

HUNDSUN[®]

恒生行情云服务

SDK Demo 使用手册 (C++)



恒生电子股份有限公司

地址： 杭州市滨江区江南大道 3588 号恒生大厦 邮编：310053

网址： <http://www.hundsun.com>

客户服务电话： 0571-26695555 转 7，输入客户号和密码进入服务通道

客户服务传真： 0571-28823456


客服企业 QQ： 800088399

客户服务邮箱： service.hq@hundsun.com

版权所有 © 恒生电子股份有限公司 2014-2016。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

 **天堂恒生® 你我他®** 为恒生电子股份有限公司的注册商标。

注意

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。

目 录

目 录.....	I
1. 前 言.....	0
1.1. 概述	0
1.2. 产品版本.....	0
1.3. 读者对象.....	0
1.4. 修改记录.....	0
1.5. 排版约定.....	1
2. 快速开始.....	2
2.1. WINDOWS.32, WINDOWS.64	2
2.2. LINUX.32, LINUX.64	2
2.3. 配置文件.....	2
2.4. 基本测试流程	3
3. 文件列表.....	5
3.1. 行情 SDK 文件	5
3.2. 配置文件.....	5
3.3. 工程文件.....	5
3.4. 工具类文件.....	6
3.5. DEMO 实现文件.....	6
4. SDK 调用简介.....	7
4.1. SDK 说明文档	7
4.2. SDK 初始化	8
4.3. SDK 关闭	9
4.4. SDK 消息创建	10
4.5. SDK 消息发送	10
4.6. SDK 消息响应	11
4.7. SDK 消息解析	12
5. DEMO 流程简介	15
5.1. 框架简介.....	15
5.2. 流程图示意.....	17

5.3. 消息处理示意..... 21

6. 内存泄漏检测23

6.1. WINDOWS..... 23

6.2. LINUX 23

7. 注意事项.....23

1. 前 言

1.1. 概述

欢迎您使用恒生行情云服务，本文档描述了 SDK Demo（C++版）简要介绍和使用指南。

1.2. 产品版本

与本文档相对应的产品版本如下所示。

产品名称	产品版本
H5SDK C++ Demo	v1.0.0.0

1.3. 读者对象

本文档（本指南）主要适用于以下人员：

- 开发人员

1.4. 修改记录

文档版本	更改记录
01 (2016-09-05)	第一次发布

1.5.排版约定

格式

带尖括号“<>”表示操作员从终端输入的信息；带方括号“[]”表示人机界面、菜单名、数据表和字段名等。多级菜单用“/”隔开(方括号采用半角符号)，如[文件/新建/文件夹] 多级菜单表示[文件]菜单下的[新建]子菜单下的[文件夹]菜单项。

键盘操作约定


键盘操作约定格式	意义
加尖括号的宋体字符	表示键名或按钮名，如<Enter>, <Tab>, <Backspace>, <a>等分别表示回车、制表、退格、小写字母 a。
<键 1+键 2>	表示在键盘上同时按下几个键，如<Ctrl+Alt+A>表示同时按下“Ctrl”、“Alt”、“A”这三个键。
<键 1，键 2>	表示先按第一键，释放，再按第二键，如<Alt ， F>表示先按<Alt>键，释放后，再按<F>键。


鼠标操作约定

鼠标操作约定格式	意义
单击	快速按下并释放鼠标的的一个按钮。
双击	连续两次按下并释放鼠标的的一个按钮。
拖动	按住鼠标的的一个键不放，移动鼠标。

各类标志

本文档中还采用各种醒目标志来表示在操作过程中应该特别注意的地方，这些标志的意义如下：

 小心、注意、警告、危险：提醒操作中应该注意的事项。

 说明、提示、窍门、思考：对操作内容的描述进行必要的补充和说明。

2. 快速开始

2.1. Windows.32, Windows.64

将对应版本 sdk 的 dll 拷贝到 [c++\demo\bin] 下. 分别是[lib-win32]和[lib-win64]。
使用 vs2010 打开工程: [c++\demo\src\h5sdk_c++_demo\h5sdk_c++_demo.vcxproj]
修改配置文件: [c++\demo\bin\configure],具体见 2.4
编译, (生成的可执行文件默认放到了[c++\demo\bin]下)
运行。

2.2. Linux.32, Linux.64

进入工程所在路径
将对应 sdk 的 so 拷贝到 [c++\demo\bin] 下
<cd c++/demo/src/h5sdk_c++_demo>
<make -f Makefile32 或 Makefile64> (生成的可执行文件默认放到了[../bin]下)
<cd ../bin>
<vi configure> 配置文件,具体见 2.4
<./run.sh> 设置动态库路径, 然后执行 demo.

2.3. 配置文件

路径	意义
[c++/demo/bin/configure]	Windows.Linux 配置文件

定义项目	意义
user	用户名
pass	密码
app_key	恒生开放平台的服务凭证
app_secret	恒生开放平台的服务凭证
ip_1	第一组测试服务器 ip

定义项目	意义
port_1	第一组测试服务器 port
ip_2	第二组测试服务器 ip
port_2	第二组测试服务器 port
maxNumberTest	性能测试时，样本数

- user/pass, app_key/app_secret 请联系客服，注册恒生开放平台的应用获取。
- ip,port 也请联系客服获取。
- level_2 所用账号及服务器，也请联系客服获取。
- demo 中，ip_2 与 ip_1 设置成一样。(为了测试多组 session 切换)

2.4.基本测试流程

2.4.1. 连接服务器，并登陆。成功登陆后继续测试，否则退出程序。

```
启动SDK会话
H5SdkCallbackImpl::OnConnect:
    建立连接 服务器121.43.74.8:9999 本地192.168.1.102:38406
H5SdkCallbackImpl::OnSdkLogin:
    sdk登陆入口
    用户登录开始.
    登录成功.
回车 继续
```

