深圳证券信息有限公司		文档编号				
		名　　称	深交所行情互联网接入服务-上交所 LDDS 行情接口说明书			
编写	签名：　日期：		**密级**	内部公开	版本	V1.0.0
审核	签名：　日期：		**批准**			

修订记录：

版本编号	编写/修订内容	修订人	修订日期

深圳证券信息有限公司

目 录

1. 引言

深圳证券信息有限公司负责深圳证券交易所的基础行情、增强行情的牌照授权工作，目前所有从本公司获得合法行情牌照授权的客户，仅能通过深证通、上证信息、其他信息商等途径获取深交所行情数据，存在着接入成本高、服务不稳定、响应不及时、行情数据延时等种种问题。为保障已授权客户的权益，本公司建立深交所基础行情系统，为所有已授权基础行情客户提供深交所、上交所基础行情数据接入服务。

深交所行情互联网接入服务，旨在为各类已授权基础行情客户提供深交所、上交所的基础行情数据接入服务，系统建设目标是通过多种服务方式覆盖各类客户群体的行情数据接入需求，提供低成本、低门槛、稳定、高效的行情接入方案。

2. 安装及应用发布

2.1. C 语言版本 LDDSApi

C 语言版本的 LDDSApi 支持的操作系统列表如下：

操作系统	说明
Windows 7	Windows7 或以上，仅支持 64 位操作系统
Linux	CentOS 6.7、CentOS 7.2、RedHat 6.7、RedHat 7.2

C 语言版本的 LDDSApi 由以下几个文件组成：

文件名	说明
ldds_api.h	LDDSApi 头文件
ldds_api.lib	LDDSApi 库文件
ldds_api.dll	LDDSApi 动态链接库文件，Windows 平台
libldds_api.so libboost_atomic.so.1.62.0 libboost_chrono.so.1.62.0 libboost_date_time.so.1.62.0 libboost_filesystem.so.1.62.0 libboost_locale.so.1.62.0 libboost_log_setup.so.1.62.0 libboost_log.so.1.62.0	LDDSApi 动态链接库文件，Linux 平台

libboost_program_options.so.1.62.0 libboost_regex.so.1.62.0 libboost_system.so.1.62.0 libboost_thread.so.1.62.0 libmsg_parser.so	

2.2. Java 语言版本 LDDSApi

Java 语言版本的 LDDSApi 支持的操作系统列表如下：

操作系统	说明
Windows	Windows7 或以上，仅支持 64 位操作系统
Linux	CentOS 6.7、CentOS 7.2、RedHat 6.7、RedHat 7.2

Java 语言版本的 LDDSApi 由以下几个文件组成：

文件名	说明
ldds_api.jar	LDDSApi jar 包
ldds_api.dll	LDDSApi 动态链接库文件，Windows 平台
libldds_api.so libboost_atomic.so.1.62.0 libboost_chrono.so.1.62.0 libboost_date_time.so.1.62.0 libboost_filesystem.so.1.62.0 libboost_locale.so.1.62.0 libboost_log_setup.so.1.62.0 libboost_log.so.1.62.0 libboost_program_options.so.1.62.0 libboost_regex.so.1.62.0 libboost_system.so.1.62.0 libboost_thread.so.1.62.0 libmsg_parser.so	LDDSApi 动态链接库文件，Linux 平台

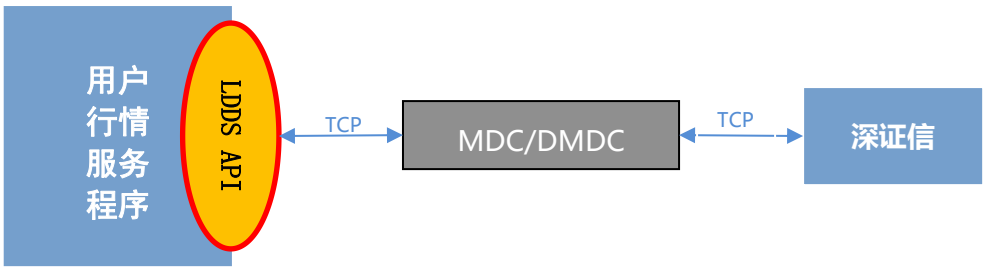
3. 使用概述

3.1. 功能

LDDSApi 是一组提供给用户调用的 C/Java 语言应用程序接口，用户可以调用该 API 开发，实现上交所 LDDS 行情的接收，LDDSApi 可以完成通信的自动连接，接收数据包等

功能，应用程序只需要设置好对应的回调函数就可以接收上交所二进制行情数据，API 处于最底层，由应用程序调用，通过 TCP 连接与上层节点通信。将接收到的数据包处理后，通过回调接口通知给调用的应用程序。

3.2. 应用环境



LDDS API 应用环境

在深交所行情互联网接入服务中，MDC/DMDC 负责接收上游行情服务节点推送的行情数据，对下游提供上交所 LDDS 标准协议接口，LDDS API 通过直接连接 MDC/DMDC 获取行情数据。

3.3. 线程安全性

参看下面 4.3 节。

3.4. 应用系统的安全性

用户在应用软件中调用了 LDDSApi， 在 LDDSApi 中不会包含故意攻击用户系统的软件或代码，不会故意影响用户系统的安全运行。用户系统自身的安全性，应该由用户自己进行检查和维护，LDDSApi 无法保证用户系统本身的安全性。

4. 编程参考

4.1. 常量定义

4.1.1. 长度常量

名称	定义值	说明
API_MAXLEN_ERRORSTR	256	错误码字符串的最大长度

API_MAXLEN_LOGSTR	512	日志字符串的最大长度

4.1.2. 函数返回值

返回值	含义
0	成功
-1	无效指针
-2	无效参数

4.2. 数据结构说明

4.2.1. ConnInfo

该结构用来定义 LDDS 连接 TCP 参数

结构定义：

```
struct ConnInfo
{
    char        ip[32];
    uint16_t port;
    char        sender_comp_id[32];
    char        target_comp_id[32];
    char        password[64];
};
```

字段说明：

字段	说明
ip	服务地址
port	服务端口
sender_comp_id	SenderCompID
target_comp_id[	TargetCompID
password	Password

4.2.2. LDDSConnInfo

该结构用来定义 LDDSApi 连接结构定义：

```
struct LDDSConnInfo
{
    ConnInfo realtime_conn_info;
    ConnInfo retran_conn_info;
    uint16_t heartbeat_interval;
    uint32_t retran_times;
```

```
    uint32_t retran_interval;
    bool        multi_callback;
};
```

字段说明：

字段	说明
realtime_conn_info	实时连接信息
retran_conn_info	重传连接信息
heartbeat_interval	心跳间隔
retran_times	重传次数
retran_interval	重传间隔
multi_callback	多条记录回调标志 true：每次回调可能会有多条记录， false：每次回调一条记录

4.2.3. LDDSCashAuctionMarketDataSnapshotLevel1

该结构用来定义 LDDS 系统 Level1 现货快照行情数据

结构定义：

```
struct LDDSCashAuctionMarketDataSnapshotLevel1
{
    char        md_stream_id[8];
    char        security_id[8];
    char        symbol[32];
    uint64_t num_trades;
    uint64_t trade_volume;
    double     trade_value_traded;
    double     prev_close_px;
    double     prev_set_px;
    uint64_t total_long_position;
    double     match_px;
    double     index;
    double     open_price;
    double     close_price;
    double     settle_price;
    double     high_px;
    double     low_px;
    double     iopv;
    double     nav;
    double     dynamic_price;
    double     bid_price[5];
    uint64_t bid_volume[5];
```

```
    double     offer_price[5];
    uint64_t offer_volume[5];
    char       trading_phase_code[10];
};
```

