行情接收 api 开发指导手册

2020 年 12 月

目录

第一章 开发接口 5
1.1 数据结构 5
1.1.1 sze_hpf_head 协议头(深交所每个行情都含有该结构) 5
1.1.2 price_quantity_unit 深交所 lev2 快照行情买卖十档结构体 5
1.1.3 sze_heartbeat 深交所 lev2 行情心跳结构体 5
1.1.4 sze_hpf_lev2 深交所 lev2 快照行情 6
1.1.5 sze_hpf_idx 深交所指数行情 7
1.1.6 sze_hpf_order 深交所逐笔委托行情 7
1.1.7 sze_hpf_exe 深交所逐笔成交行情 8
1.1.8 sze_hpf_tree 深交所建树行情 8
1.1.9 sze_hpf_close_px 深交所盘后定价行情 9
1.1.10 sze_hpf_ibr_tree（深交所 ibr 建树快照行情） 10
1.1.11 sze_hpf_turnover（深交所成交量行情） 11
1.1.12 sze_hpf_bond_snap（深交所债券快照行情） 11
1.1.13 sze_hpf_bond_order（深交所债券逐笔委托行情） 13
1.1.14 sze_hpf_bond_exe（深交所债券逐笔成交行情） 14
1.1.15 sse_hpf_head 协议头（上交所每个行情都含有该结构） 15
1.1.16 sse_lev2_price_quantity_unit 上交所 lev2 快照行情买卖结构体 15
1.1.17 sse_heartbeat 上交所 lev2 行情心跳结构体 15
1.1.18 sse_hpf_lev2 上交所 lev2 快照行情 16
1.1.19 sse_hpf_idx 上交所指数行情 18
1.1.20 sse_hpf_exe 上交所成交订单行情 19
1.1.21 sse_option_price_quantity_unit 上交所期权买卖五档结构体 20
1.1.22 sse_hpf_stock_option 上交所期权行情 20
1.1.23 sse_hpf_tree 上交所建树快照行情 22
1.1.24 sse_hpf_order 上交所逐笔委托快照行情 24
1.1.25 sse_lev2_bond_price_quantity_unit 上交所 lev2 债券快照档位单元 25
1.1.26 sse_hpf_bond_snap 上交所债券快照行情 25
1.1.27 sse_hpf_bond_tick 上交所债券逐笔行情 28
1.1.28 sse_hpf_tick_merge 上交所逐笔合并通道行情 30

1.1.29 sse_hpf_etf 上交所 ETF 统计行情 31
1.1.30 sse_static_msg_header 上交所静态消息行情头 33
1.1.31 sse_static_msg_body 上交所静态消息行情体 33
1.2 i_efh_sze_lev2 与 i_efh_sse_lev2 接口 35
1.2.1 create_efh_sze_lev2_api_function 35
1.2.2 destroy_efh_sze_lev2_api_function 36
1.2.3 init_sze 36
1.2.4 start_sze 36
1.2.5 stop_sze 36
1.2.6 close_sze 36
1.2.7 efh_sze_lev2_error（需继承 efh_sze_lev2_api_depend） 37
1.2.8 efh_sze_lev2_debug（需继承 efh_sze_lev2_api_depend） 37
1.2.9 efh_sze_lev2_info（需继承 efh_sze_lev2_api_depend） 37
1.2.10 on_report_efh_sze_lev2_idx（需继承 efh_sze_lev2_api_event） 37
1.2.11 on_report_efh_sze_lev2_snap（需继承 efh_sze_lev2_api_event） 37
1.2.12 on_report_efh_sze_lev2_tick（需继承 efh_sze_lev2_api_event） 38
1.2.13 on_report_efh_sze_close_px（需继承 efh_sze_lev2_api_event） 38
1.2.14 on_report_efh_sze_lev2_tree（需继承 efh_sze_lev2_api_event） 38
1.2.15 on_report_efh_sze_lev2_ibr_tree（需继承 efh_sze_lev2_api_event） 38
1.2.16 on_report_efh_sze_lev2_turnover（需继承 efh_sze_lev2_api_event） 38
1.2.17 on_report_efh_sze_lev2_bond_snap（需继承 efh_sze_lev2_api_event） 39
1.2.18 on_report_efh_sze_lev2_bond_tick（需继承 efh_sze_lev2_api_event） 39
1.2.19 create_efh_sse_lev2_api_function 39
1.2.20 destroy_efh_sse_lev2_api_function 39
1.2.21 init_sse 40
1.2.22 start_sse 40
1.2.23 stop_sse 40
1.2.24 close_sse 40
1.2.25 get_static_info(需要 i_efh_sse_lev2_api) 40

1.2.26 efh_sse_lev2_error（需继承 efh_sse_lev2_api_depend）.....41
1.2.27 efh_sse_lev2_debug（需继承 efh_sse_lev2_api_depend）.....41
1.2.28 efh_sse_lev2_info（需继承 efh_sse_lev2_api_depend）.....41
1.2.29 on_report_efh_sse_lev2_idx（需继承 efh_sse_lev2_api_event）.....41
1.2.30 on_report_efh_sse_lev2_snap（需继承 efh_sse_lev2_api_event）.....41
1.2.31 on_report_efh_sse_option（需继承 efh_sse_lev2_api_even）.....42
1.2.32 on_report_efh_sse_lev2_tick（需继承 efh_sse_lev2_api_event）.....42
1.2.33 on_report_efh_sse_lev2_tree（需继承 efh_sse_lev2_api_event）.....42
1.2.34 on_report_efh_sse_lev2_bond_snap（需继承 efh_sse_lev2_api_even）
42
1.2.35 on_report_efh_sse_lev2_bond_tick（需继承 efh_sse_lev2_api_even）
43
第二章 接口字段说明.....43
2.1 深交所字段说明.....44
2.2 上交所字段说明.....46
第三章 接口应用实例.....47
3.1 功能说明.....47
3.2 配置文件.....48
3.3 使用.....57

第一章 开发接口

行情接收系统 API 提供了 i_efh_sze_quote_api 接口用于接收行情， 通过继承 efh_sze_quote_event 并重载回调函数来处理查询系统的响应.

1.1 数据结构

sze_hpf_define.h 与 sse_hpf_define.h 中定义了 api 所有对外接口中所涉及到的数据结构。

1.1.1 sze_hpf_head 协议头(深交所每个行情都含有该结构)

```
struct sze_hpf_head
{
    unsigned int          m_sequence;               /// 盛立行情序号
    unsigned short        m_reserved_1;             /// 保留字段
    unsigned short        m_reserved_2;             /// 保留字段
    unsigned char         m_message_type;           /// 行情类型
    unsigned char         m_security_type;          /// 证券类型
    unsigned char         m_sub_security_type;      /// 证券子类型
    unsigned char         m_symbol[SZE_SYMBOL_LEN]; /// 证券代码
    unsigned char         m_exchange_id;            /// 交易所编号
    unsigned long long    m_quote_update_time;      /// 行情更新时间
    unsigned short        m_channel_num;            /// 频道号
    unsigned long long    m_sequence_num;           /// 包序号
    unsigned int          m_md_stream_id;           /// 行情类别
};
```

1.1.2 price_quantity_unit 深交所 lev2 快照行情买卖十档结构体

```
struct price_quantity_unit
{
    unsigned int          m_price;                  /// 价格，实际值需要除以10000
    unsigned long long    m_quantity;               /// 数量，实际值需要除以100
};
```

1.1.3 sze_heartbeat 深交所 lev2 行情心跳结构体

```
struct sze_heartbeat
{
```

```
    unsigned int          m_sequence;                 /// 盛立行情序号
    unsigned short        m_reserved_1;               /// 保留字段
    unsigned short        m_reserved_2;               /// 保留字段
    unsigned char         m_message_type;             /// 行情类型
    unsigned char         m_reserved_3[7];            /// 保留字段
}
```

1.1.4 sze_hpf_lev2 深交所 lev2 快照行情

```
struct sze_hpf_lev2
{
    sze_hpf_lev2()
    {
        memset(this, 0, sizeof(sze_hpf_lev2));
    }

    sze_hpf_head          m_head;                     /// 包头
    unsigned char         m_trading_status;     /// 交易状态码
    //*************************************************************************************
    // Bit7-Bit4(m_trading_status&0xF0):
    // 0x00:表示启动（开市前）0x10:表示开盘集合竞价 0x20:表示连续交易 0x30:表示休市
    // 0x40:表示收盘集合竞价  0x50:表示已闭市 0x60:表示临时停牌  0x70:表示盘后交易
    // 0x80:表示波动性中断  0x15:表示无意义
    // Bit3(m_trading_status&0x08):
    // 0x08:表示此全天停牌 0x00:当成此产品可正常交易处理
    // Bit2-Bit0:预留位
    //*************************************************************************************
    unsigned long long    m_total_trade_num;          /// 成交笔数
    unsigned long long    m_total_quantity;           /// 总成交量，实际值需要除以100
    unsigned long long    m_total_value;              /// 总成交额，实际值需要除以1000000
    unsigned int          m_pre_close_price;          /// 昨收价，   实际值需要除以10000
    unsigned int          m_last_price;               /// 最新价，   实际值需要除以10000
    unsigned int          m_open_price;               /// 开盘价，   实际值需要除以10000
    unsigned int          m_day_high_price;           /// 最高价，   实际值需要除以10000
    unsigned int          m_day_low_price;            /// 最低价 ，  实际值需要除以10000
    unsigned int          m_today_close_price;        /// 收盘价，   实际值需要除以10000
    unsigned int          m_total_bid_weighted_avg_price;/// 买方挂单均价，实际值需要除
以10000
    unsigned long long    m_total_bid_quantity;       /// 买总量，  实际值需要除以100
    unsigned int          m_total_ask_weighted_avg_price;/// 卖方挂单均价，实际值需要除
以10000
    unsigned long long    m_total_ask_quantity;       /// 卖总量，实际值需要除以100
```

```
    unsigned int          m_lpv;                    /// 基金 T-1 日净值，实际值需要除以
10000
    unsigned int          m_iopv;                   /// 基金实时参考净值，实际值需要除
以10000
    unsigned int          m_upper_limit_price;      /// 涨停价，实际值需要除以10000
    unsigned int          m_low_limit_price;        /// 跌停价，实际值需要除以10000
    unsigned int          m_open_interest;          /// 合约持仓量，实际值需要除以10000
    price_quantity_unit           m_bid_unit[SZE_LEV2_DEPTH];/// 买方十档
    price_quantity_unit           m_ask_unit[SZE_LEV2_DEPTH];/// 卖方十档
};
```