2.4.2. 主菜单

```
-----
1: 测试速度
2: IP地址和切换测试
3: 断开连接测试
-----
4: 市场分类信息.同步
5: K Line.异步
6: 快照.异步
7: 分时.异步
8: 行情快照订阅.主推。<'q'键 退出>
-----
9: 逐笔成交.订阅.同步<'q'键 退出><level2,没账号就没响应>
a: 逐笔委托.订阅.同步<'q'键 退出><level2,没账号就没响应>
-----
b: 性能测试.K线.
c: 性能测试.快照.
-----
Q: logout, 退出
输入选项,回车确定
-----
```


编制部门	公共平台发展部
批准日期	2016/09/09

2.4.3. 按提示，做进一步的测试。以[8.行情快照]为例:

2.4.3.1. 输入<8>,进入行情快照的推送模式

```
开始订阅
推送中, 'q' 键 退出 !!!
订阅-同步返回:
      code=600570.SS error_no=0
count=1001

<code=600570.XSHG.ESA.M open_px=60010 close_px=0 last_px=60000>
      bid entrust_px=60000 entrust_amount=25300
      bid entrust_px=59990 entrust_amount=200
      bid entrust_px=59980 entrust_amount=200
      bid entrust_px=59960 entrust_amount=100
      bid entrust_px=59950 entrust_amount=100
      offer entrust_px=60020 entrust_amount=500
      offer entrust_px=60050 entrust_amount=700
      offer entrust_px=60060 entrust_amount=1300
      offer entrust_px=60070 entrust_amount=200
      offer entrust_px=60080 entrust_amount=900

1001<支>
```

2.4.3.2. 输入<q>,退出推送模式

```
-----
      订阅-异步返回:
      退订
      code=600570.SS error_no=0
count=1001
回车 继续
```

2.4.3.3. <enter>, 进入主菜单，等待下一个测试。

3. 文件列表

3.1. 行情 SDK 文件

路径	意义
[C++/include]	各版本 SDK 的头文件
[lib-linux64]	linux64 库文件
[lib-linux32]	linux32 库文件
[lib-win32]	win32 库文件
[lib-win64]	win64 库文件
[c++/demo/bin]	可执行路径。配置文件，demo 执行文件放在这里。各版本的动态库需要拷贝至此

3.2. 配置文件

路径	意义
[c++/demo/bin/configure]	各版本的配置文件

3.3. 工程文件

路径	意义
[c++\demo\src\h5sdk_c++_demo\h5sdk_c++_demo.sln]	Windows.vs2010.工程
[c++/demo/src/h5sdk_c++_demo/Makefile64]	Linux.64.工程
[c++/demo/src/h5sdk_c++_demo/Makefile32]	Linux.32.工程

3.4. 工具类文件

路径[c++/demo/src/MyUtil]	意义
[MyThread.h.cpp]	线程类
[MyEvent.h.cpp]	信号量
[MyAutoLock.h.cpp]	锁

3.5. Demo 实现文件

路径[c++/demo/src/h5sdk_c++_demo]	意义
[demo_main.cpp]	主线程入口
[keyboard_input.cpp]	键盘输入线程
[main_environment.h.cpp]	主线程，环境
[main_message_queue.h.cpp]	主线程，消息队列
[my_vld.h]	vld 内存泄漏检测.
[sdk_callback.h.cpp]	SDK 回调
[sdk_capability.h.cpp]	性能测试处理
[sdk_configure.cpp]	配置文件读取
[sdk_data_parser.h.cpp]	缓冲线程
[sdk_util.h.cpp]	SDK 应用，消息创建及发送，数据包解析

4.SDK 调用简介

4.1.SDK 说明文档

- 4.1.1. 参看[H5 行情服务协议(修订版).xls],里面包含了所有的行情 SDK 接口定义。
- 4.1.2. 以 login 为例,选中下面的[消息目录],找到 login,点击[H5SDK_MSG_LOGIN],跳转到 4.1.3

系统类消息		
消息宏定义	消息ID	消息英文名称
H5SDK MSG LOGIN	1001	login
H5SDK MSG LOGOUT	1002	logout
H5SDK MSG HEARTBEAT	1003	heartbeat
H5SDK SERVER INFO	1004	server_info
H5SDK MSG BATCH	1006	batch
H5SDK MSG REGISTER	1005	register
H5SDK MSG CHANGE USER INFO	1007	change_user_info
H5SDK MSG USER KICK OFF	1008	user_kick_off

- 4.1.3. 消息:[Login] 的具体定义:

- 4.1.3.1. [消息头]: 功能号:[1001] 消息宏定义[H5SDK_MSG_LOGIN]

消息名称	登入请求	版本号	1.0.0.0	更新日期	20130822
功能号域	1001	功能英文名	login	类型名称	REQUEST
消息宏	H5SDK_MSG_LOGIN				
操作角色					
描述	登录请求消息,已经注册用户一般是连接后的第一个请求。				

- 4.1.3.2. [请求参数]:第三个参数为例:[app_key], 消息宏[H5SDK_TAG_APP_KEY],类型[rawdata],是否可选[Y],备注[...].