字段说明：

字段	说明
md_stream_id	行情记录标签
security_id	产品代码
symbol	产品简称
num_trades	成交笔数
trade_volume	成交量，单位：张或股
trade_value_traded	成交金额，单位：元
prev_close_px	昨收盘价
prev_set_px	昨结算价
total_long_position	未平仓合约数量
match_px	成交价
index	指数
open_price	今开盘价
close_price	今收盘价
settle_price	今结算价
high_px	当日最高成交价
low_px	当日最低成交价
iopv	基金 IOPV
nav	最新净值
dynamic_price	动态参考价格
bid_price	申买价
bid_volume	申买量
offer_price	申卖价
offer_volume	申卖量
trading_phase_code	当前产品状态

字段具体含义和格式定义等明细参见《上海证券交易所 LDDS 系统 Level-1FAST 行情接口说明书》。

4.2.4. LDDSCashAuctionMarketStatusLevel1

该结构用来定义市场状态消息数据

结构定义：

```
struct LDDSCashAuctionMarketStatusLevel1
```

```
{
    char        security_type[3];
    int16_t     trade_ses_mode;
    char        trading_session_id[10];
    int32_t     tot_no_related_sym;
};
```

字段说明：

字段	说明
security_type	证券类型
trade_ses_mode;	交易盘交易模式
trading_session_id	全市场行情状态
tot_no_related_sym	最大产品数目（包括指数）

字段具体含义和格式定义等明细参见《上海证券交易所 LDDS 系统 Level-1FAST 行情接口说明书》。

4.2.5. LDDSCashAuctionMarketDataSnapshotLevel2

该结构用来定义 LDDS 系统 Level2 现货快照行情数据

结构定义：

```
struct LDDSCashAuctionMarketDataSnapshotLevel2
{
    int32_t     category_id;
    int32_t     msg_seq_id;
    int32_t     date_timestamp;
    char        security_id[8];
    int32_t     pre_close_px;
    int32_t     open_px;
    int32_t     high_px;
    int32_t     low_px;
    int32_t     last_px;
    int32_t     close_px;
    char        instrument_status[32];
    char        trading_phase_code[10];
    int32_t     num_trades;
    int64_t     total_volume_trade;
    int64_t     total_value_trade;
    int64_t     total_bid_qty;
    int32_t     weighted_avg_bid_px;
```

```
    int32_t   alt_weighted_avg_bid_px;
    int64_t   total_offer_qyt;
    int32_t   weighted_avg_offer_px;
    int32_t   alt_weighted_avg_offer_px;
    int32_t   iopv;
    int32_t   etf_buy_number;
    int64_t   etf_buy_amount;
    int64_t   etf_buy_money;
    int32_t   etf_sell_number;
    int64_t   etf_sell_amount;
    int64_t   etf_sell_money;
    int32_t   yield_to_maturity;
    int64_t   total_warrant_exec_qty;
    int64_t   war_lower_px;
    int64_t   war_upper_px;
    int32_t   withdraw_buy_number;
    int64_t   withdraw_buy_amount;
    int64_t   withdraw_buy_money;
    int32_t   withdraw_sell_number;
    int64_t   withdraw_sell_amount;
    int64_t   withdraw_sell_money;
    int32_t   total_bid_number;
    int32_t   total_offer_number;
    int32_t   bid_trade_max_duration;
    int32_t   offer_trade_max_duration;
    int32_t   num_bid_orders;
    int32_t   num_offer_orders;
    int32_t   bid_px[10];
    int64_t   bid_volume[10];
    int32_t   offer_px[10];
    int64_t   offer_volume[10];
    int64_t   bid_one_volume[50];
    int64_t   offer_one_volume[50];
};
```

字段说明：

字段	说明
category_id	产品类别
msg_seq_id	消息序号
date_timestamp	最新订单时间（秒）
security_id	证券代码
pre_close_px	昨收盘价

open_px	开盘价
high_px	最高价
low_px	最低价
last_px	现价
close_px	今日收盘价格
instrument_status	当前品种交易状态
trading_phase_code	当前产品状态
num_trades	成交笔数
total_volume_trade	成交总量
total_value_trade	成交总金额（元）
total_bid_qty	委托买入总量
weighted_avg_bid_px	加权平均委买价格（元）
alt_weighted_avg_bid_px	债券加权平均委买价格（元）
total_offer_qyt	委托卖出总量
weighted_avg_offer_px	加权平均委卖价格（元）
alt_weighted_avg_offer_px	债券加权平均委买价格（元）
iopv	ETF 净值估值
etf_buy_number	ETF 申购笔数
etf_buy_amount	ETF 申购数量
etf_buy_money	ETF 申购金额
etf_sell_number	ETF 赎回笔数
etf_sell_amount	ETF 赎回数量
etf_sell_money	ETF 申购金额
yield_to_maturity	债券到期收益率
total_warrant_exec_qty	权证执行的总数量
war_lower_px	债券质押式回购品种加权平均价(加权平均回购利率)该字段只对债券质押式协议回购有效
war_upper_px	权证涨停价格（元）
withdraw_buy_number	买入撤单笔数
withdraw_buy_amount	买入撤单数量
withdraw_buy_money	买入撤单金额
withdraw_sell_number	卖出撤单笔数
withdraw_sell_amount	卖出撤单数量
withdraw_sell_money	卖出撤单金额
total_bid_number	买入总笔数
total_offer_number	卖出总笔数
bid_trade_max_duration	买入委托成交最大等待时间
offer_trade_max_duration	卖出委托成交最大等待时间
num_bid_orders	买方委托价位数
num_offer_orders	卖方委托价位数
bid_px	申买价

bid_volume	申买量
offer_px	申卖价
offer_volume	申卖量
bid_one_volume	买一前 50 笔委托数量
offer_one_volume	卖一前 50 笔委托数量

字段具体含义和格式定义等明细参见《上海证券交易所 LDDS 系统 Level-2 行情接口说明书》。

4.2.6. LDDSCashAuctionVirtualAuctionPriceSnapshot

该结构用来定义 Level2 虚拟集合竞价快照数据

结构定义：

```
struct LDDSCashAuctionVirtualAuctionPriceSnapshot
{
    int32_t   category_id;
    int32_t   msg_seq_id;
    int32_t   date_time_stamp;
    char      security_id[8];
    int32_t   price;
    int64_t   virtual_auction_qty;
    int64_t   leave_qty;
    char      side;
};
```

字段说明：

字段	说明
category_id	产品类别
msg_seq_id	消息序号
date_time_stamp	行情时间（秒）
security_id[8]	证券代码
price	虚拟参考价格
virtual_auction_qty	虚拟匹配量
leave_qty	虚拟未匹配量
side	0:无未匹配量，买卖两边的未匹配量都为 0; 1:买方有未匹配量，卖方未匹配量=0;2:卖方有未匹配量，买方未匹配量=0

字段具体含义和格式定义等明细参见《上海证券交易所 LDDS 系统 Level-2 行情接口说明书》。

4.2.7. LDDSIndexMarketDataSnapshot

该结构用来定义 Level2 指数行情快照数据

结构定义：

```
struct LDDSIndexMarketDataSnapshot
{
    int32_t   category_id;
    int32_t   msg_seq_id;
    int32_t   date_time_stamp;
    char       security_id[8];
    int64_t   pre_close_index;
    int64_t   open_index;
    int64_t   turnover;
    int64_t   high_index;
    int64_t   low_index;
    int64_t   last_index;
    int32_t   trade_time;
    int64_t   total_volume_traded;
    int64_t   close_index;
};
```

字段说明：

字段	说明
category_id	产品类别
msg_seq_id	消息序号
date_time_stamp	行情时间（秒）
security_id[8]	证券代码
pre_close_index	前收盘指数
open_index	今开盘指数
turnover	参与计算相应指数的成交金额（元）
high_index	最高指数
low_index	最低指数
last_index	最新指数
trade_time	成交时间
total_volume_traded	参与计算相应指数的交易数量（手）
close_index	今日收盘指数（大于 0 为有效值）