1.1.5 sze_hpf_idx 深交所指数行情

```
struct sze_hpf_idx
{
    sze_hpf_idx()
    {
        memset(this, 0, sizeof(sze_hpf_idx));
    }

    sze_hpf_head          m_head;                   /// 包头
    unsigned long long    m_total_trade_num;        /// 成交笔数
    unsigned long long    m_total_quantity;         /// 总成交量，实际值需要除以100
    unsigned long long    m_total_value;            /// 总成交额，实际值需要除以1000000
    unsigned int          m_last_price;             /// 最新价，   实际值需要除以10000
    unsigned int          m_pre_close_price;        /// 昨收价，   实际值需要除以10000
    unsigned int          m_open_price;             /// 开盘价，   实际值需要除以10000
    unsigned int          m_day_high_price;         /// 最高价，   实际值需要除以10000
    unsigned int          m_day_low_price;          /// 最低价，   实际值需要除以10000
    unsigned int          m_today_close_price;      /// 收盘价，   实际值需要除以10000
    char                  m_reserved[5];                /// 保留字段
};
```

1.1.6 sze_hpf_order 深交所逐笔委托行情

```
struct sze_hpf_order
{
    sze_hpf_order()
    {
        memset(this, 0, sizeof(sze_hpf_order));
    }
```

```
    sze_hpf_head          m_head;                 /// 包头
    unsigned int          m_order_price;          /// 价格，实际值需要除以10000
    unsigned long long    m_order_quantity;       /// 数量，实际值需要除以100
    char                  m_side_flag;            /// 方向
    char                  m_order_type;           /// 订单类型
    //*************************************************************************************
    // '1'=市价  '2'=限价  'U'=本方最优
    //*************************************************************************************
    char                  m_reserved[7];                /// 保留字段
};
```

1.1.7 sze_hpf_exe 深交所逐笔成交行情

```
struct sze_hpf_exe
{
    sze_hpf_exe()
    {
        memset(this, 0, sizeof(sze_hpf_exe));
    }

    sze_hpf_head          m_head;                 /// 包头
    long long             m_trade_buy_num;        /// 买方包序号
    long long             m_trade_sell_num;       /// 卖方包序号
    unsigned int          m_trade_price;          /// 价格，实际值需要除以10000
    long long             m_trade_quantity;       /// 数量，实际值需要除以100
    char                  m_trade_type;           /// 成交类型
    //*************************************************************************************
    // '4'=撤销  'F'=成交
    //*************************************************************************************
};
```

1.1.8 sze_hpf_tree 深交所建树行情

```
struct sze_hpf_tree
{
    sze_hpf_tree()
    {
        memset(this, 0, sizeof(sze_hpf_tree));
    }

    sze_hpf_head          m_head;                 /// 包头
```

```
    long long             m_total_trade_num;        /// 成交笔数
    unsigned long long    m_total_quantity;         /// 总成交量，实际值需要除以100
    long long             m_total_value;            /// 总成交额，实际值需要除以1000000
    unsigned int          m_pre_close_price;        /// 昨收价，   实际值需要除以10000
    unsigned int          m_last_price;             /// 最新价，   实际值需要除以10000
    unsigned int          m_open_price;             /// 开盘价，   实际值需要除以10000
    unsigned int          m_day_high_price;         /// 最高价，   实际值需要除以10000
    unsigned int          m_day_low_price;          /// 最低价，   实际值需要除以10000
    unsigned int          m_today_close_price;      /// 收盘价，   实际值需要除以10000
    unsigned int          m_total_bid_weighted_avg_price;/// 买方挂单均价， 实际值需要
除以10000
    unsigned long long    m_total_bid_quantity;     /// 买总量，实际值需要除以100
    unsigned int          m_total_ask_weighted_avg_price;/// 卖方挂单均价，实际值需要除
以10000
    unsigned long long    m_total_ask_quantity;     /// 卖总量，实际值需要除以100
    unsigned int          m_upper_limit_price;      /// 涨停价，实际值需要除以10000
    unsigned int          m_low_limit_price;        /// 跌停价，实际值需要除以10000
    unsigned long long    m_market_open_total_bid;  /// Bid 边市价单挂单总量，实际值需
要除以100
    unsigned long long    m_market_open_total_ask;  /// Ask 边市价单挂单总量，实际值需
要除以100
    unsigned int          m_bid_depth;          /// Bid 边总档位数
    unsigned int          m_ask_unit;           /// Ask 边总档位数
    price_quantity_unit   m_bid_unit[SZE_LEV2_DEPTH];/// 买方十档
    price_quantity_unit   m_ask_unit[SZE_LEV2_DEPTH];/// 卖方十档
    char                  m_ch_gap_flag;            /// 1=此消息前逐笔消息有断号发生；
0=逐笔消息序号连续，正常；
    char                  m_reserved[4];            /// 保留字段
};
```

1.1.9 sze_hpf_close_px 深交所盘后定价行情

```
struct sze_hpf_after_close
{
    sze_hpf_after_close()
    {
        memset(this, 0, sizeof(sze_hpf_after_close));
    }

    sze_hpf_head          m_head;                   /// 包头
    unsigned char         m_trading_status;     /// 交易状态码
    //*********************************************************************************
    // Bit7-Bit4(m_trading_status&0xF0):
```

```
    // 0x00:表示启动（开市前）0x10:表示开盘集合竞价 0x20:表示连续交易 0x30:表示休市
    // 0x40:表示收盘集合竞价  0x50:表示已闭市 0x60:表示临时停牌  0x70:表示盘后交易
    // 0x80:表示波动性中断  0x15:表示无意义
    // Bit3(m_trading_status&0x08):
    // 0x08:表示此全天停牌 0x00:当成此产品可正常交易处理
    // Bit2-Bit0:预留位
    //*************************************************************************
    long long             m_total_trade_num;        /// 成交笔数
    unsigned long long    m_total_quantity;         /// 总成交量，实际值需要除以100
    unsigned long long    m_total_value;            /// 总成交额，实际值需要除以1000000
    unsigned int          m_pre_close_price;        /// 昨收价，   实际值需要除以10000
    unsigned int          m_trade_price;            /// 成交价，   实际值需要除以10000
    unsigned long long    m_bid_quantity;           /// 买总量，   实际值需要除以100
    unsigned long long    m_ask_quantity;           /// 卖总量，   实际值需要除以100
    char                  m_reserved[4];            /// 保留字段
};
```

1.1.10 sze_hpf_ibr_tree（深交所 ibr 建树快照行情）

```
struct sze_hpf_ibr_tree
{
    sze_hpf_ibr_tree()
    {
        memset(this, 0, sizeof(sze_hpf_ibr_tree));
    }

    sze_hpf_head              m_head;                        // 包头
    unsigned int              m_total_bid_weighted_avg_price;// 加权平均委托买价格，实
际值需要除以10000
    unsigned long long        m_total_bid_quantity;          // 委托买入总量，实际值需
要除以100
    unsigned int              m_total_ask_weighted_avg_price;// 加权平均委托卖价格，实
际值需要除以10000
    unsigned long long        m_total_ask_quantity;          // 委托卖总量，实际值需要
除以100
    unsigned int              m_bid_depth;// bid边总档位数，据300191和300192计算
    unsigned int              m_ask_depth;// ask边总档位数，据300191和300192计算
    price_quantity_unit       m_bid_unit[SZE_IBR_DEPTH];     // bid各档位价格信息
    price_quantity_unit       m_ask_unit[SZE_IBR_DEPTH];     // ask各档位价格信息
    unsigned char             m_reserved[5];                 // 保留字段
};
```

1.1.11 sze_hpf_turnover（深交所成交量行情）

```
struct sze_hpf_turnover
{
    sze_hpf_turnover()
    {
        memset(this, 0, sizeof(sze_hpf_turnover));
    }

    sze_hpf_head                    m_head;                 // 包头
    unsigned char                   m_trading_status;       // 交易阶段
    long long                       m_pre_close_px;         // 昨收价
    long long                       m_total_trade_num;              // 成交
笔数
    unsigned long long              m_total_quantity;       // 成交量，实际值需要
除以100
    unsigned long long              m_total_value;          // 成交金额，实际值需
要除以1000000
    unsigned int                    m_stock_num;            // 统计量指标样
本个数
};
```

1.1.12 sze_hpf_bond_snap（深交所债券快照行情）

```
struct sze_hpf_bond_snap
{
    sze_hpf_bond_snap()
    {
        memset( this , 0 , sizeof( sze_hpf_bond_snap ) );
    }
    sze_hpf_head                m_head;                 // 包头
    unsigned char               m_trading_status;       // 交易状态码

//*************************************************************************************
*****
    // Bit7-Bit4(m_trading_status&0xF0):
    // 0x00:表示启动（开市前）
    // 0x10:表示开盘集合竞价
    // 0x20:表示连续交易
    // 0x30:表示闭市
```

```
    // 0x40:表示合约停牌
    // 0x50:表示可恢复交易的熔断时段（盘中集合竞价）
    // 0x60:表示不可恢复交易的熔断时段（暂停交易至闭市）
    // 0x70:表示收盘集合竞价时段
    // 0x80:表示波动性中断（适用于股票期权）
    // 0x15:表示无意义
    // Bit3(m_trading_status&0x08):
    // 0x08:"表示此合约可正常交易 0x00:表示此合约不可正常交易
    // Bit2(m_trading_status&0x04):
    // 0x04:表示已上市 0x00:表示未上市
    // Bit1(m_trading_status&0x02):
    // 0x02:表示此合约在当前时段接受订单申报 0x00:表示此合约在当前时段不接受订单申报
    // Bit0:预留

//*************************************************************************************
*****
    unsigned long long              m_total_trade_num;              // 成交笔数
    unsigned long long              m_total_quantity;               // 总成交量,
实际值需要除以 100
    unsigned long long              m_total_value;                  // 总成交额,
实际值需要除以 1000000
    unsigned int                    m_pre_close_price;              // 昨收价,
实际值需要除以 10000
    unsigned int                    m_last_price;                   // 最新价,
实际值需要除以 10000
    unsigned int                    m_open_price;                   // 开盘价,
实际值需要除以 10000
    unsigned int                    m_day_high_price;               // 最高价,
实际值需要除以 10000
    unsigned int                    m_day_low_price;                // 最低价,
实际值需要除以 10000
    unsigned int                    m_today_close_price;            // 收盘价,
实际值需要除以 10000
    unsigned int                    m_total_trade_weighted_avg_price; // 加权平均价,
实际值需要除以 10000
    int                             m_fluctuation_1;                // 升跌 1,
实际值需要除以 10000
    int                             m_fluctuation_2;                // 升跌 2,
实际值需要除以 10000
    unsigned int                    m_total_bid_weighted_avg_price;  // 买方挂单均
价,       实际值需要除以 10000
    unsigned long long              m_total_bid_quantity;           // 买总量,
实际值需要除以 100
    unsigned int                    m_total_ask_weighted_avg_price;  // 卖方挂单均
```

```
价，       实际值需要除以 10000
    unsigned long long              m_total_ask_quantity;           // 卖总量，
实际值需要除以 100

    int                             m_weighted_avg_price_BP;        // 加权平均涨
跌 BP     实际值需要除以 10000
    unsigned int                    m_weighted_avg_pre_close_price; // 昨收盘加权
平均价     实际值需要除以 10000
    unsigned int                    m_auction_trade_last_price;     // 匹配成交最
新价       实际值需要除以 10000
    unsigned char                   m_reserved[4];                  // 保留字段
    unsigned long long              m_auction_volume_trade;         // 匹配成交成交
量       实际值需要除以 100
    unsigned long long              m_auction_value_trade;          // 匹配成交成交
金额     实际值需要除以 1000000

    price_quantity_unit             m_bid_unit[SZE_LEV2_DEPTH];     // 买方十档
    price_quantity_unit             m_ask_unit[SZE_LEV2_DEPTH];     // 卖方十档
};
```