- 消息宏[H5SDK_TAG_APP_KEY]: 当需要使用 app_key 时,用消息宏定义去指定。
- 域名[app_key]:点击域名,可以跳转到该参数的具体定义。比如,枚举值,说明,等等。
- 是否可选[Y]:表面,这是个可选的参数。可以不填入具体的值,有缺省值。如果是[N],那就必须填入一个具体的参数。

请求	域名	类型	数值范围	是否可选	备注
H5SDK_TAG_USER_IN	user info	rawdata		Y	用户信息(ison信息加密)
H5SDK_TAG_SDK_VER	sdk version	uint32		Y	SDK版本信息
H5SDK_TAG_APP_KEY	app key	rawdata		Y	应用识别符

4.1.3.3. [消息域定义]:右上[Navigate Back],点击，可以回到跳转前的位置。

										Navigate Back
字段英文名	字段宏	字段ID	域类型	元素类型	枚举说明	数组最大项数	最小值	最大值	字段中	
chi_spelling_grp	H5SDK_TAG_CHI_SPELLING_GRP	20122	sequence						拼音简称重负组	
user_info	H5SDK_TAG_USER_INFO	20123	rawdata						用户信息	
app_key	H5SDK_TAG_APP_KEY	20124	rawdata						应用识别符	
gamma_index	H5SDK_TAG_GAMMA_INDEX	20130	int32						gamma指标	

4.1.3.4. [应答参数]:

应答	域名	类型			是否可选	
H5SDK_TAG_ERROR_NO	error no	uint32			Y	错误号。使用H5PROTO_ENUM
H5SDK_TAG_ERROR_INFO	error info	bytevector			Y	错误信息
H5SDK_TAG_USER_INFO	user info	rawdata			Y	登录返回数据。用户信息 (
H5SDK_FINANCE_MIC_GRP	finance mic grp	sequence			Y	收费的交易所代码 (vip)
H5SDK_TAG_FINANCE_MIC	finance mic	bytevector				交易所代码
H5SDK_TAG_HEARTBEAT_INTERVAL	heartbeat interval	uint32		d	Y	心跳间隔
修改记录						

4.1.3.5. 总结

- 选择 excel 下部的[标签栏],找到具体的消息定义。
- 消息定义,包含：消息头，输入，应答。
- 点击域名，或备注，可以跳转到参数的具体定义，和补充说明。
- 具体的利用消息定义，实现消息的调用和应答解析，见下面章节。

4.2.SDK 初始化

```
bool HqSdkUtil::sdk_init(void)
{
    //////////////////////////////////////
    // 设置环境变量
    //////////////////////////////////////
    //设置 SDK 日志路径
    SetH5DataPath("h5data");
    //设置 SDK 日志路径
    SetH5LogPath("H5LOG");
    //设置日志级别配置路径
    SetSdkParamPath("SdkParamPath.json");

    //////////////////////////////////////
    // 会话创建
    //////////////////////////////////////
    m_pSessionOptions = CreateSessionOptions();
    // 设置会话选项 1
    m_pSessionOptions->SetHeartbeatTimeout(3);

    //设置两组 ip 和端口
    m_pSessionOptions->SetServerIp(m_strIp1.c_str());
```

```
m_pSessionOptions->SetServerPort(m_iPort1);

m_pSessionOptions->SetServerIp(m_strIp2.c_str(),1);
m_pSessionOptions->SetServerPort(m_iPort2,1);

// 创建会话
m_pSession = CreateSession(m_pSessionOptions);

m_pSession->SetAppInfo(m_strAppKey.c_str(), m_strAppSecret.c_str());

// 开始为所有会话设置回调
StartSetCallback();
//两个会话可使用同一个回调类，也可以使用不同的回调类
m_pSession->SetCallback(&m_h5SdkCallbackImpl);

//首次启动时，不会自动连接，需要后面手工连接[除非特别需要，不需要调用此函数]
//SetInitAutoConnectFlag(false);

//设置代理
//SetProxyOptions(SDKHTTPPROXY, "10.26.210.42", 808, "test", "test");
return true;
}

//Init 后，启动 session，开始处理行情事务。
StartAllSessions();
```

4.3.SDK 关闭

```
void HqSdkUtil::sdk_close(void)
{
    my_log("\t 关闭 sdk\n");
    StopAllSessions();
    if(m_pSessionOptions)
    {
        m_pSessionOptions->Release();
        m_pSessionOptions = NULL;
    }
    if(m_pSession)
    {
        m_pSession->Release();
        m_pSession = NULL;
    }
}
```

```
}  
}
```

4.4.SDK 消息创建

```
IHsCommMessage *request = m_pSession->CreateMessage(BIZ_H5PROTO,  
H5SDK_MSG_MARKET_TYPES, REQUEST);  
if (NULL == request)  
{  
    my_log("\t 消息创建失败\n");  
    return;  
}  
// 根据 H5SDK_FINANCE_MIC_GRP 字段类型为 sequence, 创建 IGroup 句柄  
IGroup *group = request->SetGroup(H5SDK_FINANCE_MIC_GRP);  
// 为 H5SDK_FINANCE_MIC_GRP 添加第一条记录  
IRecord *record = group->AddRecord();  
// 为第一条记录的字段 H5SDK_TAG_FINANCE_MIC 设置值"SS",  
// H5SDK_TAG_FINANCE_MIC 字段类型为 bytevector, 故调用 SetString, 传入字符串值  
record->GetItem(H5SDK_TAG_FINANCE_MIC)->SetString("SS");  
// 为 H5SDK_FINANCE_MIC_GRP 添加第二条记录  
record = group->AddRecord();  
// 为第二条记录的字段 H5SDK_TAG_FINANCE_MIC 设置值"SZ"  
record->GetItem(H5SDK_TAG_FINANCE_MIC)->SetString("SZ");  
my_log("\t 输入市场: SS SZ\n");
```

4.5.SDK 消息发送

4.5.1. 同步发送

```
IHsCommMessage *ansMsg=NULL;  
int ret = m_pSession->SyncCall(request,&ansMsg,3000);
```