字段具体含义和格式定义等明细参见《上海证券交易所 LDDS 系统 Level-2 行情接口说明书》。：

4.2.8. LDDSMarketOverviewSnapshot

该结构用来定义 Level2 市场总览快照数据

结构定义：

```
struct LDDSMarketOverviewSnapshot
{
    int32_t   category_id;
    int32_t   msg_seq_id;
    int32_t   date_time_stamp;
    char       security_id[8];
    int32_t   orig_time;
    int32_t   orig_date;
};
```

字段说明：

字段	说明
category_id	产品类别
msg_seq_id	消息序号
date_time_stamp	行情时间（秒）
security_id[8]	证券代码
orig_time	市场时间（百分之一秒）
orig_date	市场日期

字段具体含义和格式定义等明细参见《上海证券交易所 LDDS 系统 Level-2 行情接口说明书》。

4.2.9. LDDSCashAuctionMarketDataTickExecution

该结构用来定义 Level2 逐笔成交数据

结构定义：

```
struct LDDSCashAuctionMarketDataTickExecution
{
    int32_t   category_id;
    int32_t   msg_seq_id;
    int32_t   trade_index;
    int32_t   trade_channel;
    char       security_id[8];
    int32_t   trade_time;
    int32_t   trade_price;
```

```
    int64_t   trade_qty;
    int64_t   trade_money;
    int64_t   trade_buy_no;
    int64_t   trade_sell_no;
    char       trade_bs_flag;
};
```

字段说明：

字段	说明
category_id	产品类别
msg_seq_id	消息序号
trade_index	成交序号从 1 开始，按 TradeChannel 连续
trade_channel	成交通道
security_id[8]	证券代码
trade_time	成交时间（百分之一秒）
trade_price	成交价格(元)
trade_qty	成交数量
trade_money	成交金额(元)
trade_buy_no	买方订单号
trade_sell_no	卖方订单号
trade_bs_flag	内外盘标志

字段具体含义和格式定义等明细参见《上海证券交易所 LDDS 系统 Level-2 行情接口说明书》。

4.2.10.LDDSCallParam

该结构用来定义 LDDSApi 的回调函数设置等

结构定义：

```
struct LDDSCallParam
{
    void* user_ptr = nullptr;
    void(*OnLog)(void* user_ptr, const LOGLevel& level, const char
log_msg[API_MAXLEN_LOGSTR]) = nullptr;
    void(*OnCashAuctionMarketDataSnapshotLevel1)(void* user_ptr, const
LDDSCashAuctionMarketDataSnapshotLevel1* marketdata_snapshot_ptr, const uint32_t& count)
= nullptr;
    void(*OnCashAuctionMarketStatusLevel1)(void* user_ptr, const
LDDSCashAuctionMarketStatusLevel1* market_status_ptr, const uint32_t& count) = nullptr;
    void(*OnCashAuctionMarketDataSnapshotLevel2)(void* user_ptr, const
```

```
LDDSCashAuctionMarketDataSnapshotLevel2* marketdata_snapshot_ptr, const uint32_t& count)
= nullptr;
    void(*OnCashAuctionVirtualAuctionPriceSnapshot)(void* user_ptr, const
LDDSCashAuctionVirtualAuctionPriceSnapshot* virtual_auction_price_snapshot_ptr, const
uint32_t& count) = nullptr;
    void(*OnIndexMarketDataSnapshot)(void* user_ptr, const LDDSIndexMarketDataSnapshot*
index_marketdata_snapshot_ptr, const uint32_t& count) = nullptr;
    void(*OnMarketOverviewSnapshot)(void* user_ptr, const LDDSMarketOverviewSnapshot*
market_overview_snapshot_ptr, const uint32_t& count) = nullptr;
    void(*OnCashAuctionMarketDataTickExecution)(void* user_ptr, const
LDDSCashAuctionMarketDataTickExecution* maket_data_tick_execution_ptr, const uint32_t&
count) = nullptr;
};
```

字段说明：

字段	说明	线程安全性
user_ptr	开发人员根据需要绑定的数据指针	无
OnLog	LDDSApi 日志回调函数	不保证在同一线程
OnCashAuctionMarketDataSnapshotLevel1	Level1 现货快照数据回调函数	不保证在同一线程
OnCashAuctionMarketStatusLevel1	Level1 市场状态消息数据回调函数	不保证在同一线程
OnCashAuctionMarketDataSnapshotLevel2	Level2 市场行情快照数据回调函数	不保证在同一线程
OnCashAuctionVirtualAuctionPriceSnapshot	Level2 虚拟集合竞价快照数据回调函数	不保证在同一线程
OnIndexMarketDataSnapshot	Level2 指数行情快照数据回调函数	不保证在同一线程
OnMarketOverviewSnapshot	Level2 市场总览快照数据	不保证在同一线程
OnCashAuctionMarketDataTickExecution	Level2 逐笔成交数据	不保证在同一线程

字段具体含义和格式定义等明细参见《上海证券交易所 LDDS 系统 Level-2 行情接口说明书》。

4.3. C 函数接口

4.3.1. 函数清单

LDDS API 是一个简明易用的编程接口，它提供了如下 4 个函数：

序号	函数名称	函数功能	线程安全性
1	LDDSCreate	初始化相关资源	是
2	LDDSStart	启动 API 的运行	是
3	LDDSStop	停止 API 的运行	否
4	LDDSDestroy	释放相关资源	否

4.3.2. LDDSCreate

LDDSApi 的初始化函数，该函数对 API 进行初始化，分配获取相关的资源。

函数原型：

```
LDDS_API void* LDDSCreate(const LDDSConnInfo& conn_info, const LDDSCallParam&
call_param, int32_t& err_code, char err_msg[API_MAXLEN_ERRORSTR]);
```

参数说明 ：

参数	说明
conn_info[in]	配置连接信息
call_param [in]	API 的回调函数
err_code[out]	调用失败时的错误码
err_msg[out]	调用失败时的错误描述

