发人员更好的使用 HMISDK，解说每个接口函数使用的方法及含义，提高开发效率，减少重复工作。

1.4 术语

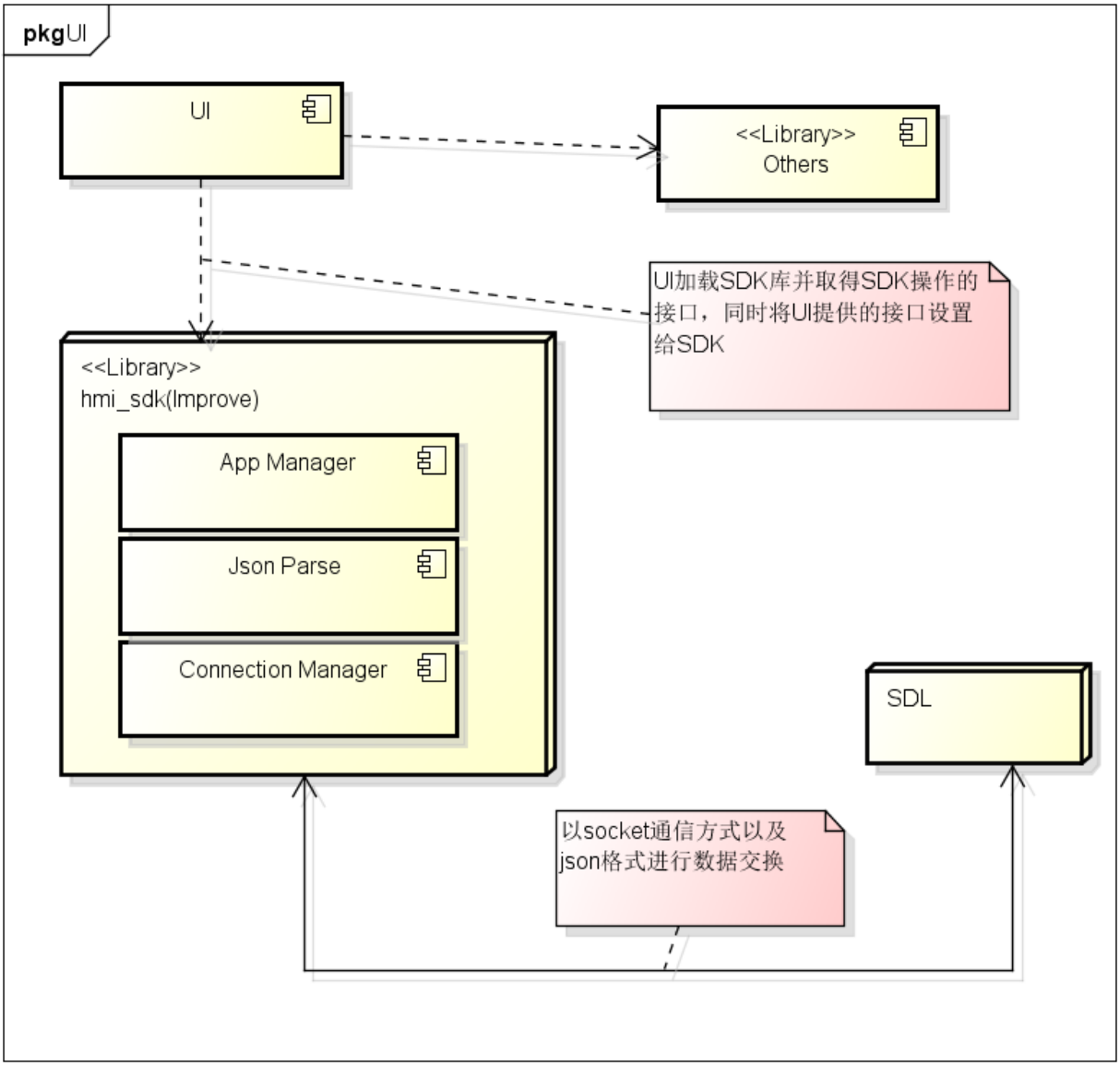
缩写	含义
HMI	人机交互界面
RPC	远程过程调用
SDK	HMISDK 部分，也是此次重构的重点
UI	画面显示部分
SDL	SmartDeviceLink
VR	语音识别
TTS	文本转换语音

