Backtester C/C++ Code Listings

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 ats Namespace Reference

Namespaces

- date_time
- detail
- order_book_detail
- sim

Classes

- struct bar
- class bracket_order
- class csv_reader
- class csv_single_message_reader
- class data_feed
- class double_key_lookup
- class event_handler
- class event_handler< ReturnT(Args...)>
- class exchange_message_reader_base
- class exchange_order_book
- struct execution
- class execution_engine
- class historical_data_feed
- class indicator
- struct instrument_message
- struct instrument_message_packet
- class l2_message_reader
- class level2_exchange_message_reader
- class level2_execution_engine
- class level2_historical_data_feed
- struct level2_message
- class limit_order
- class limit_order_container
- class limit_order_container1
- class market_order

- struct message
- class message_reader
- struct metainfo
- class multievent_handler
- class multievent_memfunc_handler
- class order
- class order_book
- class order_book_manager
- class order_container
- class order_container1
- class order_processor
- struct order_status_cancelled_message
- struct order_status_filled_message
- struct order_status_message
- struct order_status_new_message
- struct order_status_partially_filled_message
- struct order_status_pending_new_message
- struct order_status_rejected_message
- struct performance_item
- struct performance_item_basic
- struct pnl_item
- class portfolio_base
- class position
- struct position_offset_message
- class recursive_timer
- class report_engine
- class security_base
- class security_container
- class single_message_reader
- class stop_order
- class stream_reader
- struct symbol_key
- struct trade_message

Typedefs

- typedef std::function< void(const ats::order_status_message &)> order_status_handler
- typedef std::function< void(const ats::position &)> position_change_handler
- typedef std::function< void(const ats::level2_message_packet &)> order_book_changed_handler
- typedef ats::instrument_message_packet< level2_message > level2_message_packet
- typedef ats::instrument_message_packet< ats::trade_message > trade_message_packet
- typedef std::function< void(const ats::order_status_message &)> on_order_status_changed_handler
- typedef std::function< void(const ats::level2_message_packet &)> on_order_book_changed_handler
- typedef std::function< void(const ats::trade_message &)> on_trade_handler
- using price_t = int
- using timestamp_t = ats::date_time::date_time
- using orderid_t = uint64_t

Enumerations

- enum trading_session_id { trading_session_id::PreOpening = 0, trading_session_id::Opening, trading_session_id::Continuous }
- enum update_action { update_action::New = 0, update_action::Change, update_action::Delete, update_action::Overlay }
- enum entry_type { entry_type::Bid = 0, entry_type::Ask, entry_type::Trade }
- enum aggressor_side { aggressor_side::Undefined = 0, aggressor_side::Buy, aggressor_side::Sell }
- enum market_state { market_state::PreOpening, market_state::Opening, market_state::ContinuousTrading }
- enum order_side { order_side::Buy, order_side::Sell, order_side::BuyCover, order_side::SellShort }
- enum order_time_in_force {
  order_time_in_force::Day, order_time_in_force::GTC, order_time_in_force::IOC, order_time_in_force::AtTheOpening,
  order_time_in_force::AtTheClose, order_time_in_force::FUNARI, order_time_in_force::FOK, order_time_in_force::GTX,
  order_time_in_force::Date }
- enum order_status {
  order_status::New = 0, order_status::PartiallyFilled, order_status::Filled, order_status::DoneForDay,
  order_status::Canceled, order_status::Replaced, order_status::PendingCancel, order_status::Rejected,
  order_status::Suspended, order_status::PendingNew, order_status::Calculated, order_status::Expired,
  order_status::AcceptedForBidding, order_status::PendingReplace }
- enum order_type { order_type::Market = 1, order_type::Limit, order_type::Stop, order_type::StopLimit }
- enum execution_mode { execution_mode::BackTesting, execution_mode::WarmUp, execution_mode::RealTime }
- enum subscription {
  subscription::Level2, subscription::Level1, subscription::TimeAndSales, subscription::Bar,
  subscription::Daily, subscription::Custom }

Functions

- static bool parse_fix_msg (const char *fix_msg, ats::level2_message_packet &result, const FIX::DataDictionary &dictionary, const std::unordered_set< std::string > &symbols)
- template<size_t num_cols>
  static size_t tokenize (const std::string &line, std::array< std::string, num_cols > &result, char sep=',')
- static void to_spread (const std::string &x_file, const std::string &y_file, const std::string &out_file, const std::string &header="Symbol; Timestamp; BidPrc; AskPrc")
- static void fix_to_csv (const std::string &fix_file, const std::string &csv_file, const std::string &fix_specs_xml, const std::unordered_set< std::string > &symbols, bool print_seq_num=false, const std::string &eop_str="EOP")