返回值说明：

返回值	说明
NULL	初始化失败
非 NULL	初始化成功，返回一个连接句柄

4.3.3. LDDSStart

运行 API，建立连接，启用数据推送。

函数原型：

```
LDDS_API int     LDDSStart(void* handle);
```

参数说明 ：

参数	说明
handle[in]	初始化返回的句柄

返回值说明：

返回值	说明
0	成功返回
非 0	运行失败

4.3.4. LDDSStop

停止 API 的运行，断开与上游直接的连接，停止数据推送。

函数原型：

```
LDDS_API int    LDDSStop(void* handle);
```

参数说明 ：

参数	说明
handle[in]	初始化返回的句柄

返回值说明：

返回值	说明
0	成功返回
非 0	运行失败

4.3.5. LDDSDestroy

销毁 api 句柄

函数原型：

```
LDDS_API void   LDDSDestroy(void* handle);
```

参数说明 ：

参数	说明
handle[in]	初始化返回的句柄

返回值说明：

返回值	说明
无	无

4.3.6. 调用顺序

按顺序依次调用 LDDSCreate，LDDSStart。若要停止 LDDSApi，则按顺序调用 LDDSStop，LDDSDestroy。

5. 日志输出

LDDSApi 的日志输出是以回调函数的方式返回给用户的，回调函数请参考 4.2.10 章节。

常见的日志有程序的启动信息，网络建立和断开信息、登录成功或失败信息等。

6. C++编程示例

```
#include <string>
#include <iostream>
#include <lddsapi/ldds_api.h>

void OnLog(void* user_ptr, const LOGLevel& level, const char* log)
{
    std::cout << "LEVEL: " << level << "||" << log << std::endl;
}

void OnCashAuctionMarketDataSnapshotLevel1(void* user_ptr, const
LDDSCashAuctionMarketDataSnapshotLevel1* marketdata_snapshot_ptr, const uint32_t& count)
{
    std::cout << "OnCashAuctionMarketDataSnapshotLevel1, count: " << count << std::endl;
    std::cout << "md_stream_id" << marketdata_snapshot_ptr[count - 1].md_stream_id <<
std::endl;
    std::cout << "security_id" << marketdata_snapshot_ptr[count - 1].security_id << std::endl;
    std::cout << "symbol" << marketdata_snapshot_ptr[count - 1].symbol << std::endl;
    std::cout << "num_trades" << marketdata_snapshot_ptr[count - 1].num_trades << std::endl;
    std::cout << "trade_volume" << marketdata_snapshot_ptr[count - 1].trade_volume <<
std::endl;
    std::cout << "trade_value_traded" << marketdata_snapshot_ptr[count -
1].trade_value_traded << std::endl;
    std::cout << "prev_close_px" << marketdata_snapshot_ptr[count - 1].prev_close_px <<
std::endl;
    std::cout << "prev_set_px" << marketdata_snapshot_ptr[count - 1].prev_set_px << std::endl;
    std::cout << "total_long_position" << marketdata_snapshot_ptr[count -
1].total_long_position << std::endl;
    std::cout << "match_px" << marketdata_snapshot_ptr[count - 1].match_px << std::endl;
    std::cout << "index" << marketdata_snapshot_ptr[count - 1].index << std::endl;
    std::cout << "open_price" << marketdata_snapshot_ptr[count - 1].open_price << std::endl;
    std::cout << "close_price" << marketdata_snapshot_ptr[count - 1].close_price << std::endl;
    std::cout << "settle_price" << marketdata_snapshot_ptr[count - 1].settle_price << std::endl;
    std::cout << "high_px" << marketdata_snapshot_ptr[count - 1].high_px << std::endl;
    std::cout << "low_px" << marketdata_snapshot_ptr[count - 1].low_px << std::endl;
    std::cout << "iopv" << marketdata_snapshot_ptr[count - 1].iopv << std::endl;
    std::cout << "nav" << marketdata_snapshot_ptr[count - 1].nav << std::endl;
    std::cout << "dynamic_price" << marketdata_snapshot_ptr[count - 1].dynamic_price <<
std::endl;
```

```
    std::cout << "bid_price[4]" << marketdata_snapshot_ptr[count - 1].bid_price[4] << std::endl;
    std::cout << "bid_volume[4]" << marketdata_snapshot_ptr[count - 1].bid_volume[4] <<
std::endl;
    std::cout << "offer_price[4]" << marketdata_snapshot_ptr[count - 1].offer_price[4] <<
std::endl;
    std::cout << "offer_volume[4]" << marketdata_snapshot_ptr[count - 1].offer_volume[4] <<
std::endl;
    std::cout << "trading_phase_code" << marketdata_snapshot_ptr[count -
1].trading_phase_code << std::endl;
}
void OnCashAuctionMarketStatusLevel1(void* user_ptr, const
LDDSCashAuctionMarketStatusLevel1* market_status_ptr, const uint32_t& count)
{
    std::cout << "OnCashAuctionMarketStatusLevel1, count: " << count << std::endl;
    std::cout<<"security_type"<<market_status_ptr[count-1].security_type<<std::endl;
    std::cout<<"trade_ses_mode"<<market_status_ptr[count-1].trade_ses_mode<<std::endl;

std::cout<<"trading_session_id"<<market_status_ptr[count-1].trading_session_id<<std::endl;

std::cout<<"tot_no_related_sym"<<market_status_ptr[count-1].tot_no_related_sym<<std::endl;
}
void OnCashAuctionMarketDataSnapshotLevel2(void* user_ptr, const
LDDSCashAuctionMarketDataSnapshotLevel2* marketdata_snapshot_ptr, const uint32_t& count)
{
    std::cout<<"OnCashAuctionMarketDataSnapshotLevel2,count:"<<count<<std::endl;
    std::cout<<"category_id"<<marketdata_snapshot_ptr[count-1].category_id<<std::endl;
    std::cout<<"msg_seq_id"<<marketdata_snapshot_ptr[count-1].msg_seq_id<<std::endl;
    std::cout<<"date_timestamp"<<marketdata_snapshot_ptr[count-1].date_timestamp
            <<std::endl;
    std::cout<<"security_id[8]"<<marketdata_snapshot_ptr[count-1].security_id<<std::endl;
    std::cout<<"pre_close_px"<<marketdata_snapshot_ptr[count-1].pre_close_px<<std::endl;
    std::cout<<"open_px"<<marketdata_snapshot_ptr[count-1].open_px<<std::endl;
    std::cout<<"high_px"<<marketdata_snapshot_ptr[count-1].high_px<<std::endl;
    std::cout<<"low_px"<<marketdata_snapshot_ptr[count-1].low_px<<std::endl;
    std::cout<<"last_px"<<marketdata_snapshot_ptr[count-1].last_px<<std::endl;
    std::cout<<"close_px"<<marketdata_snapshot_ptr[count-1].close_px<<std::endl;
    std::cout<<"instrument_status[32]"
            <<marketdata_snapshot_ptr[count-1].instrument_status[32]<<std::endl;
    std::cout<<"trading_phase_code[10]"
            <<marketdata_snapshot_ptr[count-1].trading_phase_code[10]<<std::endl;
    std::cout<<"num_trades"<<marketdata_snapshot_ptr[count-1].num_trades<<std::endl;
    std::cout<<"total_volume_trade"