1.5 参考资料

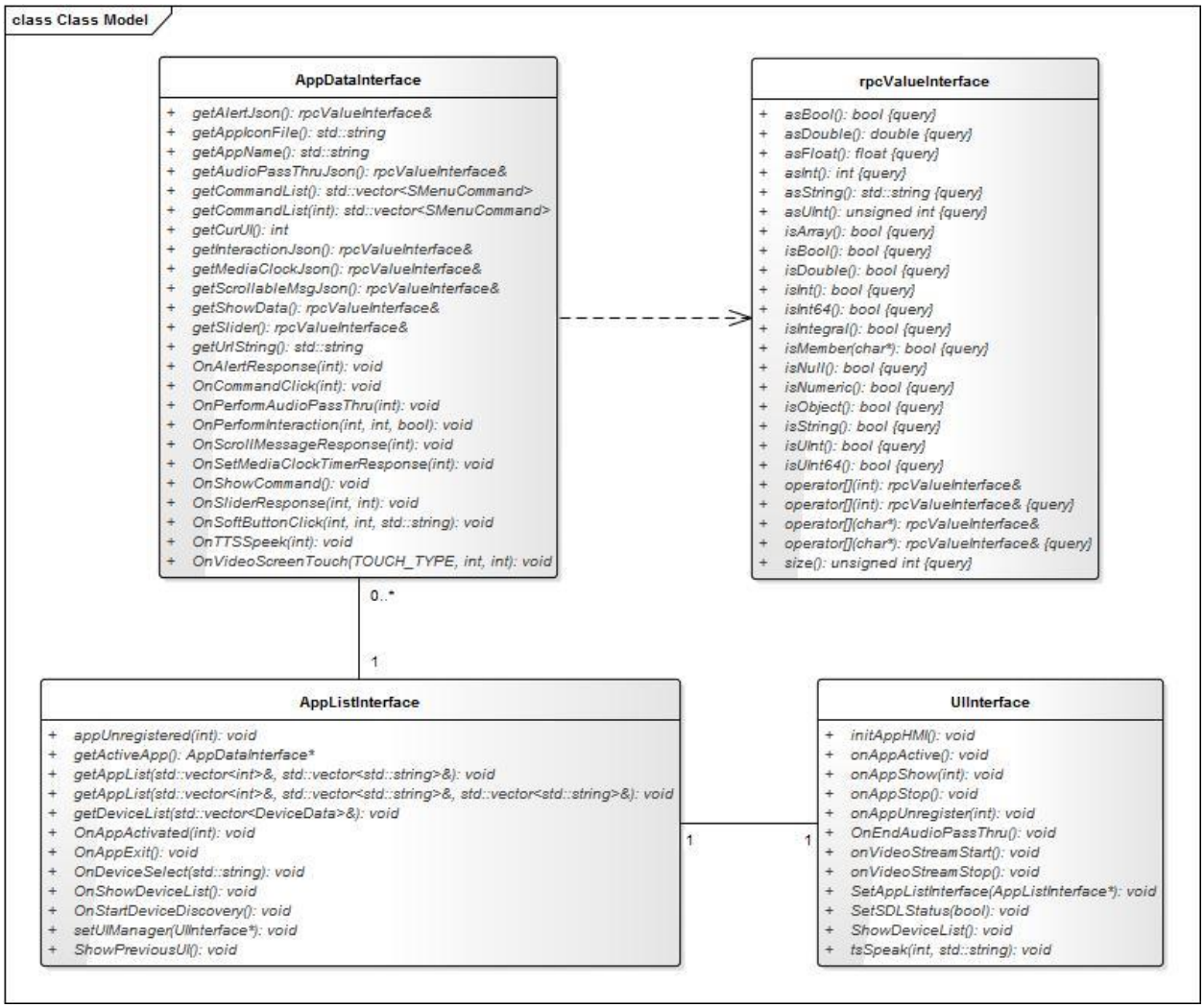
《HMI UI 设计指导说明书》

2 框架结构

2.1 模块结构

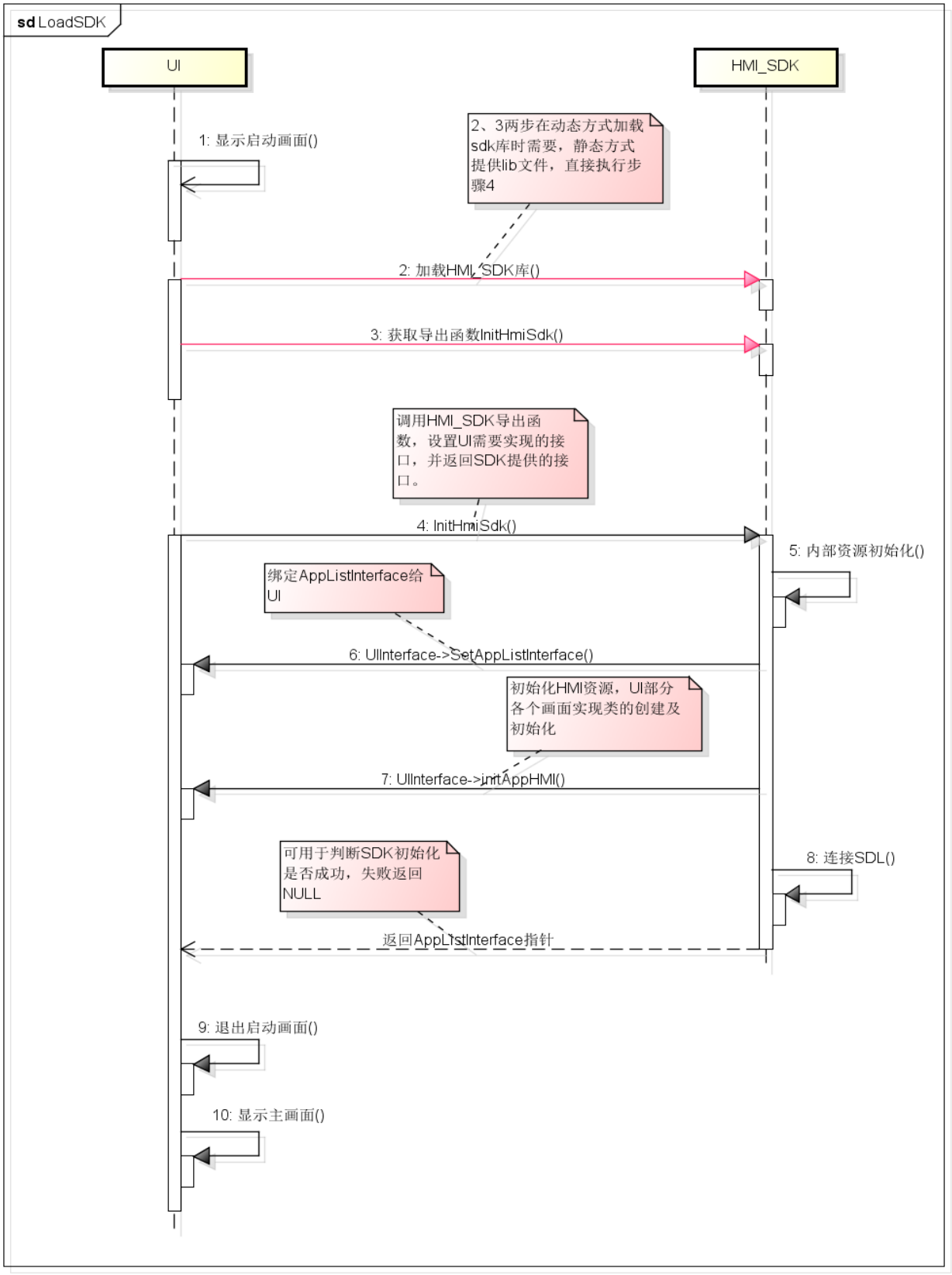


2.2 接口关系

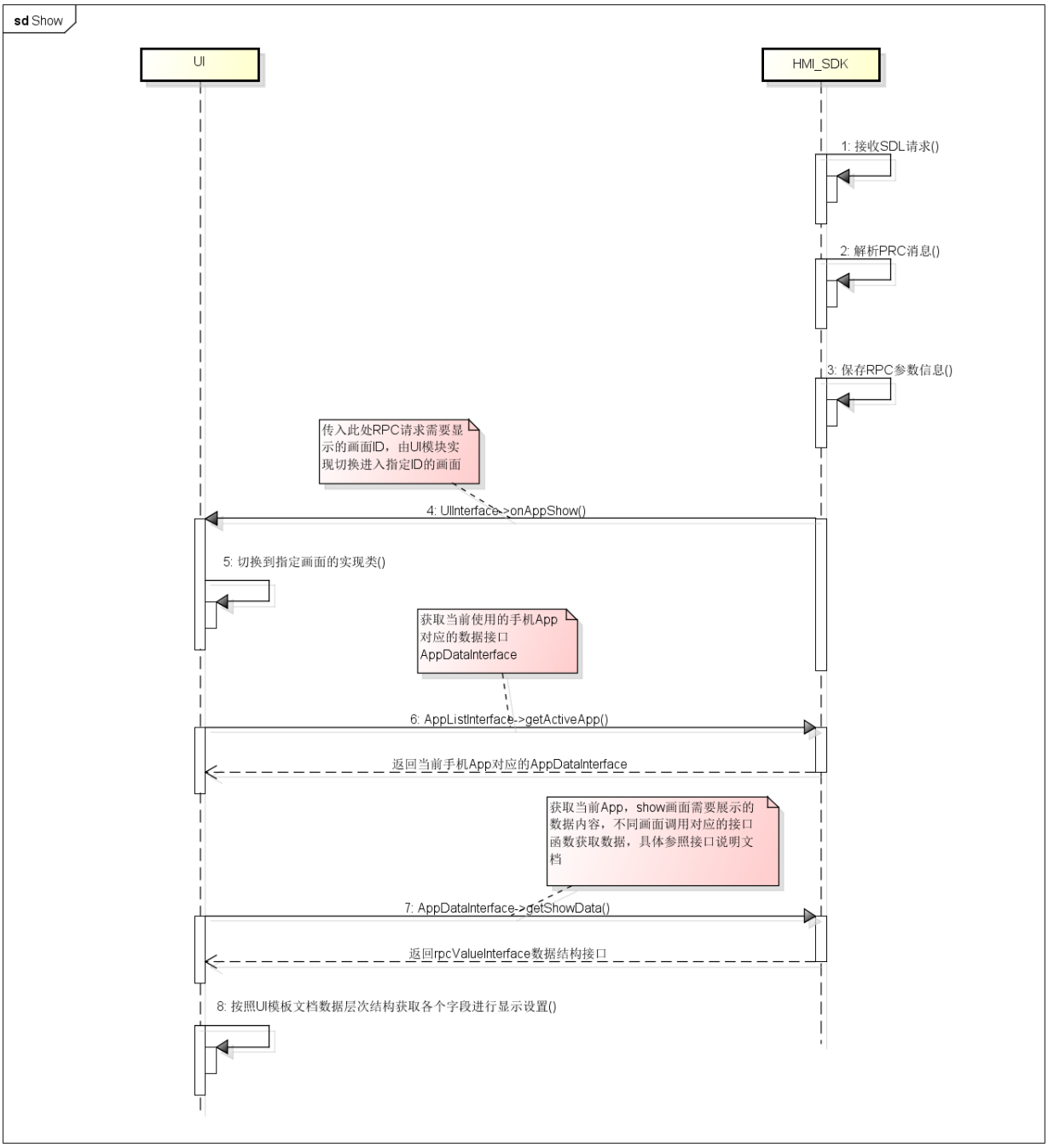


3 时序图

3.1 初始化



3.2 Show 画面请求



4 导出函数

4.1 作用

HMI SDK 动态库导出函数，当调用者通过库的方式使用 HMISDK 时，需要加载库，并通过动态库导出函数初始化及释放 SDK 资源，具体使用流程参照时序图 3.1 见上方

4.2 HMI SDK 初始化

HMISDK_EXPORT AppListInterface* InitHmiSdk (UIInterface * *pUI*)

描述

HMI SDK初始化

参数:

in	*pUI*	UIInterface 接口

返回:

返回值AppListInterface可用于判断SDK初始化是否成功，失败返回NULL

参见:

UIInterface.h AppListInterface.h

注解:

初始化SDK及HMI资源，并连接SDL

4.3 HMI SDK 释放

HMISDK_EXPORT void UnInitHmiSdk ()

HMI SDK释放

返回:

无

参见:

注解:

释放App数据管理等资源

4.4 使用示例 1

4.4.1 场景说明

在 main 函数中 UI 模块初始化，需要使用 HMISDK 库完成与 SDL 用户交互时，在 UI 端首先弹出启动画面，期间 loadsdk 函数中调用 InitHmiSdk 加载 SDK 库，关联 UI 与 SDK 的相关接口，在 InitHmiSdk 返回关联完成后退出启动画面，具体时序请参考 3.1 见上方（具体代码请参考源码根目

录下 testSDK 中子工程实现）

4.4.2 相关代码

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

#if (QT_VERSION <= QT_VERSION_CHECK(5,0,0))
    QTextCodec *codec=QTextCodec::codecForName("UTF-8");
    QTextCodec::setCodecForLocale(codec);
    QTextCodec::setCodecForCStrings(codec);
    QTextCodec::setCodecForTr(codec);
#else
    QTextCodec *codec=QTextCodec::codecForName("UTF-8");
    QTextCodec::setCodecForLocale(codec);
#endif

    g_pUIManager = new CGen3UIManager;
    QTimer::singleShot(500,g_pUIManager,SLOT(loadsdk()));