4.5.2. 异步发送

- request->GetHead()->GetItem(H5PROTO_TAG_USER_KEY)->
SetRawData("aaa", strlen("aaa"));
//用户自定义KEY. 设置一个标识。当消息回调时, 可以从数据包里, 再取出这个KEY. 从而确认其来源。

```
m_pSession->AsyncSend(request); //异步调用, OnReceived 响应回调
```

```
//!!!释放.  
request->Release();
```

4.6.SDK 消息响应

4.6.1. 同步响应:

- 命令执行后，有响应才返回。
- 返回 0,成功。
- 返回 H5SDK_DISCONNECT, 就不应该再重试了。需要等待 Login 成功。
- 返回 H5SDK_SERIALIZATION_ERROR, 可能是编程出问题，需要 GetLastErrInfo 打出来分析下。
- 返回 H5SDK_TIMEOUT, 超时返回，应该重试。

```
if (ret ==0 && ansMsg)  
{  
    my_log("\t 市场分类.同步返回:\n");  
    parse_market_types(ansMsg);  
}  
else  
{  
    my_log("\t 同步请求错误 ret=%d。如果是超时，最好重试\r\n",ret);  
}  
//!!!释放.  
if(ansMsg)  
{  
    ansMsg->Release();  
}  
//!!!释放.  
request->Release();
```

4.6.2. 异步响应:

- 在SDK回调函数: `void H5SDKAPI H5SdkCallbackImpl::OnReceived(Session *session, IHsCommMessage *response)` 中处理。
- 同步，异步消息的响应数据包，完全一致。可以用同样的解析方法。
- 异步消息，可以通过“用户自定义Key”，来标识异步响应的来源。

```
void H5SDKAPI H5SdkCallbackImpl::OnReceived(Session *session,  
                                             IHsCommMessage *response)
```



```
{
    //代码里，是缓冲到缓冲队列里。这里，为了方便理解，整理成如下代码
    int headErrorNo = response->GetHead()->
        GetItem(H5PROTO_TAG_ERROR_NO)->GetInt32();

    if(headErrorNo != 0)
    { //错误处理
        //取第一个用户自定义 Key.
        int len = response->GetHead()->
            GetItem(H5PROTO_TAG_USER_KEY)->GetRawDataLen(0);
        //用户输入的 key.用于与异步命令一一对应。
        char *user_key = (char *)response->GetHead()->
            GetItem(H5PROTO_TAG_USER_KEY)->GetRawData(&len);
        my_log("\n\t%s: errNo=%d user_key=%s\n", __FUNCTION__,
            headErrorNo, user_key);
        response->Release();
        return;
    }

    int funcid = response->GetFunction();
    switch (funcid)
    {
    case H5SDK_MSG_MARKET_TYPES: // 市场分类信息
        my_log("\t 市场分类信息.异步返回:\n");
        MyEnvironment::GetHqSdkUtil()->parse_market_types(m_hsMsgPackage);
        break;
    }
    response->Release();
}
```

4.7.SDK 消息解析

```
void HqSdkUtil::parse_market_types(IHsCommMessage* response)
{
    // 因为字段 H5SDK_FINANCE_MIC_GRP 为 sequence, 先获取 IGroup 句柄
    IGroup *finance_mic_grp = response->GetGroup(H5SDK_FINANCE_MIC_GRP);
    if(finance_mic_grp != NULL)
    {
        // 获取 H5SDK_FINANCE_MIC_GRP 的记录条数
        int finance_mic_grp_cnt = finance_mic_grp->GetRecordCount();
        // 解析每一条记录
        for (int i = 0; i < finance_mic_grp_cnt; i++)
```