1.1.13 sze_hpf_bond_order（深交所债券逐笔委托行情）

```
struct sze_hpf_bond_order
{
    sze_hpf_bond_order()
    {
        memset( this , 0 , sizeof( sze_hpf_bond_order ) );
    }

    sze_hpf_head                    m_head;                         // 包头
    /// 对逐笔委托价格转换说明如下：
    //
**********************************************************************************
************************
    //  |订单类型 \ 买卖方向 |                  买（1）          |
卖（2）                  |
    //  |     市价（1）      |                  直接透传         |
直接透传                 |
    //  |    本方最优(U)     |                  直接透传         |
直接透传                 |
    //  |     限价（2）      |   如果小于 3 的取为 3，否则取原值     |   如果大于
999999997，取为 999999997，否则取原值     |
```

```
    //
**********************************************************************************************
************************
    unsigned int                    m_price;                        // 价格，  实际
值需要除以 10000
    unsigned long long              m_quantity;                     // 数量，  实际
值需要除以 100
    char                            m_side;                         // 方向，‘1’=
买  ‘2’=卖
    char                            m_order_type;                   // 订单类型

//**********************************************************************************************
*****
    // ’1’=市价  ’2’=限价  ’U’=本方最优

//**********************************************************************************************
*****
    char                            m_reserved[15];                  // 保留字段
};
```

1.1.14 sze_hpf_bond_exe（深交所债券逐笔成交行情）

```
struct sze_hpf_bond_exe
{
    sze_hpf_bond_exe()
    {
        memset( this , 0 , sizeof( sze_hpf_bond_exe ) );
    }
    sze_hpf_head                    m_head;                         // 包头
    long long                       m_buy_num;                      // 买方包序号
    long long                       m_sell_num;                     // 卖方包序号
    unsigned int                    m_price;                        // 价格，  实际
值需要除以 10000
    long long                       m_quantity;                     // 数量，  实际
值需要除以 100
    char                            m_type;                         // 成交类型

//**********************************************************************************************
*****
    // ’4’=撤销  ’F’=成交

//**********************************************************************************************
```

```
*****
};
```

1.1.15 sse_hpf_head 协议头（上交所每个行情都含有该结构）

```
struct sse_hpf_head
{
    unsigned int          m_sequence;                /// 盛立行情序号
    unsigned short        m_reserved_1;              /// 保留字段
    unsigned short        m_reserved_2;              /// 保留字段
    unsigned char         m_message_type;            /// 行情类型
    unsigned short        m_message_len;             /// 包括此消息头的长度
    unsigned char         m_exchange_id;             /// 交易所编号
    unsigned short        m_quote_date_year;         /// 行情更新年份
    unsigned char         m_quote_date_month;        /// 行情更新月份
    unsigned char         m_quote_date_day;          /// 行情更新日期
    unsigned int          m_send_time;               /// 交易所发送时间
    unsigned char         m_category_id;             /// 数据产品类别
    unsigned int          m_msg_seq_id;              /// 行情包序号
    unsigned char         m_seq_lost_flag;           /// 1=有丢包，0=没有丢包
};
```

1.1.16 sse_lev2_price_quantity_unit　上交所 lev2 快照行情买卖结构体

```
struct sse_lev2_price_quantity_unit
{
    sse_lev2_price_quantity_unit()
    {
        memset(this, 0, sizeof(sse_lev2_price_quantity_unit));/// 初始化结构体
    }

    char                  m_lev_operator;       /// 辅助信息
    char                  m_resv[3];            /// 保留字段
    unsigned int          m_price;              /// 价格，实际值需要除以1000
    unsigned long long    m_quantity;           /// 数量，实际值需要除以1000
};
```

1.1.17 sse_heartbeat　上交所 lev2 行情心跳结构体

```
struct sse_heartbeat
{
```

```
    unsigned int          m_sequence;                ///盛立行情序号
    unsigned short        m_reserved_1;              ///保留字段
    unsigned short        m_reserved_2;              ///保留字段
    unsigned char         m_message_type;            ///行情类型
    unsigned short        m_message_len;             ///包括此消息头的长度
    unsigned char         m_reserved_3[5];           ///保留字段
}
```

1.1.18 sse_hpf_lev2 上交所 lev2 快照行情

```
struct sse_hpf_lev2
{
    sse_hpf_lev2()
    {
        memset(this, 0, sizeof(sse_hpf_lev2));                    // 初始化结构体
    }

    sse_hpf_head                    m_head;                       // 包头
    unsigned int                    m_quote_update_time;          // 行情更新时
间,格式如：HHMMSS（秒）HH=00-23, MM=00-59, SS=00-60(秒)
    char                            m_symbol[SSE_SYMBOL_LEN];     // 证券代码

//*************************************************************************************
*****
    //0：指数    1：股票 2：基金 3：债券 4：回购 5：权证 6：期货 7：外汇 8：利率 9：期权
10：其他

//*************************************************************************************
*****
    unsigned char                   m_security_type;              // 证券类型
    unsigned char                   m_image_status;               // 快照类型，1=
全量，2=更新
    unsigned char                   m_reserved_1;                 // 保留字段
    unsigned int                    m_pre_close_price;            // 昨收价，
实际值需要除以 1000
    unsigned int                    m_open_price;                 // 开盘价，
实际值需要除以 1000
    unsigned int                    m_day_high_price;             // 最高价，
实际值需要除以 1000
    unsigned int                    m_day_low_price;              // 最低价，
实际值需要除以 1000
    unsigned int                    m_last_price;                 // 最新价，
实际值需要除以 1000
    unsigned int                    m_today_close_price;          // 收盘价，
```

```
实际值需要除以 1000
    unsigned char                   m_instrument_status;            // 当前品种交易
状态
//*************************************************************************************
*****
    // 0:其他     1:启动 2:开市集合竞价 3:连续自动撮合 4:停牌 5:收盘集合竞价
    // 6:闭市，自动计算闭市价格  7:交易结束  8:产品未上市

//*************************************************************************************
*****
    unsigned char                   m_trading_status;               //当前产品状态

//*************************************************************************************
*****
    // Bit7-Bit4(m_trading_status&0xF0):
    // 0x00:表示启动（开市前）
    // 0x10:表示开盘集合竞价
    // 0x20:表示连续交易
    // 0x30:表示闭市
    // 0x40:表示合约停牌
    // 0x50:表示可恢复交易的熔断时段（盘中集合竞价）
    // 0x60:表示不可恢复交易的熔断时段（暂停交易至闭市）
    // 0x70:表示收盘集合竞价时段
    // 0x80:表示波动性中断（适用于股票期权）
    // 0x15:表示无意义
    // Bit3(m_trading_status&0x08):
    // 0x08:"表示此合约可正常交易 0x00:表示此合约不可正常交易
    // Bit2(m_trading_status&0x04):
    // 0x04:表示已上市 0x00:表示未上市
    // Bit1(m_trading_status&0x02):
    // 0x02:表示此合约在当前时段接受订单申报 0x00:表示此合约在当前时段不接受订单申报
    // Bit0:预留

//*************************************************************************************
*****
    unsigned short                  m_reserved_2;                   // 保留字段
    unsigned int                    m_total_trade_num;              // 成交笔数
    unsigned long long              m_total_quantity;               // 成交总量，
实际值需要除以 1000
    unsigned long long              m_total_value;                  // 成交总额，
实际值需要除以 100000
    unsigned long long              m_total_bid_quantity;           // 委托买入总量，
实际值需要除以 1000
    unsigned int                    m_total_bid_weighted_avg_price; // 买方加权平均
```

```
委买价格，    实际值需要除以 1000
    unsigned long long              m_total_ask_quantity;           // 委托卖出总量，
实际值需要除以 1000
    unsigned int                    m_total_ask_weighted_avg_price;  // 卖方加权平均
委买价格，    实际值需要除以 1000
    unsigned int                    m_yield_to_maturity;            // 债券到期收益
率
    unsigned char                   m_bid_depth;                    // 买盘价位数量
    unsigned char                   m_ask_depth;                    // 卖盘价位数量
    sse_lev2_price_quantity_unit    m_bid_unit[SSE_LEV2_DEPTH];     // 买边十档
    sse_lev2_price_quantity_unit    m_ask_unit[SSE_LEV2_DEPTH];     // 卖边十档
};
```

1.1.19 sse_hpf_idx　上交所指数行情

```
// 上交所lev2指数消息
struct sse_hpf_idx
{
    sse_hpf_idx()
    {
        memset(this, 0, sizeof(sse_hpf_idx));                       // 初始化结构体
    }

    sse_hpf_head                    m_head;                         // 包头
    unsigned int                    m_quote_update_time;            // 行情更新时
间,格式如：HHMMSS（秒）HH=00-23, MM=00-59, SS=00-60(秒)
    char                            m_symbol[SSE_SYMBOL_LEN];       // 证券代码

//*************************************************************************************
*****
    //0：指数    1：股票 2：基金 3：债券 4：回购 5：权证 6：期货 7：外汇 8：利率 9：期权
10：其他

//*************************************************************************************
*****
    unsigned char                   m_security_type;                // 0=指数，  10=
其他
    unsigned int                    m_pre_close_price;              // 昨收价，
实际值需要除以 100000
    unsigned int                    m_open_price;                   // 开盘价，
实际值需要除以 100000
    unsigned long long              m_total_value;                  // 成交额，
实际值需要除以 10
    unsigned int                    m_day_high_price;               // 最高价，
```

```
实际值需要除以 100000
    unsigned int                    m_day_low_price;                // 最低价，
实际值需要除以 100000
    unsigned int                    m_last_price;                   // 最新价，
实际值需要除以 100000
    unsigned long long              m_total_quantity;               // 成交量，
实际值需要除以 100000
    unsigned int                    m_today_close_price;            // 今收盘价，
实际值需要除以 100000
};
```

1.1.20 sse_hpf_exe　上交所成交订单行情

```
// 上交所lev2成交消息
struct sse_hpf_exe
{
    sse_hpf_exe()
    {
        memset(this, 0, sizeof(sse_hpf_exe));
    }

    sse_hpf_head                    m_head;                         // 包头
    unsigned int                    m_trade_index;                  // 成交序号
    unsigned int                    m_channel_num;                  // 频道号
    char                            m_symbol[SSE_SYMBOL_LEN];       // 证券代码
    unsigned int                    m_trade_time;                   // 成交时间,格
式如：HHMMSSss（百分之一秒）HH=00-23, MM=00-59, SS=00-60(秒) ss=00-99(百分之一秒)
    unsigned int                    m_trade_price;                  // 成交价，
实际值需要除以 1000
    unsigned long long              m_trade_quantity;               // 成交量，
实际值需要除以 1000
    unsigned long long              m_trade_value;                  // 成交额，
实际值需要除以 100000
    unsigned long long              m_trade_buy_num;                // 买方订单号
    unsigned long long              m_trade_sell_num;               // 卖方订单号
    char                            m_trade_bs_flag;                // 内外盘标志：
B-外盘，主动买；S-内盘，主动卖；N-未知
    unsigned long long              m_biz_index;                    // 业务序号
    unsigned char                   m_reserved[4];                  // 保留字段
};
```