    QDialog diaStart;
#ifdef ANDROID

diaStart.setGeometry(0,0,QApplication::desktop()->width(),QApplication::de
sktop()->height()-30);
#else
    diaStart.setGeometry(0,0,800,480);
#endif
    diaStart.setStyleSheet("border-image:url(:/images/Screen.png);");
    QObject::connect(g_pUIManager,SIGNAL(finishLoadSDK()),&diaStart,
SLOT(accept()));

    // 启动画面，等待 HMISDK 初始化完成
    diaStart.exec();

    g_pUIManager->onAppShow(ID_MAIN);
    g_pUIManager->onAppShow(ID_APPLINK);

    return a.exec();
```

```
}

void CGen3UIManager::loadsdk()
{
    std::string strFilePath = GetSDKLibPath();
    strFilePath += "hmi_sdk";
    m_sdk.setFileName(strFilePath.c_str());

    // 初始化 HMISDK，动态调用 InitHmiSdk 函数
    InitFunc Init = (InitFunc)m_sdk.resolve("InitHmiSdk");
    if (Init) {
        AppListInterface * pApp = Init(this);
    } else {
        LOGE("can't load hmi sdk lib, %s", strFilePath.data());
    }

    // 通知初始化完成
    emit finishLoadSDK();
}
```

4.5 使用示例 2

4.5.1 场景说明

在 UI 模块退出，需要释放 HMISDK 库及相关资源，在 CGen3UIManager 类析构时，调用 UnInitHmiSdk 函数实现（具体代码请参考源码根目录下 testSDK 中子工程实现）

4.5.2 相关代码

```
CGen3UIManager::~CGen3UIManager()
{
    std::string strFilePath = GetSDKLibPath();
    strFilePath += "hmi_sdk";
    m_sdk.setFileName(strFilePath.c_str());

    // 释放 HMISDK，动态调用 UnInitHmiSdk 函数
    CloseFunc unInit = (CloseFunc)m_sdk.resolve("UnInitHmiSdk");
    if (unInit) {
        unInit();
    } else {
```