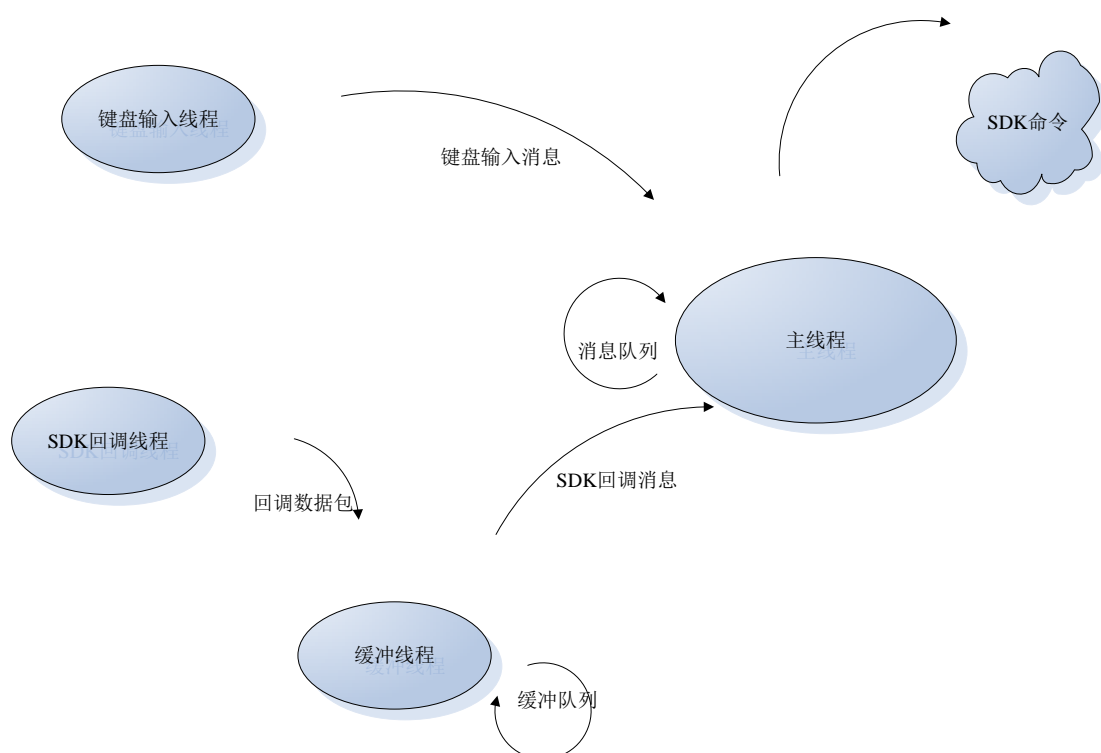
```
{
    // 获取记录的句柄
    IRecord *finance_mic_record = finance_mic_grp->GetRecord(i);
    // 根据各个字段的数据类型，调用不同函数获取字段值
    const char *finance_mic = finance_mic_record->
        GetItem(H5SDK_TAG_FINANCE_MIC)->GetString();
    const char *finance_name = finance_mic_record->
        GetItem(H5SDK_TAG_FINANCE_NAME)->GetString();
    uint32 market_date = finance_mic_record->
        GetItem(H5SDK_TAG_MARKET_DATE)->GetInt32();
    uint32 init_date = finance_mic_record->
        GetItem(H5SDK_TAG_INIT_DATE)->GetInt32();
    int32 timezone = finance_mic_record->
        GetItem(H5SDK_TAG_TIMEZONE)->GetInt32();
    const char *timezone_code = finance_mic_record->
        GetItem(H5SDK_TAG_TIMEZONE_CODE)->GetString();
    uint8 dst = finance_mic_record->
        GetItem(H5SDK_TAG_DST_FLAG)->GetInt8();

    string gbkName;
    UTF82GBK(finance_name, gbkName);
    my_log("\t\tmic=%s name=%s\n", finance_mic, gbkName.c_str());
    // 因为 H5SDK_TAG_TYPE_GRP 类型为 sequence，所以先必须获取 IGroup 句柄

    IGroup *type_grp = finance_mic_record->
        GetGroup(H5SDK_TAG_TYPE_GRP);
    if(type_grp)
    {
        // 获取 H5SDK_TAG_TYPE_GRP 的记录条数
        int type_grp_cnt = type_grp->GetRecordCount();
        // 解析每一条记录
        for (int j = 0; j < type_grp_cnt; j++)
        {
            // 获取记录的句柄
            IRecord *type_record = type_grp->GetRecord(j);
            // 根据各个字段的数据类型，调用不同函数获取字段值
            const char *hq_type_code = type_record->
                GetItem(H5SDK_TAG_HQ_TYPE_CODE)->GetString();
            const char *hq_type_name = type_record->
                GetItem(H5SDK_TAG_HQ_TYPE_NAME)->GetString();
            uint32 px_scale = type_record->
                GetItem(H5SDK_TAG_PRICE_SCALE)->GetInt32();
            int32 px_precision = type_record->
```


5. Demo 流程简介

5.1. 框架简介



5.1.1. 主线程

- 循环处理消息队列里的消息。
- 打印主菜单
- 往 SDK 里发命令。
- 主函数:[demo_main.cpp]:[main()]

5.1.2. 键盘输入线程

- 非阻塞模式获取键盘输入，传递消息给主线程的消息队列。
- 主函数:[keyboard_input.cpp]:[KeyboardInputThread::MainFun()]

5.1.3. SDK 回调线程

- 处理 session 连接的相关回调。OnClose, OnConnect, OnCore
- 异步命令的响应，推送消息的回调，直接送入缓冲线程的缓冲队列。
- 回调函数:[sdk_callback.cpp]:[H5SdkCallbackImpl::OnReceived()]