5.1.1 Typedef Documentation

5.1.1.1 level2_message_packet

```
typedef ats::instrument_message_packet<level2_message> ats::level2_message_packet
```

5.1.1.2 on_order_book_changed_handler

```
typedef std::function<void(const ats::level2_message_packet&)> ats::on_order_book_changed_handler
```

5.1.1.3 on_order_status_changed_handler

```
typedef std::function<void(const ats::order_status_message&)> ats::on_order_status_changed_handler
```

5.1.1.4 on_trade_handler

```
typedef std::function<void(const ats::trade_message&)> ats::on_trade_handler
```

5.1.1.5 order_book_changed_handler

```
typedef std::function<void(const ats::level2_message_packet&)> ats::order_book_changed_handler
```

5.1.1.6 order_status_handler

```
typedef std::function< void(const ats::order_status_message &)> ats::order_status_handler
```

5.1.1.7 orderid_t

```
using ats::orderid_t = typedef uint64_t
```

5.1.1.8 position_change_handler

```
typedef std::function< void(const ats::position &)> ats::position_change_handler
```

5.1.1.9 price_t

```
using ats::price_t = typedef int
```

5.1.1.10 timestamp_t

```
using ats::timestamp_t = typedef ats::date_time::date_time
```

5.1.1.11 trade_message_packet

```
typedef ats::instrument_message_packet<ats::trade_message> ats::trade_message_packet
```

5.1.2 Enumeration Type Documentation

5.1.2.1 aggressor_side

```
enum ats::aggressor_side [strong]
```

Enumerator

Undefined	
Buy	
Sell	

```
{ Undefined = 0, Buy, Sell };
```

5.1.2.2 entry_type

```
enum ats::entry_type [strong]
```

Enumerator

Bid	
Ask	
Trade	

```
{ Bid = 0, Ask, Trade };
```

5.1.2.3 execution_mode

enum ats::execution_mode [strong]

Enumerator

BackTesting	
WarmUp	
RealTime	

```
    {
        BackTesting,
        WarmUp,
        RealTime
    };
```

5.1.2.4 market_state

enum ats::market_state [strong]

Enumerator

PreOpening	
Opening	
ContinuousTrading	

```
    {
        PreOpening, Opening, ContinuousTrading
    };
```

5.1.2.5 order_side

enum ats::order_side [strong]

Enumerator

Buy	
Sell	
BuyCover	
SellShort	

```
    {
        Buy,
        Sell,
         BuyCover,
         SellShort
     };
```

5.1.2.6 order_status

enum ats::order_status [strong]

Enumerator

New	
PartiallyFilled	
Filled	
DoneForDay	
Canceled	
Replaced	
PendingCancel	
Rejected	
Suspended	
PendingNew	
Calculated	
Expired	
AcceptedForBidding	
PendingReplace	

```
     {
         New = 0,                // outstanding order with no executions
         PartiallyFilled,        // partially filled and is still working; sent on every order execution
         Filled,                 // completely filled (final order status)
         DoneForDay,
         Canceled,               // cancelled (final order status)
         Replaced,
         PendingCancel,
         Rejected,               // rejected for some reason (final order status)
         Suspended,
         PendingNew,             // received by sell-side's system but not yet accepted for execution
         Calculated,
         Expired,
         AcceptedForBidding,
         PendingReplace
     };
```

5.1.2.7 order_time_in_force

enum ats::order_time_in_force [strong]

Enumerator

Day	
GTC	
IOC	
AtTheOpening	
AtTheClose	
FUNARI	
FOK	
GTX	
Date	

```
{
    Day,
    GTC,
    IOC,
    AtTheOpening,
    AtTheClose,
    FUNARI,
    FOK,
    GTX,
    Date
};
```

5.1.2.8 order_type

enum ats::order_type [strong]

Enumerator

Market	
Limit	
Stop	
StopLimit	

```
{
    Market = 1,
    Limit,
    Stop,
    StopLimit
};
```

5.1.2.9 subscription

enum ats::subscription [strong]

Enumerator

Level2	

Enumerator

Level1	
TimeAndSales	
Bar	
Daily	
Custom	

```
    {
        Level2,
        Level1,
        TimeAndSales,
        Bar,
        Daily,
        Custom
    };
```

5.1.2.10 trading_session_id

enum ats::trading_session_id [strong]