```
        LOGE("can't load hmi sdk lib, %s", strFilePath.data());
    }

    for (int i = 0; i < ID_UI_MAX; ++i) {
        if (m_vUIWidgets[i]) {
            delete m_vUIWidgets[i];
            m_vUIWidgets[i] = NULL;
        }
    }
}
```

5 HMI_SDK 提供的接口

5.1 AppDataInterface

5.1.1 接口说明

手机端 App 数据接口，手机端每一个 App 的数据都有一个该接口的实例进行保存，数据包含各个 RPC 请求在 HMI 端需要显示的属性数据，比如 Show、Alert 请求中的画面元素显示的内容等，在 UI 需要使用这些数据时，通过该接口中对应的方法取得，具体每种数据获取的方法，请参见该接口中函数的说明。

5.1.2 成员函数说明

virtual rpcValueInterface& AppDataInterface::getAlertJson ()

获取Alert画面需要的rpc数据

返回:

rpcValueInterface&

参见:

注解:

virtual std::string AppDataInterface::getAppIconFile ()

获取当前手机端App图标存放路径

返回:

std::string 当前手机端App图标存放路径

参见:

注解:

virtual std::string AppDataInterface::getAppName ()

获取当前手机端App名称

返回:

std::string 当前手机端App名称

参见:

注解:

virtual rpcValueInterface& AppDataInterface::getAudioPassThruJson ()

获取AudioPassThru画面需要的rpc数据

返回:

rpcValueInterface&

参见:

注解:

virtual std::vector<SMenuCommand> AppDataInterface::getCommandList ()

获取所有菜单命令项数据

返回:

std::vector<SMenuCommand>

参见:

注解:

virtual std::vector<SMenuCommand> AppDataInterface::getCommandList (int *subMenuID*)

获取指定菜单下所有菜单命令项数据

参数:

in	*subMenuID*	菜单标识

返回:

std::vector<SMenuCommand>

参见:

注解:

virtual int AppDataInterface::getCurUI ()

获取当前手机端App显示画面的ID

返回:

int 当前手机端App显示画面的ID

参见:

AppCommon.h ShowType枚举值定义

注解:

virtual rpcValueInterface& AppDataInterface::getInteractionJson ()

获取ChoiceSet画面需要的rpc数据

返回:

rpcValueInterface&

参见:

注解:

virtual rpcValueInterface& AppDataInterface::getMediaClockJson ()

获取MediaShow画面需要的rpc数据

返回:

rpcValueInterface&

参见:

注解:

Show画面在media模式时，可通过SetMediaClockTimer rpc请求设置媒体播放进度栏

virtual rpcValueInterface& AppDataInterface::getScrollableMsgJson ()

获取ScrollableMessage画面需要的rpc数据

返回:

rpcValueInterface&

参见:

注解:

virtual rpcValueInterface& AppDataInterface::getShowData ()

获取Show画面需要的rpc数据

返回:

rpcValueInterface&

参见:

注解:

virtual rpcValueInterface& AppDataInterface::getSlider ()

获取Slider画面需要的rpc数据

返回:

rpcValueInterface&

参见:

注解:

virtual std::string AppDataInterface::getUrlString ()

获取VideoStream数据源地址

返回:

std::string 视频流url源地址

参见:

注解:

保留

virtual void AppDataInterface::OnAlertResponse (int *reason*)

Alert画面执行的结果通知

参数:

in	*reason*	执行结果，参照 RESULT_CODE

返回:

无

参见:

AppCommon.h RESULT_CODE

注解:

将Alert画面执行的结果通知给SDK，然后回到前一画面

virtual void AppDataInterface::OnCommandClick (int *cmdID*)

菜单命令项点击通知

参数:

in	*cmdID*	菜单命令项 ID

返回:

无

参见:

注解:

用户在菜单命令画面点击菜单命令项时，调用SDK该函数将通知传递给手机端App

virtual void AppDataInterface::OnPerformAudioPassThru (int *code*)

录音执行的结果通知

参数:

in	*code*	执行结果

返回:

无

参见:

ProtocolDefines.h

注解:

将录音执行的结果通知给SDK，该函数保留

virtual void AppDataInterface::OnPerformInteraction (int *code*, int *choiceID*, bool *bVR* = `false`)

ChoiceSet执行的结果通知

参数:

in	*code*	执行结果

返回:

无

参见:

ProtocolDefines.h

注解:

将ChoiceSet执行的结果通知给SDK，该函数保留

virtual void AppDataInterface::OnScrollMessageResponse (int *reason*)

ScrollMessage画面执行的结果通知

参数:

in	*reason*	执行结果，参照 RESULT_CODE

返回:

无

参见:

AppCommon.h RESULT_CODE

注解:

将ScrollMessage画面执行的结果通知给SDK，然后回到前一画面

virtual void AppDataInterface::OnSetMediaClockTimerResponse (int *iCode*)

SetMediaClockTimer执行的结果通知

参数:

in	*code*	执行结果，参照 RESULT_CODE

返回:

无

参见:

AppCommon.h RESULT_CODE

注解:

将SetMediaClockTimer执行的结果通知给SDK

virtual void AppDataInterface::OnShowCommand ()

请求展示当前App菜单列表画面

返回:

无

参见:

AppCommon.h ShowType枚举值定义

注解:

会调用显示ID_COMMAND画面

virtual void AppDataInterface::OnSliderResponse (int *code*, int *sliderPosition*)

Slider画面执行的结果通知

参数:

in	*reason*	执行结果，参照 RESULT_CODE

返回:

无

参见:

AppCommon.h RESULT_CODE

注解:

将Slider画面执行的结果通知给SDK，然后回到前一画面

virtual void AppDataInterface::OnSoftButtonClick (int *sbID*, int *mode*, std::string *strName* = "")

软按钮点击通知

参数:

in	*sbID*	软按钮 ID
in	*mode*	点击模式
in	*strName*	软按钮名称

返回:

无

参见:

ProtocolDefines.h BUTTON_SHORT BUTTON_LONG

注解:

用户点击画面软按钮时，调用SDK该函数将通知传递给手机端App

virtual void AppDataInterface::OnTTSSpeek (int *code*)

TTSSpeek执行的结果通知

参数:

in	*code*	执行结果

返回:

无

参见:

ProtocolDefines.h

注解:

将TTSSpeek执行的结果通知给SDK，该函数保留

virtual void AppDataInterface::OnVideoScreenTouch (TOUCH_TYPE *touch*, int *x*, int *y*)

VideoStream画面点击移动操作通知

参数:

in	*touch*	软按钮 ID
in	*x*	鼠标操作的屏幕坐标 x 轴位置
in	*y*	鼠标操作的屏幕坐标 y 轴位置

返回:

无

参见:

ProtocolDefines.h TOUCH_TYPE

注解:

VideoStream画面点击移动操作时通知SDK，调用SDK该函数将通知传递给手机端App

virtual int AppDataInterface::getAppID ()

获取当前手机端App的ID

返回:

int 当前手机端App的ID

参见:

注解:

5.2 AppListInterface