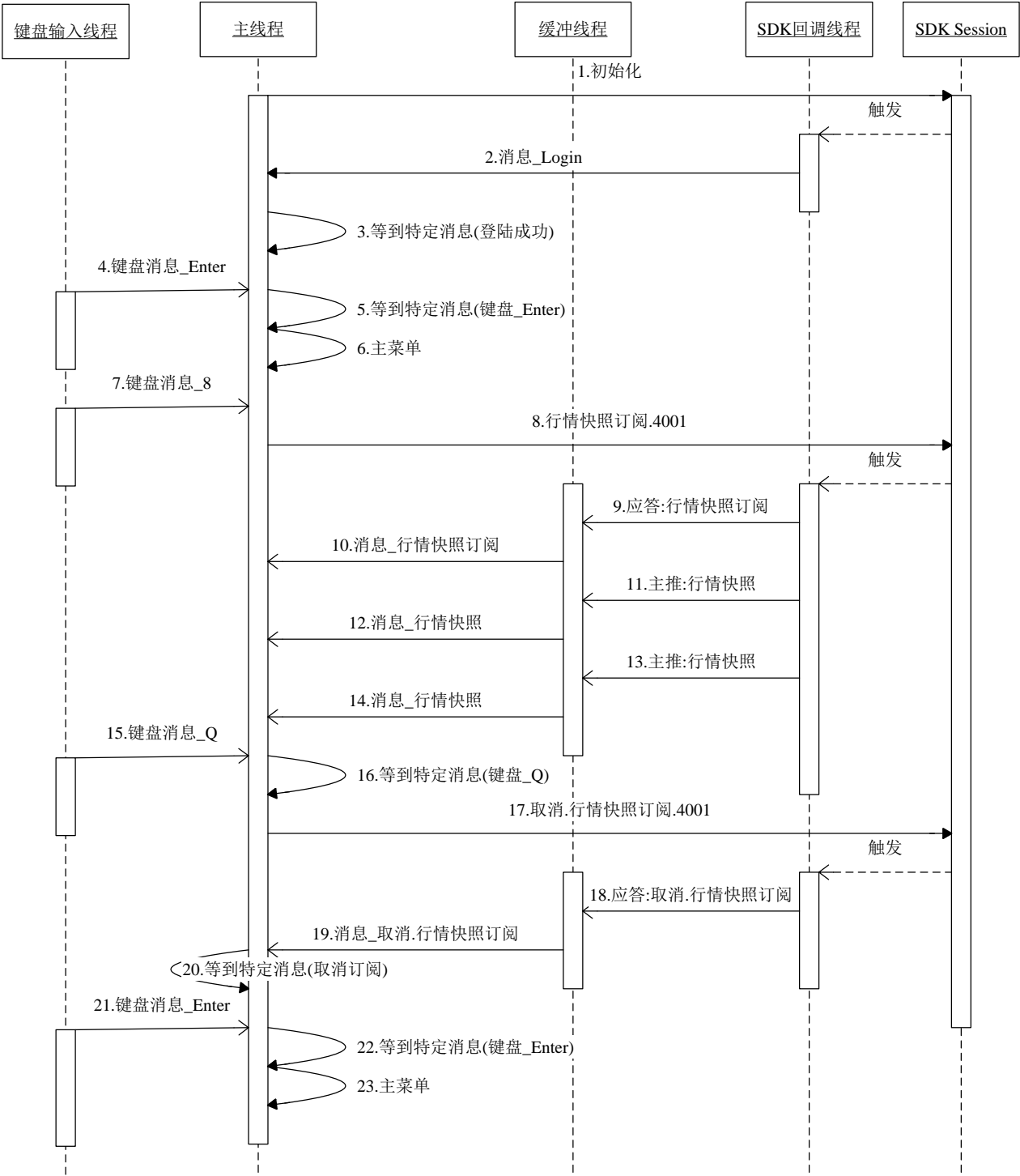
5.1.4. 缓冲线程

- 解析缓冲队列里的数据包
- 发 SDK 回调消息给主线程的消息队列。

编制部门	公共平台发展部
批准日期	2016/09/09

-
- 主函数:[sdk_data_parser.cpp]:[SdkBufferHandler::MainFun()]

5.2. 流程图示意



5.2.1. SDK 初始化: (调用栈如下)

- `main():myEnvironment.Init()` //主环境初始化
- `bool MyEnvironment::Init(void)` //线程初始化, SDK 初始化
- `StartAllSessions();` //开始启动session.

5.2.2. 消息_Login

- `H5SdkCallbackImpl::OnSdkLogin(Session *session)` //sdk login 回调
- `LoginAnsInfo *lp=session->LoginByUser(...)` //用帐号登陆
- `MyEnvironment::PostMainMessage(Command_SdkCallback, SdkCallback_Login, (WPARAM)true);` //回调线程, 发消息给主线程。

5.2.3. 等到特定消息(登陆成功)

- `main() :`
`MyEnvironment::GetHqSdkUtil()->SetLoginStatus((bool)pMessage->m_lParam);`
//主线程, 处理login消息。
- `myEnvironment.WaitforMessageAndHandleIt(...)` //主线程, 等候到了该消息,
- `bContinue = pMsgWaitFor->m_funcCallback()` //调用相应的回调函数。
- `PrintWaitForEnter()` //回调函数, 打印“回车 继续”

5.2.4. 键盘消息_Enter

- `KeyboardInputThread::MainFun()`
- `MyEnvironment::PostMainMessage(Command_KeyInput, ch);` //获取键盘输入, 发消息给主线程。

5.2.5. 等到特定消息(键盘_Enter)

- `myEnvironment.WaitforMessageAndHandleIt(...)` //主线程, 等候到了该消息,
- `bContinue = pMsgWaitFor->m_funcCallback()` //调用相应的回调函数。

5.2.6. 主菜单

- `MyEnvironment::PrintMenu(void)` //打印主菜单

5.2.7. 键盘消息_8

- `KeyboardInputThread::MainFun()`
- `MyEnvironment::PostMainMessage(Command_KeyInput, ch);` //获取键盘输入, 发消息给主线程。
- `main() : myEnvironment.SelectMenu(pMessage->m_wParam)` //选择菜单项
- `MyEnvironment::SelectMenu(...): case '8':` //菜单处理。

5.2.8. 行情快照订阅. 4001

- `GetHqSdkUtil()->SubscriberAll(...)` //菜单处理, 订阅。
- `HqSdkUtil::SubscriberAll(...)`
- `m_pSession->AsyncSend(request);` //异步命令

5.2.9. 应答:行情快照订阅

- `H5SdkCallbackImpl::OnReceived`
- `MyEnvironment::GetSdkBufferHandler()->Store2Buffer(...)` //缓冲

5.2.10. 消息_行情快照订阅

- `SdkBufferHandler::MainFun()`
- `item->Handle()`
- `void CSdkPackage::Handle(void)`
- `case H5SDK_MSG_SUBSCRIBE:`

- MyEnvironment::PostMainMessage(Command_SdkCallback, SdkCallback_Subscribe); //发消息给主线程

5.2.11. 主推:行情快照

- H5SdkCallbackImpl::OnReceived
- MyEnvironment::GetSdkBufferHandler()->Store2Buffer(...) //缓冲

5.2.12. 消息_行情快照

- SdkBufferHandler::MainFun()
- item->Handle()
- void CSdkPackage::Handle(void)
- case H5SDK_MSG_SNAPSHOT:
- HqSdkUtil::parse_snapshot(...)
- MyEnvironment::PostMainMessage(Command_SdkCallback, SdkCallback_Snapshot); //发消息给主线程