1.1.21 sse_option_price_quantity_unit 上交所期权买卖五档结构体

```
// 上交所lev2期权档位单元
struct sse_option_price_quantity_unit
{
    sse_option_price_quantity_unit()
    {
        memset(this, 0, sizeof(sse_option_price_quantity_unit));
    }
    unsigned int                    m_price;                    // 价格，实际值
需要除以 10000
    unsigned long long              m_quantity;                 // 数量，实际值
需要除以 100
};
```

1.1.22 sse_hpf_stock_option 上交所期权行情

```
// 上交所lev2期权档位单元
struct sse_option_price_quantity_unit
{
    sse_option_price_quantity_unit()
    {
        memset(this, 0, sizeof(sse_option_price_quantity_unit));
    }
    unsigned int                    m_price;                    // 价格，实际值
需要除以 1000
    unsigned long long              m_quantity;                 // 数量，实际值
需要除以 1000
};

// 上交所lev2期权消息
struct sse_hpf_stock_option
{
    sse_hpf_stock_option()
    {
        memset(this, 0, sizeof(sse_hpf_stock_option));          // 初始化结构体
    }

    sse_hpf_head                    m_head;                     // 包头
    unsigned int                    m_quote_update_time;        // 行情更新时
间,格式如：HHMMSSss（百分之一秒）HH=00-23, MM=00-59, SS=00-60(秒) ss=00-99(百分之一秒)

//*****************************************************************************
```

```
*****
    //0：指数    1：股票 2：基金 3：债券 4：回购 5：权证 6：期货 7：外汇 8：利率 9：期权
10：其他

//*****************************************************************************************
*****
    unsigned char                   m_security_type;                // 证券类型
    char                            m_symbol[SSE_SYMBOL_LEN];       // 合约
    unsigned char                   m_image_status;                 // 快照类型
    unsigned char                   m_reserved_1;                   // 保留字段
    unsigned int                    m_pre_close_price;              // 昨收盘价，
实际值需要除以 10000
    unsigned int                    m_pre_settle_price;             // 昨结算价，
实际值需要除以 10000
    unsigned int                    m_open_price;                   // 开盘价，
实际值需要除以 10000
    unsigned int                    m_day_high_price;               // 最高价，
实际值需要除以 10000
    unsigned int                    m_day_low_price;                // 最低价 ，
实际值需要除以 10000
    unsigned int                    m_last_price;                   // 最新价，
实际值需要除以 10000
    unsigned int                    m_today_close_price;            // 收盘价，
实际值需要除以 10000
    unsigned int                    m_today_settle_price;           // 今结算价，
实际值需要除以 10000
    unsigned int                    m_dynamic_price;                // 动态参考价
格，    实际值需要除以 10000
    unsigned char                   m_reserved_2[3];                // 保留字段
    unsigned char                   m_trading_status;               //当前产品状态

//*****************************************************************************************
*****
    // Bit7-Bit4(m_trading_status&0xF0):
    // 0x00:表示启动（开市前）
    // 0x10:表示开盘集合竞价
    // 0x20:表示连续交易
    // 0x30:表示闭市
    // 0x40:表示合约停牌
    // 0x50:表示可恢复交易的熔断时段（盘中集合竞价）
    // 0x60:表示不可恢复交易的熔断时段（暂停交易至闭市）
    // 0x70:表示收盘集合竞价时段
    // 0x80:表示波动性中断（适用于股票期权）
    // 0x15:表示无意义
```

```
    // Bit3(m_trading_status&0x08):
    // 0x08:"表示此合约可正常交易 0x00:表示此合约不可正常交易
    // Bit2(m_trading_status&0x04):
    // 0x04:表示已上市 0x00:表示未上市
    // Bit1(m_trading_status&0x02):
    // 0x02:表示此合约在当前时段接受订单申报 0x00:表示此合约在当前时段不接受订单申报
    // Bit0:预留

//*********************************************************************************
*****
    unsigned long long              m_open_interest;                // 未平仓合约数
量， 实际值需要除以 100
    unsigned int                    m_total_trade_num;              // 成交笔数
    unsigned long long              m_total_quantity;               // 成交量，
实际值需要除以 100
    unsigned long long              m_total_value;                  // 成交金额，
实际值需要除以 1000000
    unsigned char                   m_bid_depth;                    // 申买档位数
    unsigned char                   m_ask_depth;                    // 申卖档位数
    sse_option_price_quantity_unit  m_bid_unit[SSE_OPTION_DEPTH];   // 买边五档
    sse_option_price_quantity_unit  m_ask_unit[SSE_OPTION_DEPTH];   // 卖边五档
};
```

1.1.23 sse_hpf_tree 上交所建树快照行情

```
struct sse_hpf_tree
{
    sse_hpf_tree()
    {
        memset(this, 0, sizeof(sse_hpf_tree));
    }

    sse_hpf_head                    m_head;
    unsigned int                    m_channel_num;                  // 通道号
    unsigned int                    m_quote_update_time;            // 行情时间
    char                            m_symbol[SSE_SYMBOL_LEN];       // 标的名称

//*********************************************************************************
*****
    //0：指数    1：股票 2：基金 3：债券 4：回购 5：权证 6：期货 7：外汇 8：利率 9：期权
10：其他

//*********************************************************************************
```

```
*****
    unsigned char                   m_security_type;                // 股票/合约类
型
    unsigned long long              m_biz_index;                    // 业务序列号
    unsigned int                    m_pre_close_price;              // 昨收价，
实际值需要除以 1000
    unsigned int                    m_open_price;                   // 开盘价，
实际值需要除以 1000
    unsigned int                    m_day_high_price;               // 最高价，
实际值需要除以 1000
    unsigned int                    m_day_low_price;                // 最低价，
实际值需要除以 1000
    unsigned int                    m_last_price;                   // 最新价，
实际值需要除以 1000
    unsigned int                    m_today_close_price;            // 今收盘价，
实际值需要除以 1000
    unsigned char                   m_instrument_status;            // 当前品种交易
状态

//*************************************************************************************
*****
    // 0:其他    1:启动 2:开市集合竞价 3:连续自动撮合 4:停牌 5:收盘集合竞价
    // 6:闭市，自动计算闭市价格  7:交易结束  8:产品未上市

//*************************************************************************************
*****
    unsigned char                   m_trading_status;               // 当前产品状态

//*************************************************************************************
*****
    // Bit7-Bit4(m_trading_status&0xF0):
    // 0x00:表示启动（开市前）
    // 0x10:表示开盘集合竞价
    // 0x20:表示连续交易
    // 0x30:表示闭市
    // 0x40:表示合约停牌
    // 0x50:表示可恢复交易的熔断时段（盘中集合竞价）
    // 0x60:表示不可恢复交易的熔断时段（暂停交易至闭市）
    // 0x70:表示收盘集合竞价时段
    // 0x80:表示波动性中断（适用于股票期权）
    // 0x15:表示无意义
    // Bit3(m_trading_status&0x08):
    // 0x08:"表示此合约可正常交易 0x00:表示此合约不可正常交易
    // Bit2(m_trading_status&0x04):
```

```
    // 0x04:表示已上市 0x00:表示未上市
    // Bit1(m_trading_status&0x02):
    // 0x02:表示此合约在当前时段接受订单申报 0x00:表示此合约在当前时段不接受订单申报
    // Bit0:预留

//*****************************************************************************************
*****
    unsigned char                       m_reserved_0[6];                    // 6个字节的保留
字段
    unsigned int                        m_total_trade_num;                  // 总成交笔数
    unsigned long long                  m_total_quantity;                   // 总成交量,
实际值需要除以 1000
    unsigned long long                  m_total_value;                      // 总成交金额,
实际值需要除以 100000
    unsigned long long                  m_total_bid_quantity;               // 委托买入数量,
实际值需要除以 1000
    unsigned int                        m_total_bid_weighted_avg_price;     // 加权平均委托
买入价格,    实际值需要除以 1000
    unsigned long long                  m_total_ask_quantity;               // 委托卖出总量,
实际值需要除以 1000
    int                                 m_total_ask_weighted_avg_price;     // 加权平均委托
卖价格,      实际值需要除以 1000
    int                                 m_yield_to_maturity;                // 债券到期收益
率
    unsigned char                       m_bid_depth;                        // 申买档位数
    unsigned char                       m_ask_depth;                        // 申卖档位数
    unsigned char                       m_reserved_1[2];                    // 2字节的保留字
段
    sse_lev2_tree_price_quantity_unit m_bid_unit[SSE_LEV2_DEPTH];       // 申买档位信息
    sse_lev2_tree_price_quantity_unit m_ask_unit[SSE_LEV2_DEPTH];       // 申卖档位信息
};
```

1.1.24 sse_hpf_order 上交所逐笔委托快照行情

```
// 上交所逐笔委托消息
struct sse_hpf_order
{
    sse_hpf_order()
    {
        memset(this, 0, sizeof(sse_hpf_order));
    }
```

```
    sse_hpf_head                    m_head;                         // 消息头
    unsigned int                    m_order_index;                  // 委托序号
    unsigned int                    m_channel_num;                  // 通道号
    char                            m_symbol[SSE_SYMBOL_LEN];       // 标的名称
    unsigned int                    m_order_time;                   // 委托时间,格
式如：HHMMSSss（百分之一秒）HH=00-23, MM=00-59, SS=00-60(秒) ss=00-99(百分之一秒)
    char                            m_order_type;                   // 订单类型，A=
新增订单，D=删除订单
    unsigned long long              m_order_num;                    // 原始订单号
    unsigned int                    m_order_price;                  // 委托价格，
实际值需要除以 1000
    unsigned long long              m_balance;                      // 剩余委托量，
实际值需要除以 1000
    unsigned char                   m_reserved_0[15];               // 15个字节保留
字段
    char                            m_side_flag;                    // 买卖标志,B-
主动买；S-主动卖
    unsigned long long              m_biz_index;                    // 业务序列号
    unsigned char                   m_reserved_1[4];                // 4个字节的保留
字段
};
```

1.1.25 sse_lev2_bond_price_quantity_unit 上交所 lev2 债券快照档位单元

```
// 上交所lev2债券快照档位单元
struct sse_lev2_bond_price_quantity_unit
{
    sse_lev2_bond_price_quantity_unit()
    {
        memset(this, 0, sizeof(sse_lev2_bond_price_quantity_unit));
    }
    unsigned int                    m_price;                        // 价格，    实际
值需要除以 1000
    unsigned long long              m_quantity;                     // 数量，    实际
值需要除以 1000
};
```