5.2.1 接口说明

手机端 App 数据控制接口，保存所有手机端 App 数据列表（参见 AppDataInterface 说明），作为 UI 部分控制指定 App 的总入口，对于公用部分的 RPC 请求也会在此处理，指定 App 的处理将会继续传递到下一层 AppDataInterface 中进行，具体方法请参见该接口中函数的说明。

5.2.2 成员函数说明

virtual void AppListInterface::appUnregistered (int *appId*)

通知取消App注册

参数:

in	*appId*	app 对应的 id

返回:

无

参见:

注解:

App取消注册时会触发该事件，SDK会将该App数据销毁，并将画面切换到App列表

virtual AppDataInterface* AppListInterface::getActiveApp ()

获取当前AppDataInterface数据接口

返回:

AppDataInterface* 当前AppDataInterface数据接口

参见:

注解:

在展示某个画面时，需要获取当前活动App，然后取得画面的数据

virtual void AppListInterface::getAppList (std::vector< int > & *vAppIDs*, std::vector< std::string > & *vAppNames*)

获取所有App信息

参数:

out	*vAppIDs*	App 标识列表
out	*vAppNames*	App 显示名称列表

返回:

无

参见:

注解:

获取所有App标识和名称

virtual void AppListInterface::getAppList (std::vector< int > & *vAppIDs*, std::vector< std::string > & *vAppNames*, std::vector< std::string > & *vIconPath*)

获取所有App信息

参数:

out	*vAppIDs*	App 标识列表
out	*vAppNames*	App 显示名称列表
out	*vIconPath*	App 显示图标列表

返回:

无

参见:

注解:

获取所有App标识、名称及图标

virtual void AppListInterface::getDeviceList (std::vector< DeviceData > & *vDevice*)

获取当前连接所有设备信息

参数:

out	*vDevice*	当前连接所有设备信息

返回:

无

参见:

DeviceData结构定义

注解:

virtual void AppListInterface::OnAppActivated (int *appID*)

通知用户画面中选中某个App

参数:

in	*appID*	App 标识

返回:

无

参见:

注解:

通知SDK用户在App列表中点击某个App

virtual void AppListInterface::OnAppExit ()

通知退出当前App

返回:

无

参见:

注解:

通知SDK用户在画面点击退出App菜单等操作

virtual void AppListInterface::OnDeviceSelect (const std::string *id*)

通知选择了设备并要求获取设备支持的App信息

参数:

in	*id*	设备 ID

返回:

无

参见:

注解:

用户点击画面选择某个设备触发该事件，通知选择了设备并要求获取设备支持的App信息

virtual void AppListInterface::OnShowDeviceList ()

通知需要获取设备列表

返回:

无

参见:

注解:

用户点击画面获取设备列表触发该事件，通知SDK获取设备列表

virtual void AppListInterface::OnStartDeviceDiscovery ()

请求SDK扫描设备信息

返回:

无

参见:

注解:

通知SDK扫描获取有效的设备

virtual void AppListInterface::setUIManager (UIInterface * *pUI*)

设置UIInterface接口

参数:

in	*pUI*	UIInterface 接口

返回:

无

参见:

注解:

该函数会在SDK初始化时调用进行设置，保证后续AppListInterface对该接口使用有效

virtual void AppListInterface::ShowPreviousUI ()

通知切换到前一画面

返回:

无

参见:

注解:

通知SDK需要切换到前一画面或App列表画面

5.3 rpcValueInterface

5.3.1 接口说明

RPC 数据读取接口，该接口实际上是对 Json 对象的封装。HMISDK 接收 SDL 发送过来的 RPC 数据，并将数据转换为 Json 对象，最终封装为 rpcValueInterface 接口，在 UI 调用端需要显示 SDL 发送过

来的 RPC 数据时，提供以统一、只读方式获取数据，因此该接口暴露 Json 对象内部数据获取、对内部数据提供按照索引及关键字的方式获取内部数据、数据转型及判断等函数功能。

5.3.2 成员函数说明

virtual bool rpcValueInterface::asBool () const

将当前rpcValueInterface对象转换为bool类型

返回:

rpcValueInterface数据值

参见:

注解:

virtual double rpcValueInterface::asDouble () const

将当前rpcValueInterface对象转换为double类型

返回:

rpcValueInterface数据值

参见:

注解:

virtual float rpcValueInterface::asFloat () const

将当前rpcValueInterface对象转换为float类型

返回:

rpcValueInterface数据值

参见:

注解:

virtual int rpcValueInterface::asInt () const

将当前rpcValueInterface对象转换为int类型

返回:

rpcValueInterface数据值

参见:

注解:

virtual std::string rpcValueInterface::asString () const

将当前rpcValueInterface对象转换为std::string类型

返回:

rpcValueInterface数据值

参见:

注解:

virtual unsigned int rpcValueInterface::asUInt () const

将当前rpcValueInterface对象转换为unsigned int类型

返回:

rpcValueInterface数据值

参见:

注解:

virtual bool rpcValueInterface::isArray () const

判断当前rpcValueInterface对象是否为数组类型

返回:

为数组类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isBool () const

判断当前rpcValueInterface对象是否为bool类型

返回:

为bool类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isDouble () const

判断当前rpcValueInterface对象是否为double类型

返回:

为double类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isInt () const

判断当前rpcValueInterface对象是否为int类型

返回:

为int类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isInt64 () const

判断当前rpcValueInterface对象是否为64位int类型

返回:

为64位int类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isIntegral () const

判断当前rpcValueInterface对象是否为整数类型

返回:

为整数类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isMember (const char * *key*) const

判断是否存在某个字段

参数:

in	*key*	字段名，空字符结尾

返回:

如果对象具有指定字段名的成员，则返回true

参见:

注解:

virtual bool rpcValueInterface::isNull () const

判断当前rpcValueInterface对象是否为空类型

返回:

为空类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isNumeric () const

判断当前rpcValueInterface对象是否为数字类型

返回:

为数字类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isObject () const

判断当前rpcValueInterface对象是否为对象类型

返回:

为对象类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isString () const

判断当前rpcValueInterface对象是否为字符串类型

返回:

为字符串类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isUInt () const