5.2.13. 主推:行情快照

5.2.14. 消息_行情快照

5.2.15. 键盘消息_Q

- KeyboardInputThread::MainFun()
- MyEnvironment::PostMainMessage(Command_KeyInput, ch); //获取键盘输入, 发消息给主线程。

5.2.16. 等到特定消息(键盘_Q)

- myEnvironment.WaitForMessageAndHandleIt(...) //主线程, 等候到了该消息,
- bContinue = pMsgWaitFor->m_funcCallback() //调用相应的回调函数。

5.2.17. 取消.行情快照订阅.4001

- bool MyEnvironment::OnSubscribeOver() //回调函数。
- m_pSession->AsyncSend(request); //异步命令

5.2.18. 应答:行情快照订阅

- H5SdkCallbackImpl::OnReceived
- MyEnvironment::GetSdkBufferHandler()->Store2Buffer(...) //缓冲

5.2.19. 消息_行情快照订阅

- SdkBufferHandler::MainFun()
- item->Handle()
- void CSdkPackage::Handle(void)
- case H5SDK_MSG_SUBSCRIBE:
- MyEnvironment::PostMainMessage(Command_SdkCallback, SdkCallback_Subscribe); //发消息给主线程

5.2.20. 等到特定消息(取消订阅)

- myEnvironment.WaitForMessageAndHandleIt(...) //主线程, 等候到了该消息,
- bContinue = pMsgWaitFor->m_funcCallback() //调用相应的回调函数。
- MyEnvironment::PrintWaitForEnter() //打印“回车 继续”

5.2.21. 键盘消息_Enter

- KeyboardInputThread::MainFun()
- MyEnvironment::PostMainMessage(Command_KeyInput, ch); //获取键盘输入, 发消息

编制部门	公共平台发展部
批准日期	2016/09/09

给主线程。

5.2.22. 等到特定消息(键盘_Enter)

- `myEnvironment.WaitForMessageAndHandleIt(...)` //主线程，等候到了该消息，
- `bContinue = pMsgWaitFor->m_funcCallback()` //调用相应的回调函数。

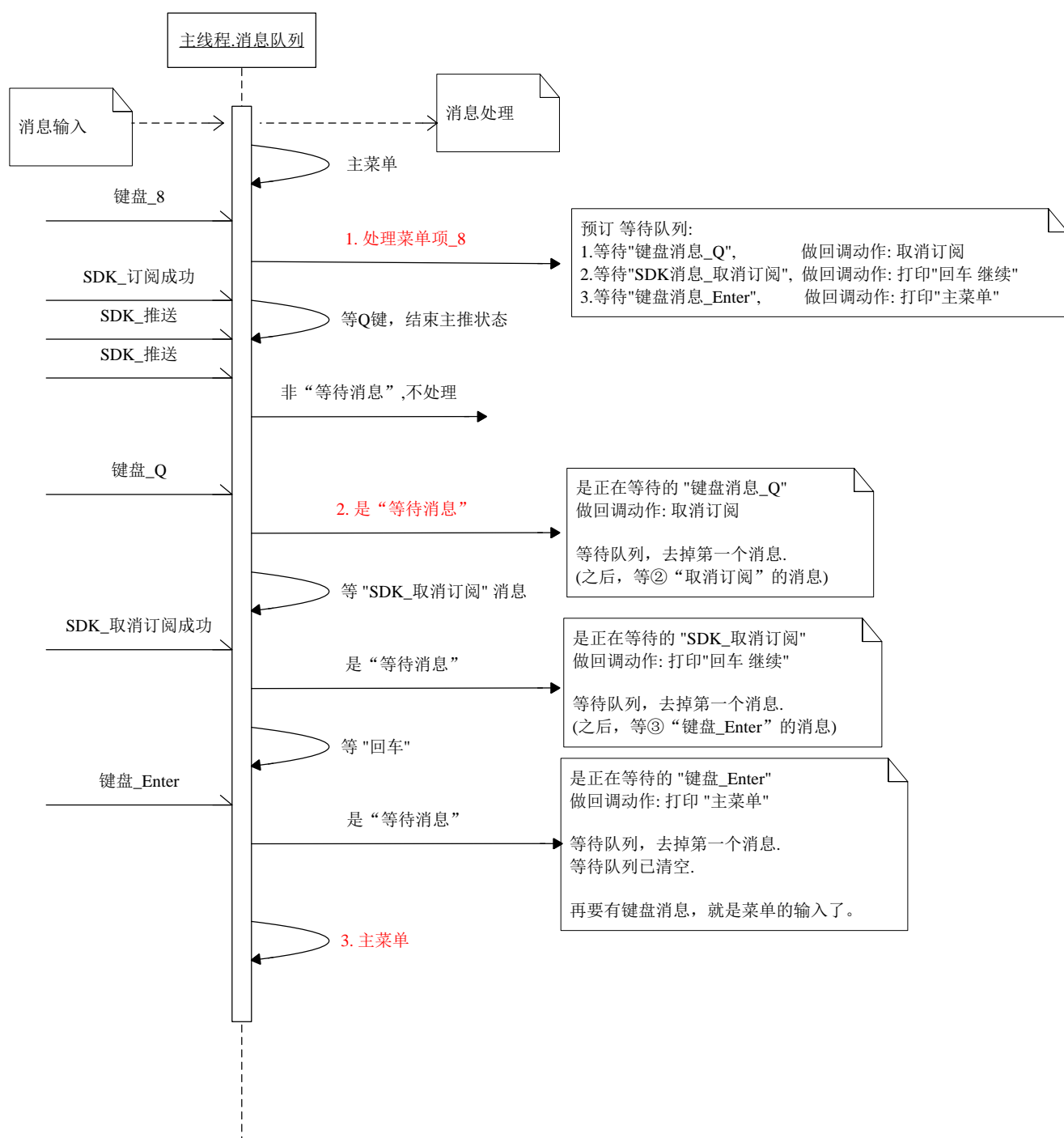
5.2.23. 主菜单

- `MyEnvironment::PrintMenu(void)` //打印主菜单

.....