1.1.26 sse_hpf_bond_snap 上交所债券快照行情

```
// 上交所lev2债券快照
struct sse_hpf_bond_snap
{
    sse_hpf_bond_snap()
    {
        memset(this, 0, sizeof(sse_hpf_bond_snap));                    // 初始化结构体
    }

    sse_hpf_head                        m_head;                        // 包头
    unsigned int                        m_quote_update_time;           // 行情更新时间,
格式如：HHMMSSsss（千分之一秒）HH=00-23, MM=00-59, SS=00-60(秒) ss=000-999(千分之一秒) 如
143025002 表示 14:30:25.002
    unsigned char                       m_reserved_0[4];               // 保留字段
    char                                m_symbol[SSE_SYMBOL_LEN];      // 合约
    //
********************************************************************************
****
    //      证券类型              证券子类型
    //      0           指数      0             指数
    //                            255           未分类
    //      1           股票      0             主板
    //                            1
    //                            2             创业板
    //                            3             B股
    //                            4             H股
    //                            5             科创板
    //                            255           未分类
    //      2           基金      248           LOF基金
    //                            249           ETF基金
    //                            255           未分类
    //      3           债券      248           国债
    //                            249           可转债
    //                            255           未分类
    //      4           回购      248           国债回购
    //                            255           未分类
    //      9           期权      248           股票期权
    //                            249           ETF期权
    //                            255           未分类
    //      10          其他
    //
********************************************************************************
****
    unsigned char                       m_security_type;               // 证券类型
    unsigned char                       m_sub_security_type;           // 证券子类型
```

```
    unsigned char                   m_reserved_1[3];                // 保留字段
    unsigned int                    m_pre_close_price;              // 昨收盘价，
实际值需要除以 1000
    unsigned int                    m_open_price;                   // 开盘价，
实际值需要除以 1000
    unsigned int                    m_day_high_price;               // 最高价，
实际值需要除以 1000
    unsigned int                    m_day_low_price;                // 最低价 ，
实际值需要除以 1000
    unsigned int                    m_last_price;                   // 最新价，
实际值需要除以 1000
    unsigned int                    m_today_close_price;            // 收盘价，
实际值需要除以 1000
    unsigned char                   m_instrument_status;            // 当前品种交易
状态

//*************************************************************************************
*****
    // 0:其他     1:启动 2:开市集合竞价 3:连续自动撮合 4:停牌
    // 6:闭市，自动计算闭市价格  7:交易结束  8:产品未上市

//*************************************************************************************
*****
    unsigned char                   m_reserved_2[3];                // 保留字段
    unsigned int                    m_total_trade_num;              // 总成交笔数
    unsigned long long              m_total_quantity;               // 总成交量，
实际值需要除以 1000
    unsigned long long              m_total_value;                  // 总成交金额，
实际值需要除以 100000
    unsigned long long              m_total_bid_quantity;           // 委托买入数量，
实际值需要除以 1000
    unsigned int                    m_total_bid_weighted_avg_price; // 加权平均委托
买入价格，    实际值需要除以 1000
    unsigned long long              m_total_ask_quantity;           // 委托卖出总量，
实际值需要除以 1000
    unsigned int                    m_total_ask_weighted_avg_price; // 加权平均委托
卖价格，      实际值需要除以 1000

    unsigned int                    m_withdraw_bid_num;             // 买入撤单笔数
    unsigned long long              m_withdraw_bid_amount;          // 买入撤单数量
实际值需要除以 1000
    unsigned long long              m_withdraw_bid_price;           // 买入撤单金额
实际值需要除以 100000
    unsigned int                    m_withdraw_ask_num;             // 卖出撤单笔数
```

```
    unsigned long long              m_withdraw_ask_amount;          // 卖出撤单数量
实际值需要除以 1000
    unsigned long long              m_withdraw_ask_price;           // 卖出撤单金额
实际值需要除以 100000
    unsigned int                    m_total_bid_num;                // 买入总笔数
    unsigned int                    m_total_ask_num;                // 卖出总笔数
    unsigned int                    m_bid_trade_max_duration;       // 买入委托成交
最大等待时间
    unsigned int                    m_ask_trade_max_duration;       // 卖出委托成交
最大等待时间

    unsigned char                   m_bid_depth;                    // 申买档位数
    unsigned char                   m_ask_depth;                    // 申卖档位数
    unsigned char                   m_reserved_3[6];                // 保留字段
    sse_lev2_bond_price_quantity_unit   m_bid_unit[SSE_LEV2_DEPTH]; // 申买信息
    sse_lev2_bond_price_quantity_unit   m_ask_unit[SSE_LEV2_DEPTH]; // 申卖信息
};
```

1.1.27 sse_hpf_bond_tick 上交所债券逐笔行情

```
// 债券逐笔
struct sse_hpf_bond_tick
{
    sse_hpf_bond_tick()
    {
        memset(this, 0, sizeof(sse_hpf_bond_tick));             // 初始化结构体
    }

    sse_hpf_head                    m_head;                         // 包头
    unsigned int                    m_tick_index;                   // 序号从1开始,
按通道连续
    unsigned int                    m_channel_num;                  // 通道号
    char                            m_symbol[SSE_SYMBOL_LEN];       // 标的名称
    //
**********************************************************************************
****
    //      证券类型            证券子类型
    //      0           指数    0           指数
    //                          255         未分类
    //      1           股票    0           主板
    //                          1
    //                          2           创业板
    //                          3           B股
```

```
    //                          4               H股
    //                          5               科创板
    //                          255             未分类
    //      2           基金    248             LOF基金
    //                          249             ETF基金
    //                          255             未分类
    //      3           债券    248             国债
    //                          249             可转债
    //                          255             未分类
    //      4           回购    248             国债回购
    //                          255             未分类
    //      9           期权    248             股票期权
    //                          249             ETF期权
    //                          255             未分类
    //      10          其他
    //
//*************************************************************************************
****
    unsigned char                   m_security_type;                // 证券类型
    unsigned char                   m_sub_security_type;            // 证券子类型
    unsigned int                    m_tick_time;                    // 订单或成交时
间,格式如：HHMMSSsss（千分之一秒）HH=00-23, MM=00-59, SS=00-60(秒) ss=000-999(千分之一秒)
如 143025002 表示 14:30:25.002
    unsigned char                   m_tick_type;                    // 类型， A新增
订单， D删除订单， 删除订单， S产品状态订单， T成交
    unsigned long long              m_buy_num;                      // 买方订单号
    unsigned long long              m_sell_num;                     // 卖方订单号
    unsigned int                    m_price;                        // 价格，对产品
状态订单无意义       实际值需要除以 1000
    unsigned long long              m_quantity;                     // 数量（手），
对产品状态订单无意义 实际值需要除以 1000
    unsigned long long              m_trade_value;                  // 成交额，仅适
用于成交消息         实际值需要除以 100000
    char                            m_side_flag;                    // 买卖标志，若
为订单： B- 买单， S- 卖单；若为成交： B- 外盘， 主动买， S- 内盘， 主动卖 N- 未知 。
    unsigned char                   m_instrument_status;            // 当前品种交易
状态，仅适用于产品状态订单

//*************************************************************************************
*****
    // 0:其他     1:启动 2:开市集合竞价 3:连续自动撮合 4:停牌
    // 6:闭市，自动计算闭市价格  7:交易结束  8:产品未上市

//*************************************************************************************
```

```
*****
    unsigned char                   m_reserved[8];                  // 保留字段
};
```

1.1.28 sse_hpf_tick_merge 上交所逐笔合并通道行情

```
struct sse_hpf_tick_merge
{
    sse_hpf_tick_merge()
    {
        memset(this, 0, sizeof(sse_hpf_tick_merge));                // 初始化结构体
    }

    sse_hpf_head                    m_head;                         // 包头
    unsigned long long              m_tick_index;                   // 序号从1开始,
按通道连续
    unsigned int                    m_channel_num;                  // 通道号
    char                            m_symbol[SSE_SYMBOL_LEN];       // 标的名称
    //
**********************************************************************************
****
    //      证券类型                证券子类型
    //      0           指数        0               指数
    //                              255             未分类
    //      1           股票        0               主板
    //                              1
    //                              2               创业板
    //                              3               B股
    //                              4               H股
    //                              5               科创板
    //                              255             未分类
    //      2           基金        248             LOF基金
    //                              249             ETF基金
    //                              255             未分类
    //      3           债券        248             国债
    //                              249             可转债
    //                              255             未分类
    //      4           回购        248             国债回购
    //                              255             未分类
    //      9           期权        248             股票期权
    //                              249             ETF期权
    //                              255             未分类
    //      10          其他
```

```
    //
*****************************************************************************************
****
    unsigned char                   m_security_type;                // 证券类型
    unsigned char                   m_sub_security_type;            // 证券子类型
    unsigned int                    m_tick_time;                    // 订单或成交时
间,格式如：HHMMSSsss（千分之一秒）HH=00-23, MM=00-59, SS=00-60(秒) ss=000-999(千分之一秒)
如 143025002 表示 14:30:25.002
    unsigned char                   m_tick_type;                    // 类型， A新增
订单， D删除订单， 删除订单， S产品状态订单， T成交
    unsigned long long              m_buy_num;                      // 买方订单号
    unsigned long long              m_sell_num;                     // 卖方订单号
    unsigned int                    m_price;                        // 价格，对产品
状态订单无意义       实际值需要除以 1000
    unsigned long long              m_quantity;                     // 数量（手），
对产品状态订单无意义 实际值需要除以 1000
    unsigned long long              m_trade_value;
    // 对于新增委托,表示已成交的委托数量， 字段类型为 uInt64_1000, 实际值需要除以 1000;
    // 对于成交,表示成交金额 （ 元 ）， 字段类型为 uInt64_100000, 实际值需要除以 100000;
    // 其他无意义

    char                            m_side_flag;                    // 买卖标志，若
为订单： B- 买单， S- 卖单；若为成交： B- 外盘， 主动买， S- 内盘， 主动卖 N- 未知 。
    unsigned char                   m_instrument_status;            // 当前品种交易
状态，仅适用于产品状态订单

//*****************************************************************************************
*****
    // 0:其他     1:启动 2:开市集合竞价 3:连续自动撮合 4:停牌
    // 6:闭市，自动计算闭市价格  7:交易结束  8:产品未上市

//*****************************************************************************************
*****
    unsigned char                   m_reserved[4];                  // 保留字段
};
```