判断当前rpcValueInterface对象是否为unsigned int类型

返回:

为unsigned int类型，则返回true

参见:

注解:

virtual bool rpcValueInterface::isUInt64 () const

判断当前rpcValueInterface对象是否为64位unsigned int类型

返回:

为64位unsigned int类型，则返回true

参见:

注解:

virtual rpcValueInterface& rpcValueInterface::operator[] (int *index*)

通过基于零的索引访问数组元素

参数:

in	*index*	文件名长度

返回:

rpcValueInterface& rpc数据结构

参见:

注解:

在数组大小为索引+ 1，如果数组包含元素数小于索引，则会返回null

virtual const rpcValueInterface& rpcValueInterface::operator[] (int *index*) const

通过基于零的索引访问数组元素

参数:

in	*index*	文件名长度

返回:

rpcValueInterface& rpc数据结构

参见:

注解:

在数组大小为索引+ 1，如果数组包含元素数小于索引，则会返回null，返回值不能修改

virtual rpcValueInterface& rpcValueInterface::operator[] (const char * *key*)

通过字段关键字访问对象元素

参数:

in	*key*	字段关键字字符串

返回:

rpcValueInterface& rpc数据结构

参见:

注解:

如果不存在，则创建空成员，关键字被限制为2^30-1个字符，超过该值将导致异常

virtual const rpcValueInterface& rpcValueInterface::operator[] (const char * *key*) const

通过字段关键字访问对象元素

参数:

in	*key*	字段关键字字符串

返回:

rpcValueInterface& rpc数据结构

参见:

注解:

如果不存在，则创建空成员，关键字被限制为2^30-1个字符，超过该值将导致异常，返回值不能修改

virtual unsigned int rpcValueInterface::size () const

获取对象或数组的元素个数

返回:

对象或数组的元素个数

参见:

注解:

6 调用侧实现的接口

6.1 rpcValueInterface

6.1.1 接口说明

调用侧按照要求需要实现的接口，用于控制画面的显示部分

6.1.2 成员函数说明

virtual void UIInterface::initAppHMI ()

初始化UI画面部分

返回:

无

参见:

注解:

需要实现初始化HMI资源，UI部分各个画面实现类的创建及初始化，保证后续对画面的操作有效

virtual void UIInterface::onAppActive ()

通知激活App

返回:

无

参见:

注解:

App列表中点击App图标会触发该事件

virtual void UIInterface::onAppShow (int *type*)

通知显示指定画面

参数:

in	*type*	画面 ID

返回:

无

参见:

AppCommon.h ShowType枚举值定义

注解:

切换到指定画面实现类实例进行显示

virtual void UIInterface::onAppStop ()

通知停止App

返回:

无

参见:

注解:

退出App时会触发该事件

virtual void UIInterface::onAppUnregister (int *appId*)

通知取消App注册

参数:

in	*appId*	app 对应的 id

返回:

无

参见:

注解:

App取消注册时会触发该事件

virtual void UIInterface::OnEndAudioPassThru ()

通知停止语音录音

返回:

无

参见:

注解:

停止语音录音时会触发该事件

virtual void UIInterface::onVideoStreamStart ()

通知开启视频流传输

返回:

无

参见:

注解:

开启视频流传输时会触发该事件

virtual void UIInterface::onVideoStreamStop ()

通知停止视频流传输

返回:

无

参见:

注解:

停止视频流传输时会触发该事件

virtual void UIInterface::SetAppListInterface (AppListInterface * *pList*)

设置AppListInterface接口

参数:

in	*pList*	AppListInterface 接口

返回:

无

参见:

注解:

该函数会在SDK初始化时调用进行设置，保证后续UIInterface对该接口使用有效

virtual void UIInterface::SetSDLStatus (bool *bConnect*)

通知当前SDL的连接状态

参数:

in	*bConnect*	连接状态，是否连接

返回:

无

参见:

注解:

virtual void UIInterface::ShowDeviceList ()

通知刷新设备列表

返回:

无

参见:

注解:

收到刷新设备列表通知时会触发该事件

virtual void UIInterface::tsSpeak (int *VRID*, std::string *strText*)

通知tts语音播放

参数:

in	*VRID*	请求类型

in	*strText*	语音播放文本

返回:

无

参见:

AppCommon.h VRID枚举值定义

注解:

实现TTS语音文本的转换并进行播放

7 数据结构

7.1 command

7.1.1 结构说明

菜单命令项信息

7.1.2 成员字段说明

int command::i_appID

App ID

int command::i_cmdID

cmd ID

unsigned char command::i_ImageType

菜单命令项图标类型

int command::i_menuID

菜单 ID

int command::i_parentID

父菜单 ID

int command::i_position

菜单命令项所在的位置

std::string command::str_ImagePath

菜单命令项图标路径

std::string command::str_menuName

菜单名

7.2 DeviceData

7.2.1 结构说明

设备信息

7.2.2 成员字段说明

std::string DeviceData::id

设备id

std::string DeviceData::name

设备名

8 枚举类型说明

8.1 enum ShowType

画面ID枚举类型

枚举值:

ID_APPLINK	App列表
ID_DEVICEVIEW	设备列表
ID_CHOICESET	ChoiceSet画面
ID_COMMAND	菜单画面
ID_SHOW	Show画面
ID_ALERT	Alert画面
ID_AUDIOPASSTHRU	录音画面
ID_CHOICESETVR	保留
ID_SCROLLMSG	ScrollMessage画面
ID_SLIDER	Slider画面

ID_NOTIFY	保留
ID_MEDIACLOCK	MediaClockTimer画面
ID_VIDEOSTREAM	VideoStream画面
ID_MAIN	Main画面
ID_UI_MAX	

8.2 enum TOUCH_TYPE

屏幕操作枚举类型

枚举值:

TOUCH_START	按下
TOUCH_MOVE	移动
TOUCH_END	弹起

9 宏定义

9.1 Alert 画面执行状态

#define **ALERT_TIMEOUT** 0 *超时*
#define **ALERT_CLICK_SOFTBUTTON** 1 *点击软按钮*
#define **ALERT_ABORTED** 2 *中断*

9.2 ScrollMessage 画面执行状态

#define **SCROLLMESSAGE_TIMEOUT** 0 *超时*
#define **SCROLLMESSAGE_CLICK_SOFTBUTTON** 1 *点击软按钮*
#define **SCROLLMESSAGE_REJECTED** 2 *拒绝*

9.3 按钮按下模式

#define **BUTTON_SHORT** 0 *短按*
#define **BUTTON_LONG** 1 *长按*

9.4 ChoiceSet 画面执行状态

#define **PERFORMINTERACTION_TIMEOUT** 10 *超时*
#define **PERFORMINTERACTION_CHOICE** 0 *选择VR命令*

9.5 Slider 画面执行状态

#define **SLIDER_OK** 0 *执行成功*
#define **SLIDER_TIMEOUT** 10 *超时*
#define **SLIDER_ABORTED** 5 *中断*

9.6 录音画面执行状态

#define **PERFORMAUDIOPASSTHRU_TIMEOUT** 0
#define **PERFORMAUDIOPASSTHRU_RETYP** 7
#define **PERFORMAUDIOPASSTHRU_DONE** 0
#define **PERFORMAUDIOPASSTHRU_CANCEL** 5

9.7 TTS 执行状态

#define **SPEEK_OK** 0
#define **SPEEK_INTERRUPTED** 5

9.8 导入导出函数宏定义