Enumerator

PreOpening	
Opening	
Continuous	

```
    {
        PreOpening = 0, Opening, Continuous
    };
```

5.1.2.11 update_action

enum ats::update_action [strong]

Enumerator

New	
Change	
Delete	
Overlay	

```
{ New = 0, Change, Delete, Overlay };
```

5.1.3 Function Documentation

5.1.3.1 fix_to_csv()

```
static void ats::fix_to_csv (
            const std::string & fix_file,
            const std::string & csv_file,
            const std::string & fix_specs_xml,
            const std::unordered_set< std::string > & symbols,
            bool print_seq_num = false,
            const std::string & eop_str = "EOP" )  [static]
```

References Ask, Bid, Change, Delete, ats::instrument_message::exchange, ats::instrument_message_packet< MessageT >::messages, New, parse_fix_msg(), and Trade.

```
{
    std::ifstream fix(fix_file);
    std::ofstream csv(csv_file);

    FIX::DataDictionary dictionary(fix_specs_xml);

    auto entry_type_to_char = [](ats::entry_type type)
    {
        switch (type)
        {
        case ats::entry_type::Bid:
            return 'B';
        case ats::entry_type::Ask:
            return 'A';
        case ats::entry_type::Trade:
            return 'T';
        default:
            throw std::invalid_argument("Unknown entry type");
        }
    };

    auto update_action_to_char = [](ats::update_action action)
    {
        switch (action)
        {
        case ats::update_action::New:
            return 'N';
        case ats::update_action::Change:
            return 'C';
        case ats::update_action::Delete:
            return 'D';
        default:
            throw std::invalid_argument("Unknown update action");
        }
    };

    std::string fix_msg;
    ats::level2_message_packet msg;
    while (std::getline(fix, fix_msg))
    {
        if (ats::parse_fix_msg(fix_msg.c_str(), msg, dictionary, symbols) && !msg.messages.empty())
        {
            for (const auto& m : msg.messages)
            {
                csv << m.exchange << ',' << m.symbol << ',';
                if (print_seq_num)
                    csv << m.seq_number << ',';
                csv << m.time.to_string("%Y%m%d %H%M%S.%f") << ',' << update_action_to_char(m.update_action) << ','
                    << entry_type_to_char(m.entry_type) << ',' << m.price << ',' << m.quantity << ','
                    << m.order_count << ',' << m.level << '\n';
            }
            csv << eop_str;
        }
    }

    csv.close();
    fix.close();
}
```

5.1.3.2 parse_fix_msg()

```
static bool ats::parse_fix_msg (
            const char * fix_msg,
            ats::level2_message_packet & result,
            const FIX::DataDictionary & dictionary,
            const std::unordered_set< std::string > & symbols )  [static]
```

References ats::level2_message::aggressor_side, Ask, Bid, Change, Delete, ats::level2_message::entry_type, ats::instrument_message::exchange, ats::level2_message::level, ats::instrument_message_packet< MessageT >::messages, New, ats::level2_message::order_count, ats::date_time::date_time::parse(), ats::level2_message::price, ats::level2_message::quantity, ats::level2_message::seq_number, ats::instrument_message::symbol, ats::message::time, Trade, and ats::level2_message::update_action.