1.1.29 sse_hpf_etf 上交所 ETF 统计行情

```
struct sse_hpf_etf
{
    sse_hpf_etf()
    {
        memset(this, 0, sizeof(sse_hpf_etf));                   // 初始化结构体
```

```
    }
    sse_hpf_head                    m_head;                         // 包头
    unsigned int                    m_quote_update_time;            // 行情更新时间,
格式如：HHMMSSsss（千分之一秒）HH=00-23, MM=00-59, SS=00-60(秒) ss=000-999(千分之一秒) 如
143025002 表示 14:30:25.002
    char                            m_symbol[SSE_SYMBOL_LEN];       // 合约
    //
**********************************************************************************
****
    //      证券类型            证券子类型
    //      0           指数    0           指数
    //                          255         未分类
    //      1           股票    0           主板
    //                          1
    //                          2           创业板
    //                          3           B股
    //                          4           H股
    //                          5           科创板
    //                          255         未分类
    //      2           基金    248         LOF基金
    //                          249         ETF基金
    //                          255         未分类
    //      3           债券    248         国债
    //                          249         可转债
    //                          255         未分类
    //      4           回购    248         国债回购
    //                          255         未分类
    //      9           期权    248         股票期权
    //                          249         ETF期权
    //                          255         未分类
    //      10          其他
    //
**********************************************************************************
****
    unsigned char                   m_security_type;                // 证券
类型
    unsigned char                   m_sub_security_type;            // 证券
子类型
    unsigned int                    m_buy_number;                   // ETF
申购笔数，取值范围>=0
    unsigned long long              m_buy_amount;                   // ETF
申购数量，取值范围>=0
    unsigned long long              m_buy_money;                    // ETF
申购金额，取值范围>=0
```

```
    unsigned int                    m_sell_number;          // ETF
赎回笔数，取值范围>=0
    unsigned long long              m_sell_amount;          // ETF
赎回数量，取值范围>=0
    unsigned long long              m_sell_money;           // ETF
赎回金额，取值范围>=0
    unsigned char                   m_reserved_0[15];       // 保留
字段
};
```

1.1.30 sse_static_msg_header 上交所静态消息行情头

```
struct sse_static_msg_header
{
    unsigned int                m_seq_num;                  // 发包的消息序
号 从1开始
    unsigned char               m_reserved[4];              // 4个字节保留字
段
    unsigned char               m_msg_type;                 // 消息类型
    unsigned short              m_msg_body_len;             // 消息体长度
    bool                        m_batch_finish_flag;        // 当前静态文件
发送完成标志
};
```

1.1.31 sse_static_msg_body 上交所静态消息行情体

```
struct sse_static_msg_body
{
    unsigned char               m_exchange_id;              // 交易
所id
    char                        m_symbol[SSE_SYMBOL_LEN];   // 证券
代码
    unsigned int                m_send_time;                // 行情
发送时间，时分秒毫秒
    char                        m_static_file_date[DATE_LEN];   // 静态
文件日期，YYYYMMDD, 以'\0'结束
    //涨跌幅限制类型：
**********************************************************************************
***
    //  ‘ ’ (十进制数字为32)表示无定义
    //  ‘N’表示交易规则（2013修订版）3.4.13规定的有涨跌幅限制类型或者权证管理办法第22
条规定
    //  ‘R’表示交易规则（2013修订版）3.4.15和3.4.16规定的无涨跌幅限制类型
```

```
    //  ‘S’表示回购涨跌幅控制类型
    //  ‘F’表示基于参考价格的涨跌幅控制
    //  ‘P’表示IPO上市首日的涨跌幅控制类型
    //  ‘U’表示无任何价格涨跌幅控制类型

//*********************************************************************************************
    char                            m_price_limit_type;                     // 跌涨停限制类型
    double                          m_up_limit_price;                       // 涨停价
    double                          m_down_limit_price;                     // 跌停价
    unsigned long long int          m_bid_qty_unit;                         // 买数量单位
    unsigned long long int          m_ask_qty_unit;                         // 卖数量单位
    unsigned long long int          m_limit_upper_qty;                      // 限价申报数量上限
    unsigned long long int          m_limit_lower_qty;                      // 限价申报数量下限
    double                          m_price_changge_unit;                   // 申报最小变价单位
    unsigned long long int          m_market_upper_qty;                     // 市价申报数量上限
    unsigned long long int          m_market_lower_qty;                     // 市价申报数量下限
    char                            m_security_name[SSE_SECURITY_NAME_LEN];     // 证券名称，以'\0'结束
    //
*********************************************************************************************
    //      证券类型                证券子类型
    //      0           指数        0               指数
    //                              255             未分类
    //      1           股票        0               主板
    //                              1
    //                              2               创业板
    //                              3               B股
    //                              4               H股
    //                              5               科创板
    //                              255             未分类
    //      2           基金        248             LOF基金
    //                              249             ETF基金
```

```
    //                          255             未分类
    //      3           债券    248             国债
    //                          249             可转债
    //                          255             未分类
    //      4           回购    248             国债回购
    //                          255             未分类
    //      9           期权    248             股票期权
    //                          249             ETF期权
    //                          255             未分类
    //      10          其他
    //
*****************************************************************************************
****
    unsigned char                   m_ssecurity_type;                   // 证券
类型
    unsigned char                   m_sub_ssecurity_type;               // 证券
子类型
    char                            m_finance_target_mark;              // 融资
标的标志 ‘ ’ (十进制数字为32)表示无定义，‘T’表示是融资标的证券，‘F’表示不是融资标的
证券。
    char                            m_ssecurity_target_mark;            // 融券
标的标志 ‘ ’ (十进制数字为32)表示无定义，‘T’表示是融资标的证券，‘F’表示不是融资标的
证券。
    char                            m_product_status[SSE_PRODUCT_STATUS_LEN];  // 产品
状态，以’\0’结束
    char                            m_listing_date[DATE_LEN];           // 上市
日期，格式为YYYYMMDD，以’\0’结束
};
```

1.2 i_efh_sze_lev2 与 i_efh_sse_lev2 接口

i_efh_sze_lev2 与 i_efh_sse_lev2 接口提接收行情数据功能，具体功能见以下方法。

1.2.1 create_efh_sze_lev2_api_function

函数	extern "C" i_efh_sze_lev2_api * create_efh_sze_lev2_api_function()
参数	NA
功能	创建 e i_efh_sze_lev2_api 实例
返回	i_efh_sze_lev2_api 实例的指针
回调函数	NA

1.2.2 destroy_efh_sze_lev2_api_function

函数	extern "C" void destroy_efh_sze_lev2_api_function(i_efh_sze_lev2_api * p_efh_sze_lev2_api)
参数	p_efh_sze_lev2_api:要销毁的实例
功能	销毁 i_efh_sze_lev2_api 实例
返回	NA
回调函数	NA

1.2.3 init_sze

函数	virtual bool init_sze(sock_udp_param* p_udp_param, int num, efh_sze_lev2_api_event* p_event, efh_sze_lev2_api_depend* p_depend = NULL) = 0;
参数	p_udp_param:通道配置数组的地址，num 数组元素的个数，p_depend:派生自回调接口类的实例,p_event:派生自回调接口类的实例
功能	初始化运行环境，只有调用后，接口才能开始工作
返回	true 表示成功初始化，false 表示初始化失败
回调函数	NA

1.2.4 start_sze

函数	virtual bool start_sze(enum_efh_nic_type nic_type) = 0;
参数	net_type: 使用的网卡类型
功能	接口开始工作，需要在 init_sze 成功后才能调用
返回	true 表示成功初始化，false 表示初始化失败
回调函数	NA

1.2.5 stop_sze

函数	virtual void stop_sze() = 0;
参数	NA
功能	停止接口
返回	NA
回调函数	NA

1.2.6 close_sze

函数	virtual void close_sze() = 0;
参数	NA
功能	关闭接口，释放资源
返回	NA

回调函数	NA

1.2.7 efh_sze_lev2_error (需继承 efh_sze_lev2_api_depend)

函数	virtual void efh_sze_lev2_error(const char* msg, int len) {}
参数	msg 为需要记录的日志内容; len 为内容的长度
功能	用于记录 i_efh_sze_lev2_api 中出现的错误信息
返回	NA
回调函数	NA

1.2.8 efh_sze_lev2_debug (需继承 efh_sze_lev2_api_depend)

函数	virtual void efh_sze_lev2_debug(const char* msg, int len) {}
参数	msg 为需要记录的日志内容; len 为内容的长度
功能	用于记录 i-efh_sze_lev2_api 中出现的调试信息
返回	NA
回调函数	NA

1.2.9 efh_sze_lev2_info (需继承 efh_sze_lev2_api_depend)

函数	virtual void efh_sze_lev2_info(const char* msg, int len) {}
参数	msg 为需要记录的日志内容; len 为内容的长度
功能	用于记录 efh_sze_lev2_api 中出现的相关信息
返回	NA
回调函数	NA

1.2.10 on_report_efh_sze_lev2_idx (需继承 efh_sze_lev2_api_event)

函数	virtual void on_report_efh_sze_lev2_idx(sze_hpf_index_pkt * p_idx) = 0;
参数	p_idx 指数行情数据
功能	用于回显给用户深交所指数行情数据
返回	NA
回调函数	NA

1.2.11 on_report_efh_sze_lev2_snap (需继承 efh_sze_lev2_api_event)

函数	virtual void on_report_efh_sze_lev2_snap(sze_hpf_lev2* p_snap) = 0;
参数	p_snap lev2 快照行情数据
功能	用于回显给用户深交所 lev2 快照行情数据
返回	NA
回调函数	NA

1.2.12 on_report_efh_sze_lev2_tick（需继承 efh_sze_lev2_api_event）

函数	virtual void on_report_efh_sze_lev2_tick(int msg_type, sze_hpf_order* p_order, sze_hpf_exe* p_exe) = 0;
参数	msg_type:消息类型, p_order:订单结构, p_exe:成交类型
功能	用于回显给用户深交所逐笔行情数据
返回	NA
回调函数	NA

1.2.13 on_report_efh_sze_close_px（需继承 efh_sze_lev2_api_event）

函数	virtual void on_report_efh_sze_lev2_after_close(sze_hpf_after_close* p_after_close) = 0;
参数	p_after_close 盘后定价行情数据
功能	用于回显给用户深交所盘后定价行情数据
返回	NA
回调函数	NA

1.2.14 on_report_efh_sze_lev2_tree（需继承 efh_sze_lev2_api_event）

函数	virtual void on_report_efh_sze_lev2_tree(sze_hpf_tree_pkt* p_tree) = 0;
参数	p_tree 建树行情数据
功能	用于回显给用户深交所建树行情数据
返回	NA
回调函数	NA

1.2.15 on_report_efh_sze_lev2_ibr_tree（需继承 efh_sze_lev2_api_event）

函数	virtual void on_report_efh_sze_lev2_ibr_tree(sze_hpf_ibr_tree* p_ibr_tree) = 0;
参数	p_ibr_tree ibr 建树行情数据
功能	用于回显给用户深交所 ibr 建树行情数据
返回	NA
回调函数	NA

1.2.16 on_report_efh_sze_lev2_turnover（需继承 efh_sze_lev2_api_event）

函数	virtual void on_report_efh_sze_lev2_turnover(sze_hpf_turnover* p_turnover) = 0;
参数	p_turnover 深交所成交量行情数据
功能	用于回显给用户深交所成交量行情数据