```

```
        <<marketdata_snapshot_ptr[count-1].total_volume_trade<<std::endl;
std::cout<<"total_value_trade"
<<marketdata_snapshot_ptr[count-1].total_value_trade<<std::endl;
std::cout<<"total_bid_qty"<<marketdata_snapshot_ptr[count-1].total_bid_qty<<std::endl;
std::cout<<"weighted_avg_bid_px"
        <<marketdata_snapshot_ptr[count-1].weighted_avg_bid_px<<std::endl;
std::cout<<"alt_weighted_avg_bid_px"
        <<marketdata_snapshot_ptr[count-1].alt_weighted_avg_bid_px<<std::endl;
std::cout<<"total_offer_qyt"
        <<marketdata_snapshot_ptr[count-1].total_offer_qyt<<std::endl;
std::cout<<"weighted_avg_offer_px"
        <<marketdata_snapshot_ptr[count-1].weighted_avg_offer_px<<std::endl;
std::cout<<"alt_weighted_avg_offer_px"
        <<marketdata_snapshot_ptr[count-1].alt_weighted_avg_offer_px<<std::endl;
std::cout<<"iopv"<<marketdata_snapshot_ptr[count-1].iopv<<std::endl;
std::cout<<"etf_buy_number"
        <<marketdata_snapshot_ptr[count-1].etf_buy_number<<std::endl;
std::cout<<"etf_buy_amount"
        <<marketdata_snapshot_ptr[count-1].etf_buy_amount<<std::endl;
std::cout<<"etf_buy_money"
        <<marketdata_snapshot_ptr[count-1].etf_buy_money<<std::endl;
std::cout<<"etf_sell_number"
        <<marketdata_snapshot_ptr[count-1].etf_sell_number<<std::endl;
std::cout<<"etf_sell_amount"
        <<marketdata_snapshot_ptr[count-1].etf_sell_amount<<std::endl;
std::cout<<"etf_sell_money"
        <<marketdata_snapshot_ptr[count-1].etf_sell_money<<std::endl;
std::cout<<"yield_to_maturity"
        <<marketdata_snapshot_ptr[count-1].yield_to_maturity<<std::endl;
std::cout<<"total_warrant_exec_qty"
        <<marketdata_snapshot_ptr[count-1].total_warrant_exec_qty<<std::endl;
std::cout<<"war_lower_px"<<marketdata_snapshot_ptr[count-1].war_lower_px<<std::endl;
std::cout<<"war_upper_px"<<marketdata_snapshot_ptr[count-1].war_upper_px<<std::endl;
std::cout<<"withdraw_buy_number"
        <<marketdata_snapshot_ptr[count-1].withdraw_buy_number<<std::endl;
std::cout<<"withdraw_buy_amount"
        <<marketdata_snapshot_ptr[count-1].withdraw_buy_amount<<std::endl;
std::cout<<"withdraw_buy_money"
        <<marketdata_snapshot_ptr[count-1].withdraw_buy_money<<std::endl;
std::cout<<"withdraw_sell_number"
        <<marketdata_snapshot_ptr[count-1].withdraw_sell_number<<std::endl;
std::cout<<"withdraw_sell_amount"
```

```
            <<marketdata_snapshot_ptr[count-1].withdraw_sell_amount<<std::endl;
    std::cout<<"withdraw_sell_money"
            <<marketdata_snapshot_ptr[count-1].withdraw_sell_money<<std::endl;
    std::cout<<"total_bid_number"
            <<marketdata_snapshot_ptr[count-1].total_bid_number<<std::endl;
    std::cout<<"total_offer_number"
            <<marketdata_snapshot_ptr[count-1].total_offer_number<<std::endl;
    std::cout<<"bid_trade_max_duration"
            <<marketdata_snapshot_ptr[count-1].bid_trade_max_duration<<std::endl;
    std::cout<<"offer_trade_max_duration"
            <<marketdata_snapshot_ptr[count-1].offer_trade_max_duration<<std::endl;
    std::cout<<"num_bid_orders"
            <<marketdata_snapshot_ptr[count-1].num_bid_orders<<std::endl;
    std::cout<<"num_offer_orders"
            <<marketdata_snapshot_ptr[count-1].num_offer_orders<<std::endl;
    std::cout<<"bid_px[0]"<<marketdata_snapshot_ptr[count-1].bid_px[0]<<std::endl;
    std::cout<<"bid_volume[0]"<<marketdata_snapshot_ptr[count-1].bid_volume[0]<<std::endl;
    std::cout<<"offer_px[0]"<<marketdata_snapshot_ptr[count-1].offer_px[0]<<std::endl;
    std::cout<<"offer_volume[0]"
            <<marketdata_snapshot_ptr[count-1].offer_volume[0]<<std::endl;
    std::cout<<"bid_one_volume[0]"
            <<marketdata_snapshot_ptr[count-1].bid_one_volume[0]<<std::endl;
    std::cout<<"offer_one_volume[0]"
            <<marketdata_snapshot_ptr[count-1].offer_one_volume[0]<<std::endl;
}
void OnCashAuctionVirtualAuctionPriceSnapshot(void* user_ptr, const
LDDSCashAuctionVirtualAuctionPriceSnapshot* virtual_auction_price_snapshot_ptr, const
uint32_t& count)
{
    std::cout<<"OnCashAuctionVirtualAuctionPriceSnapshot,count:"<<count<<std::endl;
    std::cout<<"category_id"
            <<virtual_auction_price_snapshot_ptr[count-1].category_id<<std::endl;
    std::cout<<"msg_seq_id"
            <<virtual_auction_price_snapshot_ptr[count-1].msg_seq_id<<std::endl;
    std::cout<<"date_time_stamp"
            <<virtual_auction_price_snapshot_ptr[count-1].date_time_stamp<<std::endl;
    std::cout<<"security_id[8]"
            <<virtual_auction_price_snapshot_ptr[count-1].security_id[8]<<std::endl;
    std::cout<<"price"<<virtual_auction_price_snapshot_ptr[count-1].price<<std::endl;
    std::cout<<"virtual_auction_qty"
            <<virtual_auction_price_snapshot_ptr[count-1].virtual_auction_qty<<std::endl;
    std::cout<<"leave_qty"<<virtual_auction_price_snapshot_ptr[count-1].leave_qty<<std::endl;
```

```
    std::cout<<"side"<<virtual_auction_price_snapshot_ptr[count-1].side<<std::endl;
}
void OnIndexMarketDataSnapshot(void* user_ptr, const LDDSIndexMarketDataSnapshot*
index_marketdata_snapshot_ptr, const uint32_t& count)
{
    std::cout<<"OnIndexMarketDataSnapshot,count:"<<count<<std::endl;
    std::cout<<"category_id"
            <<index_marketdata_snapshot_ptr[count-1].category_id<<std::endl;
    std::cout<<"msg_seq_id"
            <<index_marketdata_snapshot_ptr[count-1].msg_seq_id<<std::endl;
    std::cout<<"date_time_stamp"
            <<index_marketdata_snapshot_ptr[count-1].date_time_stamp<<std::endl;
    std::cout<<"security_id[8]"