```
#if defined(WIN32) || defined(WINCE)
#ifdef HMISDK_LIB
#define HMISDK_EXPORT __declspec(dllexport)
#else
#define HMISDK_EXPORT __declspec(dllimport)
#endif
#else
#define HMISDK_EXPORT
#endif
```

仅在 windows 平台需要明确声明函数导入导出关键词，编译 HMISDK 库时，需要导出，使用库时，要声明为导入。

10 接口用法示例

10.1 Alert 画面显示

10.1.1 功能说明

HMISDK 获取 SDL 端 Alert 画面展示请求，请求最终传递到 UI 端 AlertView 画面实现类，在类 showEvent 事件处理函数中，获取当前活动 App 对应的接口引用（AppDataInterface），通过该接口方法获取 App 对应的名字（getAppName），展示数据（getAlertJson）等。

10.1.2 原始 rpc 数据

名字	请求内容

UI.Alert	{
	"id": 49,
	"jsonrpc": "2.0",
	"method": "UI.Alert",
	"params": {
	"alertStrings": [
	{
	"fieldName": "alertText1",
	"fieldText": "Alert Line 1"
	},
	{
	"fieldName": "alertText2",
	"fieldText": "Alert Line 2"
	},
	{
	"fieldName": "alertText3",
	"fieldText": "Alert Line 3"
	}
	],
	"alertType": "BOTH",
	"appID": 12077,
	"duration": 0,
	"progressIndicator": true,

```
        "softButtons": [
            {
                "isHighlighted": true,
                "softButtonID": 5552,
                "systemAction": "DEFAULT_ACTION",
                "text": "Close",
                "type": "TEXT"
            },
            {
                "isHighlighted": true,
                "softButtonID": 5553,
                "systemAction": "DEFAULT_ACTION",
                "text": "Close",
                "type": "TEXT"
            },
            {
                "isHighlighted": true,
                "softButtonID": 5554,
                "systemAction": "DEFAULT_ACTION",
                "text": "Close",
                "type": "TEXT"
            }
        ]
    }
}
```

10.1.3 关键代码实现