继续循环

5.3.消息处理示意



5.3.1. 处理菜单项_8

- 枚举某市场的所有代码，放入队列：
GetHqSdkUtil()->GetCodesInMarket("XSHE.ESA.SMSE", &m_listCodes,..)
- 往预定义队列放入特定行为,包括: 消息值，回调方法。
 - ◆ WaitForMessage(Command_KeyInput, Key_Quit, OnSubscribeOver);//等待'Q'.
退出订阅模式
 - ◆ WaitForMessage(Command_SdkCallback,SdkCallback_Subscribe,
PrintWaitForEnter);//等待退订动作结束。
 - ◆ WaitForMessage(Command_KeyInput, Key_Return, PrintMenu);//等待回车.打
印菜单
- 开始订阅:
GetHqSdkUtil()->SubscriberAll(&m_listCodes,...)

5.3.2. 是“等待消息”

```
while(pMessage = MyEnvironment::PeekMainMessage())//取“当前消息”
{
    ...

    if(myEnvironment.GetWaitforMessageSize() > 0)//有“正在等待的消息”
    {
        if(! myEnvironment.WaitforMessageAndHandleIt(pMessage)) // “正在等待的消息”与“当前消息”一致，处理之
        ...
    }
    ...

} //继续消息队列
```

5.3.3. 主菜单

```
while(pMessage = MyEnvironment::PeekMainMessage())//取“当前消息”
{
    ...
    if(myEnvironment.GetWaitforMessageSize() > 0)
    {
    }
    else//无“正在等待的消息”
    {
        //没等待的消息，如果是键盘输入，就是菜单选项。
        if(pMessage->m_commandType == Command_KeyInput)
        {
            if(! myEnvironment.SelectMenu(pMessage->m_wParam))
            {
                delete pMessage;
                break;//'q',return false,退出程序
            }
        }
    }
}
```

```
    }  
}  
  
    delete pMessage;  
} //继续消息队列
```

6. 内存泄漏检测

6.1. Windows

推荐使用 vld 检测内存泄漏：

[c++\demo\src\src\h5sdk_c++_demo\my_vld.h] 里，开关 `#include <vld.h>`，切换该功能。

6.2. Linux

推荐使用 valgrind 检测内存泄漏：

- `<cd c++/demo/src/bin>`
- `export LD_LIBRARY_PATH=`pwd`./`
- `valgrind --leak-check=full ./h5-sdk-demo`

7. 注意事项

7.1. 注意数值的无符号/有符号。

- 把 int 用 %u 打印，-3（超时错误）被打印成 4294967293，无法理解
- GetInt64(),可能返回的 int64 或 uint64.

要根据文档确认返回的值，是 int64 或 uint64.下面有两个例子:

- ◆ double px_change = (double)(int64)record->GetItem(H5SDK_TAG_PX_CHANGE)->GetInt64();
H5SDK_TAG_PX_CHANGE 返回的是 int64. 如果直接转换成 double,会是一个错误值。
- ◆ uint64 last_px = record->GetItem(H5SDK_TAG_LAST_PX)->GetInt64();
H5SDK_TAG_LAST_PX 返回的是 uint64.

- 7.2. SDK 连接断开，SDK 应该在 OnCore, OnError, OnClose,这几个地方，调用 session->AsyncConnect（）去自动重连。不建议在此之外，再额外的去重连,强调不宜由用户的线程来处理。
- 7.3. 订阅，同步或异步订阅，返回值，表明这些股票代码订阅成功。里面的具体内容，是缺省值，不是真实值。内容应该忽略。另，断开重连接后，SDK 会自动重新订阅，所以，OnReceive 会收到异步的订阅消息。强调:重连成功后，用户不要再去订阅。之前的订阅，SDK 会自动重订阅。
- 7.4. 订阅的过程
- 如果订阅的产品代码有重复（比如 4001 命令，成功后，再重复一次订阅），第一次主推，会出现 group == null 的现象。
 - 如果故意扒网线，过几秒，会打印 OnClose.然后在 onclose 函数里调用 session->AsyncConnect();开始重新连接，提示 OnCore,连接超时。然后插上网线，提示连接成功，login 成功，之后，继续主推。
 - 1004 请求服务器信息. 在开市前，有自动推送表明该市场有初始化动作，需要重新订阅该市场的证券代码。
- 7.5. 市场初始化时，取行情快照，可能数据会错误。
- 7.6. 对 onReceive 的响应，不能太慢，否则可能导致 SDK 接收的数据太多，累计，导致断线。onClose 出现
- 7.7. 行情编程，要特别注意内存增长情况，太快，说明有 release 没做，或者其它内存泄漏。注意，即使没有 release 消息，因为退出时 StopAllSessions 会清理 sdk 的内存，所以，最后还是没有泄露。但是，运行过程中，内存会不停的上涨。
- 7.8. 行情的推送处理，安全稳定的条件:
- 内存不要泄漏。
 - 处理速度要尽量快。
 - 在 SDK 的回调函数里，一定不要再往 SDK 里发送同步命令(建议，异步命令最好也别用)。如果要发，弄个消息队列，或其他啥的，去其他线程做
- 7.9. 添加消息内容时，如果 addRecord 返回 null，可能是内存不够用了
- 7.10. SyncCall 命令，不要同时操作。如果有可能多线程同时执行，需要加上锁。
- 7.11. logout 后，session 调用命令，还是能成功。但是内部数据为空。