            <<index_marketdata_snapshot_ptr[count-1].security_id<<std::endl;
    std::cout<<"pre_close_index"
            <<index_marketdata_snapshot_ptr[count-1].pre_close_index<<std::endl;
    std::cout<<"open_index"
            <<index_marketdata_snapshot_ptr[count-1].open_index<<std::endl;
    std::cout<<"turnover"<<index_marketdata_snapshot_ptr[count-1].turnover<<std::endl;
    std::cout<<"high_index"<<index_marketdata_snapshot_ptr[count-1].high_index<<std::endl;
    std::cout<<"low_index"<<index_marketdata_snapshot_ptr[count-1].low_index<<std::endl;
    std::cout<<"last_index"<<index_marketdata_snapshot_ptr[count-1].last_index<<std::endl;
    std::cout<<"trade_time"<<index_marketdata_snapshot_ptr[count-1].trade_time<<std::endl;
    std::cout<<"total_volume_traded"
            <<index_marketdata_snapshot_ptr[count-1].total_volume_traded;
    std::cout<<"close_index"
            <<index_marketdata_snapshot_ptr[count-1].close_index<<std::endl;
}
void OnMarketOverviewSnapshot(void* user_ptr, const LDDSMarketOverviewSnapshot*
market_overview_snapshot_ptr, const uint32_t& count)
{
    std::cout<<"OnMarketOverviewSnapshot,count:"<<count<<std::endl;
    std::cout<<"category_id"<<market_overview_snapshot_ptr[count-1].category_id<<std::endl;
    std::cout<<"msg_seq_id"<<market_overview_snapshot_ptr[count-1].msg_seq_id<<std::endl;
    std::cout<<"date_time_stamp"
            <<market_overview_snapshot_ptr[count-1].date_time_stamp<<std::endl;
    std::cout<<"security_id[8]"
            <<market_overview_snapshot_ptr[count-1].security_id<<std::endl;
    std::cout<<"orig_time"<<market_overview_snapshot_ptr[count-1].orig_time<<std::endl;
    std::cout<<"orig_date"<<market_overview_snapshot_ptr[count-1].orig_date<<std::endl;
}
void OnCashAuctionMarketDataTickExecution(void* user_ptr, const
```

```
LDDSCashAuctionMarketDataTickExecution* maket_data_tick_execution_ptr, const uint32_t&
count)
{
    std::cout<<"OnCashAuctionMarketDataTickExecution,count:"<<count<<std::endl;
    std::cout<<"category_id"<<maket_data_tick_execution_ptr[count-1].category_id<<std::endl;
    std::cout<<"msg_seq_id"<<maket_data_tick_execution_ptr[count-1].msg_seq_id<<std::endl;
    std::cout<<"trade_index"<<maket_data_tick_execution_ptr[count-1].trade_index<<std::endl;
    std::cout<<"trade_channel"
            <<maket_data_tick_execution_ptr[count-1].trade_channel<<std::endl;

std::cout<<"security_id[8]"<<maket_data_tick_execution_ptr[count-1].security_id<<std::endl;
    std::cout<<"trade_time"<<maket_data_tick_execution_ptr[count-1].trade_time<<std::endl;
    std::cout<<"trade_price"<<maket_data_tick_execution_ptr[count-1].trade_price<<std::endl;
    std::cout<<"trade_qty"<<maket_data_tick_execution_ptr[count-1].trade_qty<<std::endl;
    std::cout<<"trade_money"
            <<maket_data_tick_execution_ptr[count-1].trade_money<<std::endl;
    std::cout<<"trade_buy_no"
            <<maket_data_tick_execution_ptr[count-1].trade_buy_no<<std::endl;
    std::cout<<"trade_sell_no"
            <<maket_data_tick_execution_ptr[count-1].trade_sell_no<<std::endl;
    std::cout<<"trade_bs_flag"
            <<maket_data_tick_execution_ptr[count-1].trade_bs_flag<<std::endl;
}

int main(int argc, char* argv[])
{
    LDDSConnInfo conn_info;
    memset(&conn_info, 0, sizeof(conn_info));
    strncpy_s(conn_info.realtime_conn_info.ip, "192.168.102.48", 32);
    strncpy_s(conn_info.realtime_conn_info.sender_comp_id, "VSS", 32);
    strncpy_s(conn_info.realtime_conn_info.target_comp_id, "VDE", 32);
    conn_info.realtime_conn_info.port = 9132;
    strncpy_s(conn_info.retran_conn_info.ip, "192.168.102.48", 32);
    strncpy_s(conn_info.retran_conn_info.sender_comp_id, "VSS", 32);
    strncpy_s(conn_info.retran_conn_info.target_comp_id, "VDE", 32);
    conn_info.retran_conn_info.port = 9232;
    conn_info.heartbeat_interval = 10;
    conn_info.retran_interval = 10;
    conn_info.retran_times = 5;
    conn_info.multi_callback = true;
    LDDSCallParam callback;
    callback.user_ptr = nullptr;
```

```
    callback.OnLog = OnLog;
    callback.OnCashAuctionMarketDataSnapshotLevel1 =
OnCashAuctionMarketDataSnapshotLevel1;
    callback.OnCashAuctionMarketStatusLevel1 = OnCashAuctionMarketStatusLevel1;
    callback.OnCashAuctionMarketDataSnapshotLevel2 =
OnCashAuctionMarketDataSnapshotLevel2;
    callback.OnCashAuctionVirtualAuctionPriceSnapshot =
OnCashAuctionVirtualAuctionPriceSnapshot;
    callback.OnIndexMarketDataSnapshot = OnIndexMarketDataSnapshot;
    callback.OnMarketOverviewSnapshot = OnMarketOverviewSnapshot;
    callback.OnCashAuctionMarketDataTickExecution =
OnCashAuctionMarketDataTickExecution;
    int32_t err_code;
    char err_msg[API_MAXLEN_ERRORSTR];
    void* handle;
    if ((handle = LDDSCreate(conn_info, callback, err_code, err_msg)) != nullptr)
    {
        std::cout << "LDDSCreate success" << std::endl;
    }
    else
    {
        std::cout << "LDDSCreate failed, err_code: " << err_code << ", err_msg: " << err_msg <<
std::endl;
        return -1;
    }

    if (LDDSStart(handle) == 0)
    {
        std::cout << "LDDSStart success" << std::endl;
    }
    else
    {
        std::cout << "LDDSStart failed" << std::endl;
        return -1;
    }
    getchar();
    if (LDDSStop(handle) == 0)
    {
        std::cout << "LDDSStop success" << std::endl;
    }
    else
    {
```

```
        std::cout << "LDDSStop failed" << std::endl;
        return -1;
    }

    LDDSDestroy(handle);
    getchar();
    return 0;
}
```