```
AppListInterface * m_pList;
#define AppControl m_pList->getActiveApp()

void AlertView::showEvent(QShowEvent * e)
{
    int iCount = 0;
    if (AppControl) {

AppBase::SetEdlidedText(m_pAppNameLab,AppControl->getAppName().c_str(),
                            width()*0.9);
```

```
        // 获取 Alert 画面 Rpc 数据接口
        rpcValueInterface& pObj = AppControl->getAlertJson();

        for (int i = 0;i != 3;++i) {
            m_aAlertLab[i].setText("");
        }

        for (int i = 0;i != 6;++i) {
            m_aSoftBtn[i].SetSize(0,0);
            m_aSoftBtn[i].SetText("");
        }


    // 判断 alertStrings 字段是否存在
    if (pObj["params"].isMember("alertStrings")) {
        // 获取 alertStrings 字段数组类型元素个数
        iCount = pObj["params"]["alertStrings"].size() >
                3 ? 3 : pObj["params"]["alertStrings"].size();
        for (int i = 0; i < iCount; ++i) {
            // 获取 alertStrings 每个元素中 fieldText 对应的数值
            AppBase::SetEdlidedText(m_aAlertLab+i,

pObj["params"]["alertStrings"][i]["fieldText"].asString().c_str(),
                    width()*0.9);
        }
    }

        if (pObj["params"].isMember("softButtons")) {
            m_Timer.start(pObj["params"]["duration"].asInt() + 20000);
            iCount = pObj["params"]["softButtons"].size() >
                    6 ? 6 : pObj["params"]["softButtons"].size();

            for (int i = 0; i < iCount; ++i) {
                m_aSoftBtn[i].Init(SOFTBTNWIDTH,SOFTBTNHEIGHT,"",
                                   ":/images/alert_softbtn_normal.png",
                                   ":/images/alert_softbtn_press.png");

m_aSoftBtn[i].SetText(pObj["params"]["softButtons"][i]["text"].asString().
c_str());
```

```
m_aSoftBtn[i].SetId(pObj["params"]["softButtons"][i]["softButtonID"].asInt
());
            }
        } else {
            // 获取 duration 字段值并转换为整型
            m_Timer.start(pObj["params"]["duration"].asInt());
        }
    }
}
```

注：按照数据组织结构（参考文档《HMI SDK UI 设计指导说明书.docx》），一层一层获取数据值

10.1.4 显示效果

10.2 画面 Touch 事件通知

10.2.1 功能说明

在 VideoStream 画面展示时，用户对屏幕的操作包括按下、释放及移动操作通知给 SDL 端，首先取得当前活动 App 对应的接口引用（AppDataInterface），然后通过接口方法 OnVideoScreenTouch，将屏幕的

操作类型及坐标传递给 SDL。

10.2.2 关键代码实现

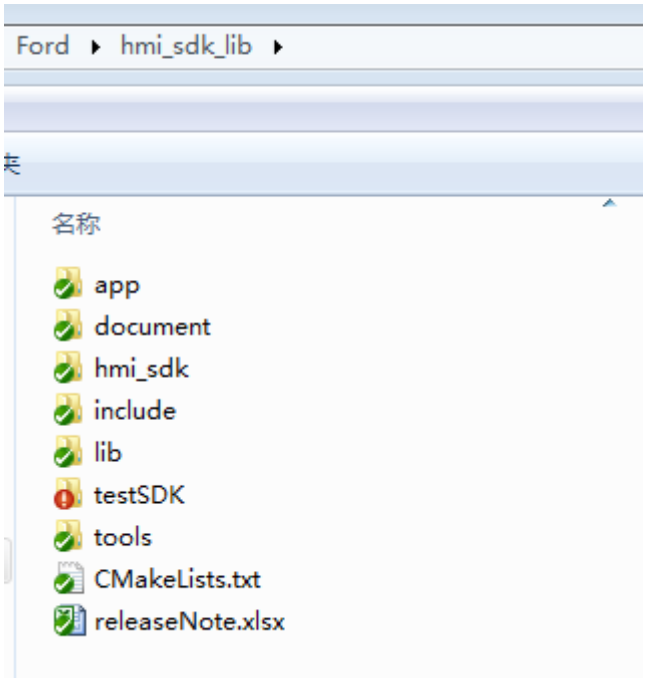
```
void CeVideoStream::mousePressEvent(QMouseEvent *e)
{
    int x = e->x();
    int y = e->y();
    x = x*videoWidth/width();
    y = y*videoHeight/height();
    m_pList->getActiveApp()->OnVideoScreenTouch(TOUCH_START,x,y);
}

void CeVideoStream::mouseMoveEvent(QMouseEvent *e)
{
    int x = e->x();
    int y = e->y();
    x = x*videoWidth/width();
    y = y*videoHeight/height();

    m_pList->getActiveApp()->OnVideoScreenTouch(TOUCH_MOVE,x,y);
}

#define ZOOMINBTNID 3
#define ZOOMOUTBTNID 4

void CeVideoStream::mouseReleaseEvent(QMouseEvent *e)
{
    int x = e->x();
    int y = e->y();
    x = x*videoWidth/width();
    y = y*videoHeight/height();

    m_pList->getActiveApp()->OnVideoScreenTouch(TOUCH_END,x,y);
}
```

注：以上完整代码，请参考源码根目录/testSDK 下的 QT 测试工程代码。

11 代码结构说明

11.1 目录结构说明

源码根目录情况如下所示



该目录下包含 hmi_sdk 和 Gen3UI 两个工程，根目录下 CmakeLists.txt 为 hmi_sdk 编译脚本，也是本次重构的重点，生成为 hmi_sdk 动态库文件，testSDK 目录下为 Gen3UI 工程，该工程主要实现画面内容显示，为 QT 库实现，需要 QT 开发环境进行编译，用于测试 hmi_sdk，另外，document 下为此项目相关的设计文件及文档类。