返回	NA
回调函数	NA

1.2.17 on_report_efh_sze_lev2_bond_snap （需继承 efh_sze_lev2_api_event）

函数	virtual void on_report_efh_sze_lev2_bond_snap(sze_hpf_bond_snap* p_snap) = 0;
参数	p_snap lev2 债券快照行情数据
功能	用于回显给用户深交所 lev2 债券快照行情数据
返回	NA
回调函数	NA

1.2.18 on_report_efh_sze_lev2_bond_tick （需继承 efh_sze_lev2_api_event）

函数	virtual void on_report_efh_sze_lev2_bond_tick(int msg_type, sze_hpf_bond_order* p_order, sze_hpf_bond_exe* p_exe) = 0;
参数	msg_type:消息类型, p_order:债券订单结构, p_exe:债券成交类型
功能	用于回显给用户深交所债券逐笔行情数据
返回	NA
回调函数	NA

1.2.19 create_efh_sse_lev2_api_function

函数	extern "C" i_efh_sse_lev2_api * create_efh_sse_lev2_api_function()
参数	NA
功能	创建 e i_efh_sse_lev2_api 实例
返回	i_efh_sse_lev2_api 实例的指针
回调函数	NA

1.2.20 destroy_efh_sse_lev2_api_function

函数	extern "C" void destroy_efh_sse_lev2_api_function(i_efh_sse_lev2_api * p_efh_sse_lev2_api)
参数	p_efh_sse_lev2_api:要销毁的实例
功能	销毁 i_efh_sse_lev2_api 实例
返回	NA
回调函数	NA

1.2.21 init_sse

函数	virtual bool init_sse(sock_udp_param* p_udp_param, int num, efh_sse_lev2_api_event* p_event, efh_sse_lev2_api_depend* p_depend = NULL) = 0;
参数	p_udp_param:通道配置数组的地址，num 数组元素的个数，p_depend:派生自回调接口类的实例; p_event:派生自回调接口类的实例
功能	初始化运行环境,只有调用后,接口才能开始工作
返回	true 表示成功初始化, false 表示初始化失败
回调函数	NA

1.2.22 start_sse

函数	virtual bool start_sse(enum_efh_nic_type nic_type) = 0;
参数	net_type: 使用的网卡类型
功能	接口开始工作，需要在 init_sse 成功后才能调用
返回	true 表示成功初始化, false 表示初始化失败
回调函数	NA

1.2.23 stop_sse

函数	virtual void stop_sse() = 0;
参数	NA
功能	停止接口
返回	NA
回调函数	NA

1.2.24 close_sse

函数	virtual void close_sse() = 0;
参数	NA
功能	关闭接口，释放资源
返回	NA
回调函数	NA

1.2.25 get_static_info(需要 i_efh_sse_lev2_api)

函数	virtual int get_static_info(const char* symbol ,sse_static_msg_body& static_info) = 0;
参数	symbol 合约号,static_info 接收静态消息的结构体
功能	获取上交所静态文件信息
返回	正常 SSE_STATIC_INFO_OK (0)

	合约格式不正确 SSE_STATIC_INFO_SYMBOL_IS_INCORRECT (-1) 未找到合约的静态信息 SSE_STATIC_INFO_NOT_FOUND_SYMBOL (-2) 找到合约信息为空值 SSE_STATIC_INFO_FOUND_SYMBOL_BUT_NO_VALUE (-3)
回调函数	NA

1.2.26 efh_sse_lev2_error (需继承 efh_sse_lev2_api_depend)

函数	virtual void efh_sse_lev2_error(const char* msg, int len) {}
参数	msg 为需要记录的日志内容; len 为内容的长度
功能	用于记录 i_efh_sse_lev2_api 中出现的错误信息
返回	NA
回调函数	NA

1.2.27 efh_sse_lev2_debug (需继承 efh_sse_lev2_api_depend)

函数	virtual void efh_sse_lev2_debug(const char* msg, int len) {}
参数	msg 为需要记录的日志内容; len 为内容的长度
功能	用于记录 i-efh_sse_lev2_api 中出现的调试信息
返回	NA
回调函数	NA

1.2.28 efh_sse_lev2_info (需继承 efh_sse_lev2_api_depend)

函数	virtual void efh_sse_lev2_info(const char* msg, int len) {}
参数	msg 为需要记录的日志内容; len 为内容的长度
功能	用于记录 efh_sse_lev2_api 中出现的相关信息
返回	NA
回调函数	NA

1.2.29 on_report_efh_sse_lev2_idx (需继承 efh_sse_lev2_api_event)

函数	virtual void on_report_efh_sse_lev2_idx(sse_hpf_index_pkt * p_idx) = 0;
参数	p_idx 指数行情数据
功能	用于回显给用户上交所指数行情数据
返回	NA
回调函数	NA

1.2.30 on_report_efh_sse_lev2_snap (需继承 efh_sse_lev2_api_event)

函数	virtual void on_report_efh_sse_lev2_snap(sse_hpf_lev2* p_snap) = 0;
参数	p_snap: lev2 快照数据结构
功能	用于回显给用户上交所盘后定价行情数据

返回	NA
回调函数	NA

1.2.31 on_report_efh_sse_option（需继承 efh_sse_lev2_api_even）

函数	virtual void on_report_efh_sse_lev2_option(sse_hpf_stock_option* p_opt) = 0;
参数	p_option :上交所期权行情数据
功能	用于回显给用户上交所期权行情数据
返回	NA
回调函数	NA

1.2.32 on_report_efh_sse_lev2_tick（需继承 efh_sse_lev2_api_event）

函数	virtual void on_report_efh_sse_lev2_tick(int msg_type, sse_hpf_order* p_order, sse_hpf_exe* p_exe) = 0;
参数	msg_type:消息类型, p_order:订单结构, p_exe:成交类型
功能	用于回显给用户上交所逐笔行情数据
返回	NA
回调函数	NA

1.2.33 on_report_efh_sse_lev2_tree（需继承 efh_sse_lev2_api_event）

函数	virtual void on_report_efh_sse_lev2_tree(sse_hpf_tree* p_tree) = 0;
参数	p_tree :上交所建树行情数据
功能	用于回显给用户上交所建树行情数据
返回	NA
回调函数	NA

1.2.34 on_report_efh_sse_lev2_bond_snap（需继承 efh_sse_lev2_api_even）

函数	virtual void on_report_efh_sse_lev2_bond_snap(sse_hpf_bond_snap* p_bond) = 0;
参数	p_bond :上交所债券快照行情数据
功能	用于回显给用户上交所债券快照行情数据
返回	NA
回调函数	NA

1.2.35 on_report_efh_sse_lev2_bond_tick（需继承 efh_sse_lev2_api_even）

函数	virtual void on_report_efh_sse_lev2_bond_tick(sse_hpf_bond_tick* p_tick) = 0;
参数	p_tick :上交所债券逐笔行情数据
功能	用于回显给用户上交所债券逐笔行情数据
返回	NA
回调函数	NA

第二章 接口字段说明

盛立证券类型				代码段（前 3 位）	
第 1 位	**证券类型**	**第 2 位**	**证券子类型**	**深交所**	**上交所**
0	指数	0	指数	395，399，98*	000
		255	未分类		
1	股票	0	主板	000-004	600，601，603，605
		1			
		2	创业板	300-309	
		3	B 股	200-209	900
		4	H 股		
		5	科创板		688，689
		255	未分类		
2	基金	248	LOF 基金	160-169，180	501，502，506，508
		249	ETF 基金	159，588000-588499	510-518，560-563，588
		255	未分类	184	500，505，550
3	债券	248	国债	100-107	009，010，018，019，020
		249	可转债	120-129	110，111，113，118，132
		255	未分类	108，109，11*，137，138，139，149，179，190，191	120-131，135-199
4	回购	248	国债回购	1318**	204
		255	未分类		
5	权证				
6	期货				
7	外汇				
8	利率				

9	期权	248	股票期权	92000001-99999999	20000001-30000000
		249	ETF 期权	90000001-91999999	10000001-20000000
		255	未分类		
10	其它				

盛立证券类型表_v1.7

2.1 深交所字段说明

Level-2 快照集合竞价特殊值处理表

行情条目类别	特殊值（原值）	转换后的值
Xe，xf, xg	MDEntryPx=999999999999900	999999999

深交所快照行情数据类别列表

MDStreamID	说明	消息类型
010	现货（股票，基金，债券等）集中竞价交易快照行情	300111
020	质押式回购交易快照行情	
030	债券分销快照行情	
040	期权集中竞价交易快照行情	
060	以收盘价交易的盘后定价大宗交易快照行情	300611
061	以成交量加权平均价交易的盘后定价大宗交易快照行情	
370	盘后定价交易快照行情	303711
630	港股实时行情	306311
900	指数快照行情	309011
910	成交量统计指标快照行情	309111
920	国证指数快照行情	309011
410	债券现券交易快照行情	300211

深交所逐笔委托行情数据类别列表

MDStreamID	说明	消息类型
011	现货（股票，基金，可转债等）集中竞价交易逐笔行情	300192
041	期权集中竞价交易逐笔行情	

051	协议交易逐笔意向行情	300592
052	协议交易逐笔定价行情	
071	转融通证券出借逐笔行情	300792
021	债券通用质押式回购匹配成交逐笔行情	300292
411	债券现券交易匹配成交逐笔行情	
413	债券现券交易点击成交逐笔行情	300392
415	债券现券交易意向申报逐笔行情	
416	债券现券交易竞买成交逐笔行情	300492
417	债券现券交易匹配成交大额逐笔行情	300392

深交所逐笔成交行情数据类别列表

MDStreamID	**说明**	**消息类型**
011	现货（股票，基金，可转债等）集中竞价交易逐笔行情	300191
041	期权集中竞价交易逐笔行情	
051	协议交易逐笔意向行情	300591
052	协议交易逐笔定价行情	
071	转融通证券出借逐笔行情	300791
021	债券通用质押式回购匹配成交逐笔行情	300291
411	债券现券交易匹配成交逐笔行情	
412	债券现券交易协商成交逐笔行情	300391
413	债券现券交易点击成交逐笔行情	
414	债券现券交易询价成交逐笔行情	
416	债券现券交易竞买成交逐笔行情	300491
417	债券现券交易匹配成交大额逐笔行情	300391

TradingPhaseCode 编码表

Bit 位	**盛立值**	**交易所值**	**描述**
Bit7-Bit4	0	‘S’	表示启动（开市前）
	1	‘O’	表示开盘集合竞价
	2	‘T’	表示连续交易
	3	‘B’	表示休市
	4	‘C’	表示收盘集合竞价
	5	‘E’	表示已闭市
	6	‘H’	表示临时停牌
	7	‘A’	表示盘后交易
	8	‘V’	表示波动性中断
	15	other	表示无意义

Bit3	0 1 0	‘0’ ‘1’ other	‘0’表示此正常状态， ‘1’表示此全天停牌 ‘ ’表示无意义，当成此产品可正常交易处理
Bit2	0		预留位，恒为 0
Bit1	0		预留位，恒为 0
Bit0	0		预留位，恒为 0