7. Java 接口

Java 接口相对于 C 语言接口，在使用上类似。使用方式为，通过继承接口类，实现接口方法，即可响应底层线程调用的回调函数。

7.1. Java 接口编程示例

```
import com.archforce.amd.lddsapi.*;

import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.Arrays;

public class Test {

    static String dataPathCashAuctionMarketDataSnapshotLevel1 = "./snapshot_level1.csv";
    static String dataPathCashAuctionMarketStatusLevel1= "./status_level1.csv";
    static String dataPathCashAuctionMarketDataSnapshotLevel2 = "snapshot_level2.csv";
    static String dataPathCashAuctionVirtualAuctionPriceSnapshot = "virtual_snapshot.csv";
    static String dataPathIndexMarketDataSnapshot = "snapshot_index.csv";
    static String dataPathMarketOverviewSnapshot = "snapshot_overview.csv";
    static String dataPathTickExecution = "tick_execution.csv";

    static {
        try {
            String libPath = System.getProperty("user.dir") + "/lib/ldds_api.dll";
            System.load(libPath);
            ArrayList<ArrayList<String>> csvDataCashAuctionMarketDataSnapshotLevel1 =
```

```
new ArrayList<ArrayList<String>>();
            csvDataCashAuctionMarketDataSnapshotLevel1.add(new
ArrayList<String>(Arrays.asList("行情记录标签", "产品代码", "产品简称", "成交笔数", "成交量", "成交
金额","昨收盘价", "昨结算价", "未平仓合约数量", "成交价", "指数", "今开盘价", "今收盘价", "今结算价", "
当日最高成交价","当日最低成交价", "基金 IOPV", "最新净值", "动态参考价格", "申买价", "申买量", "申卖
价", "申卖量", "当前产品状态")));
            ArrayToCSV(csvDataCashAuctionMarketDataSnapshotLevel1,
dataPathCashAuctionMarketDataSnapshotLevel1, false);
            ArrayList<ArrayList<String>> csvDataCashAuctionMarketStatusLevel1 = new
ArrayList<ArrayList<String>>();
            csvDataCashAuctionMarketStatusLevel1.add(new ArrayList<String>(Arrays.asList("
证券类型", "交易盘交易模式", "全市场行情状态", "最大产品数目")));
            ArrayToCSV(csvDataCashAuctionMarketStatusLevel1,
dataPathCashAuctionMarketStatusLevel1, false);

            ArrayList<ArrayList<String>> csvDataCashAuctionMarketDataSnapshotLevel2 =
new ArrayList<ArrayList<String>>();
            csvDataCashAuctionMarketDataSnapshotLevel2.add(new
ArrayList<String>(Arrays.asList("产品类别", "消息序号", "最新订单时间", "证券代码", "昨收盘价", "开
盘价","最高价", "最低价", "现价", "今日收盘价格", "当前品种交易状态", "当前产品状态", "成交笔数", "成
交总量", "成交总金额","委托买入总量", "加权平均委买价格", "债券加权平均委买价格", "委托卖出总量", "
加权平均委卖价格", "债券加权平均委买价格", "ETF 净值估值", "ETF 申购笔数", "ETF 申购数量", "ETF
申购金额", "ETF 赎回笔数", "ETF 赎回数量", "ETF 申购金额", "债券到期收益率", "权证执行的总数量", "
债券质押式回购品种加权平均价", "权证涨停价格", "买入撤单笔数","买入撤单数量", "买入撤单金额", "卖
出撤单笔数", "卖出撤单数量", "卖出撤单金额", "买入总笔数", "卖出总笔数", "买入委托成交最大等待时间
", "卖出委托成交最大等待时间","买方委托价位数", "卖方委托价位数", "申买价", "申买量", "申卖价", "申
卖量", "买一前 50 笔委托数量", "卖一前 50 笔委托数量")));
            ArrayToCSV(csvDataCashAuctionMarketDataSnapshotLevel2,
dataPathCashAuctionMarketDataSnapshotLevel2, false);
            ArrayList<ArrayList<String>> csvDataCashAuctionVirtualAuctionPriceSnapshot =
new ArrayList<ArrayList<String>>();
            csvDataCashAuctionVirtualAuctionPriceSnapshot.add(new
ArrayList<String>(Arrays.asList("产品类别", "消息序号", "行情时间", "证券代码", "虚拟参考价格", "虚
拟匹配量","虚拟未匹配量", "匹配标志")));
            ArrayToCSV(csvDataCashAuctionVirtualAuctionPriceSnapshot,
dataPathCashAuctionVirtualAuctionPriceSnapshot, false);
            ArrayList<ArrayList<String>> csvDataIndexMarketDataSnapshot = new
ArrayList<ArrayList<String>>();
            csvDataIndexMarketDataSnapshot.add(new ArrayList<String>(Arrays.asList("产品类
别", "消息序号", "行情时间", "证券代码",  "前收盘指数", "今开盘指数", "参与计算相应指数的成交金额","
最高指数", "最低指数", "最新指数", "成交时间", "今日收盘指数")));
            ArrayToCSV(csvDataIndexMarketDataSnapshot,
```

```
dataPathIndexMarketDataSnapshot, false);
                ArrayList<ArrayList<String>> csvDataMarketOverviewSnapshot = new
ArrayList<ArrayList<String>>();
                csvDataMarketOverviewSnapshot.add(new ArrayList<String>(Arrays.asList("消息序号
", "产品代码", "行情时间", "证券代码", "市场时间", "市场日期")));
                ArrayToCSV(csvDataMarketOverviewSnapshot, dataPathMarketOverviewSnapshot,
false);
                ArrayList<ArrayList<String>> csvDataTickExecution = new
ArrayList<ArrayList<String>>();
                csvDataTickExecution.add(new ArrayList<String>(Arrays.asList("产品类别", "消息序号
", "成交序号", "成交通道", "证券代码", "成交时间","成交价格", "成交数量", "成交金额", "买方订单号", "卖
方订单号", "内外盘标志")));
                ArrayToCSV(csvDataTickExecution, dataPathTickExecution, false);

            } catch (Exception e) {
                System.err.println("Native code library failed to load. " + e);
                System.exit(1);
            }
        }

        public static void ArrayToCSV(ArrayList<ArrayList<String>> data, String path, Boolean
append){
            try{
                BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream(path, append), "UTF-8"));
                for (int i = 0; i < data.size(); ++i){
                    ArrayList<String> row = data.get(i);
                    for(int j = 0; j < row.size(); ++j){
                        out.write(DelQuota(row.get(j)));
                        out.write(",");
                    }
                    out.newLine();
                }
                out.flush();
                out.close();
            }catch(Exception e){
                e.printStackTrace();
            }
        }

        public static String DelQuota(String str){
            String result = str;
```

```
        String[] strQuota = {"~", "!", "@", "#", "$", "%", "^", "&", "*", "(", ")", "`", ";", "'", ",", "/", ":", "/,", "<", ">", "?"};
        for (int i = 0; i < strQuota.length; ++i){
            result = result.replace(strQuota[i], "");
        }
        return result;
    }

    static class LDDSSpiImpl extends LDDSSpiInterface {
        public LDDSSpiImpl() {
            super();
        }

        @Override
        public void OnLog(LOGLevel logLevel, String s) {
            super.OnLog(logLevel, s);
            System.out.println(s);
        }

        static String SerializeDoubleArray(SWIGTYPE_p_double doubleArray, long l)
        {
            String result = "";
            for(long i = 0; i < l; ++i)
            {
                result += LDDSApiTools.GetDoubleByIndex(doubleArray, i) + " ";
            }
            return result;
        }

        static String SerializeIntArray(SWIGTYPE_p_int intArray, long l)
        {
            String result = "";
            for(long i = 0; i < l; ++i)
            {
                result += LDDSApiTools.GetInt32ByIndex(intArray, i) + " ";
            }
            return result;
        }

        static String SerializeInt64Array(SWIGTYPE_p_long_long int64Array, long l)
        {
            String result = "";
```

```
            for(long i = 0; i < l; ++i)
            {
                result += LDDSApiTools.GetInt64ByIndex(int64Array, i) + " ";
            }
            return result;
        }

        static String SerializeUint64Array(SWIGTYPE_p_unsigned_long_long uint64Array, long l)
        {
            String result = "";
            for(long i = 0; i < l; ++i)
            {
                result += LDDSApiTools.GetUint64ByIndex(uint64Array, i) + " ";
            }
            return result;
        }

        @Override
        public void
OnCashAuctionMarketDataSnapshotLevel1(LDDSCashAuctionMarketDataSnapshotLevel1
lddsCashAuctionMarketDataSnapshotLevel1, long l) {
            System.out.printf("Receive CashAuctionMarketDataSnapshotLevel1 message, count:
%d\n", l);
            ArrayList<ArrayList<String>> csvData = new ArrayList<ArrayList<String>>();
            for(long i = 0; i < l; ++i){
                LDDSCashAuctionMarketDataSnapshotLevel1 item =
LDDSApiTools.GetLDDSCashAuctionMarketDataSnapshotLevel1ByIndex(lddsCashAuctionMarket
DataSnapshotLevel1, i);
                csvData.add(new ArrayList<String>(Arrays.asList(item.getMd_stream_id(),
item.getSecurity_id(),
                        item.getSymbol(), item.getNum_trades()+"",
item.getTrade_volume()+"",
                        item.getTrade_value_traded()+"", item.getPrev_close_px()+"",
item.getPrev_set_px()+"",
                        item.getTotal_long_position()+"", item.getMatch_px()+"",
item.getIndex()+"",
                        item.getOpen_price()+"", item.getClose_price()+"",
item.getSettle_price()+"",
                        item.getHigh_px()+"", item.getLow_px()+"", item.getIopv()+"",
                        item.getNav()+"", item.getDynamic_price()+"",
SerializeDoubleArray(item.getBid_price(), 5),
                        SerializeUint64Array(item.getBid_volume(), 5),
```

```
SerializeDoubleArray(item.getOffer_price(), 5), SerializeUint64Array(item.getOffer_volume(), 5),
                        item.getTrading_phase_code())));
            }
            ArrayToCSV(csvData, dataPathCashAuctionMarketDataSnapshotLevel1, true);
        }

        @Override
        public void OnCashAuctionMarketStatusLevel1(LDDSCashAuctionMarketStatusLevel1
lddsCashAuctionMarketStatusLevel1, long l) {
            System.out.printf("Receive CashAuctionMarketStatusLevel1 message, count: %d\n",
l);
            ArrayList<ArrayList<String>> csvData = new ArrayList<ArrayList<String>>();
            for(long i = 0; i < l; ++i){
                LDDSCashAuctionMarketStatusLevel1 item =
LDDSApiTools.GetLDDSCashAuctionMarketStatusLevel1ByIndex(lddsCashAuctionMarketStatusL
evel1, i);
                csvData.add(new ArrayList<String>(Arrays.asList(item.getSecurity_type(),
item.getTrade_ses_mode()+"",
                        item.getTrading_session_id(), item.getTot_no_related_sym()+"")));
            }
            ArrayToCSV(csvData, dataPathCashAuctionMarketStatusLevel1, true);
        }

        @Override
        public void
OnCashAuctionMarketDataSnapshotLevel2(LDDSCashAuctionMarketDataSnapshotLevel2
lddsCashAuctionMarketDataSnapshotLevel2, long l) {
            System.out.printf("Receive CashAuctionMarketDataSnapshotLevel2 message, count:
%d\n", l);
            ArrayList<ArrayList<String>> csvData = new ArrayList<ArrayList<String>>();
            for(long i = 0; i < l; ++i){
                LDDSCashAuctionMarketDataSnapshotLevel2 item =
LDDSApiTools.GetLDDSCashAuctionMarketDataSnapshotLevel2ByIndex(lddsCashAuctionMarket
DataSnapshotLevel2, i);
                csvData.add(new ArrayList<String>(Arrays.asList(item.getCategory_id() + "",
item.getMsg_seq_id()+"", item.getDate_timestamp()+"",
                        item.getSecurity_id(), item.getPre_close_px()+"", item.getOpen_px()+"",
item.getHigh_px()+"",
                        item.getLow_px()+"", item.getLast_px()+"", item.getClose_px()+"",
                        item.getInstrument_status(), item.getTrading_phase_code(),
item.getNum_trades()+"", item.getTotal_volume_trade()+"",
                        item.getTotal_value_trade()+"", item.getTotal_bid_qty()+"",
```

```
item.getWeighted_avg_bid_px()+"",
                        item.getAlt_weighted_avg_bid_px()+"", item.getTotal_offer_qyt()+"",
item.getWeighted_avg_offer_px()+"",
                        item.getAlt_weighted_avg_offer_px()+"",item.getIopv()+"",
item.getEtf_buy_number()+"", item.getEtf_buy_amount()+"",
                        item.getEtf_buy_money()+"", item.getEtf_sell_number()+"",
item.getEtf_sell_amount()+"", item.getEtf_sell_money()+"",
                        item.getYield_to_maturity()+"", item.getTotal_warrant_exec_qty()+"",
item.getWar_lower_px()+"", item.getWar_upper_px()+"",
                        item.getWithdraw_buy_number()+"",
item.getWithdraw_buy_amount()+"", item.getWithdraw_buy_money()+"",
                        item.getWithdraw_sell_number()+"",
item.getWithdraw_sell_amount()+"", item.getWithdraw_sell_money()+"",
                        item.getTotal_bid_number()+"", item.getTotal_offer_number()+"",
item.getBid_trade_max_duration()+"",
                        item.getOffer_trade_max_duration()+"", item.getNum_bid_orders()+"",
item.getNum_offer_orders()+"",
                        SerializeIntArray(item.getBid_px(),10),
SerializeInt64Array(item.getBid_volume(),10), SerializeIntArray(item.getOffer_px(),10),
                        SerializeInt64Array(item.getOffer_volume(),10),
SerializeInt64Array(item.getBid_one_volume(),10),
SerializeInt64Array(item.getOffer_one_volume(),10))));
                }
                ArrayToCSV(csvData, dataPathCashAuctionMarketDataSnapshotLevel2, true);
            }

            @Override
            public void
OnCashAuctionVirtualAuctionPriceSnapshot(LDDSCashAuctionVirtualAuctionPriceSnapshot
lddsCashAuctionVirtualAuctionPriceSnapshot, long l) {
                System.out.printf("Receive CashAuctionVirtualAuctionPriceSnapshot message,
count: %d\n", l);
                ArrayList<ArrayList<String>> csvData = new ArrayList<ArrayList<String>>();
                for(long i = 0; i < l; ++i){
                    LDDSCashAuctionVirtualAuctionPriceSnapshot item =
LDDSApiTools.GetLDDSCashAuctionVirtualAuctionPriceSnapshotByIndex(lddsCashAuctionVirtua
lAuctionPriceSnapshot, i);
                    csvData.add(new ArrayList<String>(Arrays.asList(item.getCategory_id()+"",
item.getMsg_seq_id()+"", item.getDate_time_stamp()+"",
                        item.getSecurity_id(), item.getPrice()+"",
item.getVirtual_auction_qty()+"",
                        item.getLeave_qty()+"", item.getSide()+"")));
```

```
            }
            ArrayToCSV(csvData, dataPathCashAuctionVirtualAuctionPriceSnapshot, true);
        }

        @Override
        public void OnIndexMarketDataSnapshot(LDDSIndexMarketDataSnapshot
lddsIndexMarketDataSnapshot, long l) {
            System.out.printf("Receive IndexMarketDataSnapshot message, count: %d\n", l);
            ArrayList<ArrayList<String>> csvData = new ArrayList<ArrayList<String>>();
            for(long i = 0; i < l; ++i){
                LDDSIndexMarketDataSnapshot item =
LDDSApiTools.GetLDDSIndexMarketDataSnapshotByIndex(lddsIndexMarketDataSnapshot, i);
                csvData.add(new ArrayList<String>(Arrays.asList(item.getCategory_id()+"",
item.getMsg_seq_id()+"", item.getDate_time_stamp()+"",
                        item.getSecurity_id(), item.getPre_close_index()+"",
item.getOpen_index()+"", item.getTurnover()+"", item.getHigh_index()+"",
                        item.getLow_index()+"", item.getLast_index()+"",
item.getTrade_time()+"", item.getTotal_volume_traded()+"", item.getClose_index()+"")));
            }
            ArrayToCSV(csvData, dataPathIndexMarketDataSnapshot, true);
        }

        @Override
        public void OnMarketOverviewSnapshot(LDDSMarketOverviewSnapshot
lddsMarketOverviewSnapshot, long l) {
            System.out.printf("Receive MarketOverviewSnapshot message, count: %d\n", l);
            ArrayList<ArrayList<String>> csvData = new ArrayList<ArrayList<String>>();
            for(long i = 0; i < l; ++i){
                LDDSMarketOverviewSnapshot item =
LDDSApiTools.GetLDDSMarketOverviewSnapshotByIndex(lddsMarketOverviewSnapshot, i);
                csvData.add(new ArrayList<String>(Arrays.asList(item.getCategory_id()+"",
item.getMsg_seq_id()+"", item.getDate_time_stamp()+"",
                        item.getSecurity_id(), item.getOrig_time()+"", item.getOrig_date()+"")));
            }
            ArrayToCSV(csvData, dataPathMarketOverviewSnapshot, true);
        }

        @Override
        public void
OnCashAuctionMarketDataTickExecution(LDDSCashAuctionMarketDataTickExecution
lddsCashAuctionMarketDataTickExecution, long l) {
            System.out.printf("Receive CashAuctionMarketDataTickExecution message, count:
```

```
%d\n", l);
                ArrayList<ArrayList<String>> csvData = new ArrayList<ArrayList<String>>();
                for(long i = 0; i < l; ++i){
                    LDDSCashAuctionMarketDataTickExecution item =
LDDSApiTools.GetLDDSCashAuctionMarketDataTickExecutionByIndex(lddsCashAuctionMarketD
ataTickExecution, i);
                    csvData.add(new ArrayList<String>(Arrays.asList(item.getCategory_id()+"",
item.getMsg_seq_id()+"", item.getTrade_index()+"",
                            item.getTrade_channel()+"", item.getSecurity_id(),
item.getTrade_time()+"",
                            item.getTrade_price()+"", item.getTrade_qty()+"",
item.getTrade_money()+"",
                            item.getTrade_buy_no()+"", item.getTrade_sell_no()+"",
item.getTrade_bs_flag()+"")));
                }
                ArrayToCSV(csvData, dataPathTickExecution, true);
            }
        }

        public static void main(String[] args) {

            LDDSSpiInterface spi = new LDDSSpiImpl();
            LDDSApiInterface api = LDDSApiInterface.CreateLDDSApi(spi);
            if(null != api){
                LDDSConnInfo connInfo = new LDDSConnInfo();
                connInfo.setHeartbeat_interval(10);
                connInfo.setRetran_times(3);
                connInfo.setRetran_interval(1000);

               ConnInfo realtimeConn = new ConnInfo();
                realtimeConn.setIp("192.168.102.48");
                realtimeConn.setPort(9132);
                realtimeConn.setSender_comp_id("VSS");
                realtimeConn.setTarget_comp_id("VDE");
                connInfo.setRealtime_conn_info(realtimeConn);

                ConnInfo retranConn = new ConnInfo();
                retranConn.setIp("192.168.102.48");
                retranConn.setPort(9232);
                retranConn.setSender_comp_id("VSS");
                retranConn.setTarget_comp_id("VDE");
                connInfo.setRetran_conn_info(retranConn);
```

```
                connInfo.setMulti_callback(true);
                if(api.Init(connInfo) == 0){
                    System.out.println("Init success");
                }
                else{
                    System.out.println("Init failed");
                    return;
                }

                if(api.Start() == 0){
                    System.out.println("Start success");
                }
                else{
                    System.out.println("Start failed");
                    return;
                }
                while(true){
                    try {
                        Thread.sleep(100000000);
                        break;
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                if(api.Stop() == 0){
                    System.out.println("Stop success");
                }
                else{
                    System.out.println("Stop failed");
                    return;
                }
            }
        }
}
```

8. 注意事项

8.1. C 语言 LDDSApi 线程相关问题

LDDSApi 的函数线程安全性可参见 4.3.1 章节。回调函数由 LDDSApi 底层线程负责执行，为了保证 LDDSApi 的处理性能，不建议在回调函数中执行较耗时的函数，以免阻塞底层工作线程。

8.2. Java 语言 LDDSApi 线程安全问题

Java 语言版本的线程安全性与 C 语言版本一致，在使用上可参考上一小节。