2.2 上交所字段说明

TradingPhaseCode 编码表

Bit Num	盛立值	上交所值	描述
Bit7-Bit4	0	‘S’	启动（开市前）
	1	‘C’	开盘集合竞价
	2	‘T’	连续竞价
	3	‘E’	闭市
	4	‘P’	合约停牌
	5	‘M’	可恢复交易的熔断时段（盘中集合竞价）
	6	‘N’	不可恢复交易的熔断时段（暂停交易至闭市）
	7	‘U’	收盘集合竞价阶段
	8	‘V’	波动性中断（适用于股票期权）
	…		预留位
	15	other	无意义
Bit3	0	‘0’	此合约不可正常交易
	1	‘1’	此合约可正常交易
		‘’	无意义
Bit2	0	‘0’	未上市
	1	‘1’	已上市
		‘’	无意义
Bit1	0	‘0’	此合约在当前时段不接受订单申报
	1	‘1’	此合约在当前时段接受订单申报
		‘’	无意义
Bit0	0		预留位，恒为 0

证券交易状态表

时段	时段描述	盛立对应值
	其它	0
START	启动	1
OCALL	开市集合竞价	2
TRADE	连续自动撮合	3
SUSP	停牌	4

CCALL	收盘集合竞价	5
CLOSE	闭市，自动计算闭市价格	6
ENDTR	交易结束	7
ADD	产品未上市	8

第三章 接口应用实例

3.1 功能说明

上交所支持接收八种协议：指数/逐笔行情(逐笔委托、逐笔成交)/lev2 快照/期权/建树/债券快照/债券逐笔行情，以及一个额外的静态信息查询功能。

深交所支持接收十一种协议：指数/逐笔行情(逐笔委托、逐笔成交)/盘后定价/lev2 快照/建树/ibr 建树/成交量统计/债券快照/债券逐笔行情(逐笔委托、逐笔成交)。

可以选择是否使用 cache 缓存功能，并选择缓存大小。

网卡有四种模式：普通模式，efvi 模式，x25 模式, win 高速模式。其中,efvi 模式,x25 模式仅在 Linux 系统下可用, win 高速模式仅在 window 系统下可用。

使用某通道时要填写是否使用此通道，用于接收的 cpu 序号，组播 IP 地址（通道 IP 地址），组播端口号（通道端口号），目标 IP 地址，目标端口号，网卡名称。**cpu 序号通道使用 cat /sys/class/net/网卡/device/numa_node && lspci 确认那些 CPU ID 与指定网卡在同一个 numa node 上,特别的,对于 win 高速模式,CPU ID 应当为非零正整数。**目标 IP 地址在使用普通模式使用本地 IP，EFVI 模式不使用，x25 使用发送方 IP，目标端口号在普通模式中使用本地端口号，EFVI 模式中不使用，x25 模式中使用发送方端口号。

对于 windows 系统下的网卡名称填写，形如“0000:af:00.0”类似定义；生成规则：

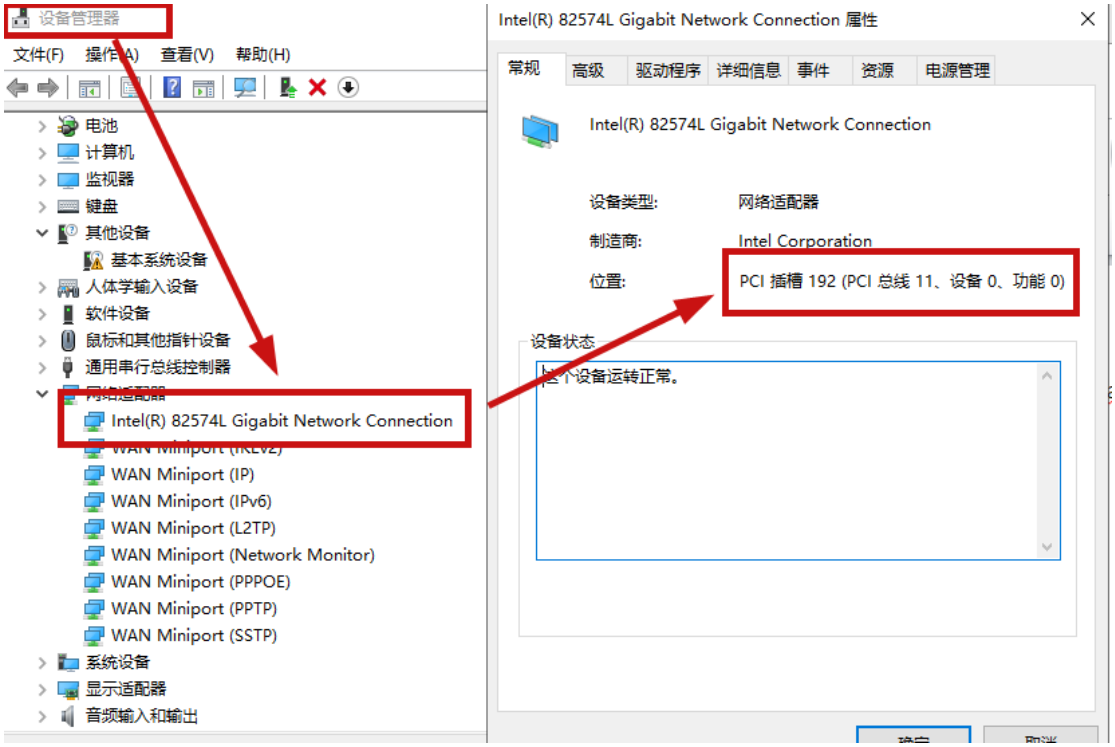
A、组成： 网卡名称由四部分组成：domain、bus、device、function；

B、domain：恒等于 0000；

C、bus：见例图右侧部分，值为 总线(bus) 部分数值的 16 进制（例图中 192 的 16 进制为 cd）；

D、device：见例图右侧部分，值为 设备(device) 部分数值的 16 进制；

E、function：见例图右侧部分，值为 功能(function)部分数值的 16 进制；



3.2 配置文件

```
#配置日志需要比较的 key,'='左边为 record 日志 key，右边是 tostring 日志 key

#k-v 在解析的时候，均以第一个'='为标志分割

#EFVI 模式不使用目标 ip 地址，目标端口号

#以下的例子是使用 EFVi 模式



[EFH_QUOTE_TYPE]

enable_sze=0    #使用深交所通道

enable_sse=1    #使用上交所通道



[REPORT]
```

```
#是否退出时才写文件，1：是，打印文本文件，会过滤合约；0：不是，打印二进制文件，不过滤合约
report_when_quit=0
#需要的合约，以","分割
symbol="002835"

[NIC_TYPE]
nic_type=1              #网卡模式，0：普通模式，1：efvi 模式，2：x25 模式, 3: win 高速模式(x710网卡), 4: win 高速模式(solarflare 网卡)

#深交所
[EFH_SZE_LEV2_SNAP]
enable=1                # 是否使用此通道，1:使用，0 为不使用
cpu_id=1                # 用于接收的 cpu 序号
multicast_ip="233.57.1.100"    # 组播 IP 地址（通道 IP 地址）
multicast_port=37100          # 组播端口号（通道端口号）
data_ip="4.4.4.63"            # 普通模式下为本机 IP 地址, win 高速模式下为发送方 IP 地址,(EFVI 模式与 x25 模式不使用)
data_port=0                 # 普通模式下为本机端口号, win 高速模式下为发送方端口号，(EFVI 模式与 x25 模式不使用)
eth_name="eth7"              # 网卡名称,对于 Windows 平台, 形如 0000:af:00.0
cache_size=256              # cache 大小，单位为 M
```

```
force_normal_socket=0          # 强制使用普通的 socket 接收(低性能模式), 1:使用低性能接
收, 0:禁用



[EFH_SZE_LEV2_IDX]
enable=1
cpu_id=2
multicast_ip="233.57.1.102"
multicast_port=37102
data_ip="4.4.4.63"
data_port=0
eth_name="eth7"
cache_size=256
force_normal_socket=0


[EFH_SZE_LEV2_TICK]
enable=1
cpu_id = 5
multicast_ip="233.57.1.101"
multicast_port=37101
data_ip="4.4.4.63"
data_port=0
```

```
eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SZE_LEV2_AFTER_CLOSE]

enable=1

cpu_id=3

multicast_ip="233.57.1.103"

multicast_port=37103

data_ip="4.4.4.63"

data_port=0

eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SZE_LEV2_TREE]

enable=1

cpu_id = 4

multicast_ip="233.57.1.104"

multicast_port=37104

data_ip="4.4.4.63"

data_port=0
```

```
eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SZE_LEV2_IBR_TREE]

enable=1

cpu_id = 4

multicast_ip="233.57.1.104"

multicast_port=37104

data_ip="4.4.4.63"

data_port=0

eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SZE_LEV2_TURNOVER]

enable=1

cpu_id = 4

multicast_ip="233.57.1.104"

multicast_port=37104

data_ip="4.4.4.63"

data_port=0
```

```
eth_name="eth7"

cache_size=256

force_normal_socket=0

#上交所

[EFH_SSE_LEV2_IDX]

enable=1

cpu_id=7

multicast_ip="233.57.1.102"

multicast_port=37102

data_ip="4.4.4.63"

data_port=0

eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SSE_LEV2_TICK]

enable=1

cpu_id = 8

multicast_ip="233.57.1.103"

multicast_port=37103

data_ip="4.4.4.63"

data_port=0
```

```
eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SSE_LEV2_OPTION]

enable=1

cpu_id = 9

multicast_ip="233.57.1.104"

multicast_port = 37104

data_ip="4.4.4.63"

data_port=0

eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SSE_LEV2_SNAP]

enable=1

cpu_id=10

multicast_ip="233.57.1.105"

multicast_port=37105

data_ip="4.4.4.63"

data_port=0
```

```
eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SSE_LEV2_TREE]

enable=1

cpu_id=10

multicast_ip="233.57.1.105"

multicast_port=37105

data_ip="4.4.4.63"

data_port=0

eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SSE_LEV2_BOND_SNAP]

enable=1

cpu_id=12

multicast_ip="233.57.1.106"

multicast_port=37106

data_ip="4.4.4.63"

data_port=0
```

```
eth_name="eth7"

cache_size=256

force_normal_socket=0



[EFH_SSE_LEV2_BOND_TICK]

enable=1

cpu_id=14

multicast_ip="233.57.1.107"

multicast_port=37107

data_ip="4.4.4.63"

data_port=0

eth_name="eth7"

cache_size=256

force_normal_socket=0



#上交所静态文件信息

[EFH_SSE_STATIC_INFO]

enable=1

cpu_id=1

multicast_ip="224.0.5.5"

multicast_port=8844

data_ip="192.168.18.177"
```

```
data_port=8844

eth_name="enp179s0f0"

cache_size=256

force_normal_socket=0
```

3.3 使用

1. 编译: make clean && make
2. 运行: ./sl_efh_lev2_recv_demo
3. 程序退出: 输入 quit 退出