Referenced by fix_to_csv().

```
    {
        result.messages.clear();

        FIX50SP2::MarketDataIncrementalRefresh msg(FIX::Message(fix_msg, dictionary, false));

        std::string value; // FIX field values will be recorder here

        size_t groups_count = 0;
        long seq_number;

        // get data from the header
        try
        {
            groups_count = std::strtol(msg.getField(268).c_str(), nullptr, 10);
            FIX::Header header = msg.getHeader();
            value = header.getField(52);
            long millis = std::strtol(&value[value.size() - 3], nullptr, 10);
            result.time.parse(value, "%Y%m%d%H%M%S");
            result.time += boost::posix_time::milliseconds(millis);
            seq_number = std::strtol(&header.getField(34)[0], nullptr, 10);
            result.exchange = header.getField(49);
        }
        catch (...)
        {
            return false;
        }

        // get data from groups of the message
        FIX::SecurityDesc securityDescField; // FIX field 107
        FIX::Symbol symbolField;             // FIX field 55
        FIX::NumberOfOrders numberOfOrdersField;

        FIX50SP2::MarketDataIncrementalRefresh::NoMDEntries group;
        for (size_t i = 1; i <= groups_count; ++i)
        {
            msg.getGroup(i, group);

            try
            {
                value = group.getField(securityDescField).getString();
                if (symbols.size() != 0 && symbols.find(value) == symbols.cend())
                    continue;

                // discard groups with quote condition = exchange best (field 276 = ’C’)
                // as they aren’t book updates
                try
                {
                    value = group.getField(276);
                    if (value == "C") continue;
                }
                catch (...) { }

                ats::level2_message m;
                m.time = result.time;
                m.seq_number = seq_number;
                m.price = std::stoi(group.getField(270));
                m.quantity = std::stoi(group.getField(271));
/*              try
                {
                    e.symbol = group.getField(symbolField).getString();;
                }
                catch(...) { }*/
                m.symbol = value;
                m.exchange = result.exchange;

                // MDUpdateAction
                value = group.getField(279);
                if (value == "0")
                    m.update_action = ats::update_action::New;
                else if (value == "1")
                    m.update_action = ats::update_action::Change;
                else if (value == "2")
                    m.update_action = ats::update_action::Delete;
                else
                    continue;

                // MDEntryType: discard anythind beyond Bid, Ask, or Trade
                value = group.getField(269);
                if (value == "0")
                    m.entry_type = ats::entry_type::Bid;
                else if (value == "1")
                    m.entry_type = ats::entry_type::Ask;
                else
                {
                    // If a trade
                    if (value == "2")
                    {
                        m.entry_type = ats::entry_type::Trade;
                        try
                        {
//                          int aggressor = std::stoi(group.getField(5797));
                            m.aggressor_side = std::stoi(group.getField(5797));//aggressor;
                            if (m.aggressor_side == 2)
                                m.aggressor_side = -1;
                                    //static_cast<ats::aggressor_side>(aggressor);
                        }
                        catch (...) { }
                        result.messages.push_back(m);
                    }
                    continue;
                }

                m.level = std::stoi(group.getField(1023));
                m.order_count = std::stoi(group.getField(numberOfOrdersField).getString());//stoi(group.getField(346));

                result.messages.push_back(m);
            }
            catch (...)
            {
                continue;
            }
        }

        return true;
    }
```

5.1.3.3 to_spread()

```
static void ats::to_spread (
              const std::string & x_file,
              const std::string & y_file,
              const std::string & out_file,
              const std::string & header = "Symbol; Timestamp; BidPrc; AskPrc" )  [static]
```

References ats::date_time::date_time::to_string().

```
    {
        struct bbo
        {
            int seq_num;
            boost::posix_time::ptime time;
            int bid, ask;
            std::string symbol;
        };

        std::vector<bbo> x, y;
        bbo bbo_;

        boost::posix_time::time_input_facet* facet = new boost::posix_time::time_input_facet(1);
        std::stringstream ss;
        ss.imbue(std::locale(std::locale(), facet));
        facet->format("%Y%m%d %H%M%S%F");

        char symbol[15], exchange[15], time[50];

        std::FILE* file = std::fopen(x_file.c_str(), "r");
        std::fscanf(file, "%*[^\n]\n", nullptr);

        while (std::fscanf(file, "%[^;];%[^;];%d;%[^;];%d;%d\n",
                symbol, exchange, &bbo_.seq_num, time, &bbo_.bid, &bbo_.ask) == 6)
        {
            ss.str(time);
            ss >> bbo_.time;
            ss.clear();

            bbo_.symbol.assign(symbol);

            x.push_back(bbo_);
        }

        std::fclose(file);

        file = std::fopen(y_file.c_str(), "r");
        std::fscanf(file, "%*[^\n]\n", nullptr);

        while (std::fscanf(file, "%[^;];%[^;];%d;%[^;];%d;%d\n",
                symbol, exchange, &bbo_.seq_num, time, &bbo_.bid, &bbo_.ask) == 6)
        {
            ss.str(time);
            ss >> bbo_.time;
            ss.clear();

            bbo_.symbol.assign(symbol);

            bbo_.bid /= 5;
            bbo_.ask /= 5;
            y.push_back(bbo_);
        }

        std::fclose(file);

        // compute the spread
        std::ofstream out(out_file);
        out << header << '\n';

        size_t i = 0, j = 0;
        int seq_num = std::min(x[0].seq_num, y[0].seq_num);

        boost::posix_time::ptime tm = x[0].time;

        if (x[0].seq_num > y[0].seq_num)
        {
            for (j = 1; j < y.size() && y[j].seq_num < x[0].seq_num; ++j) ;
            --j;
        }
        else if (x[0].seq_num < y[0].seq_num)
        {
            for (i = 1; i < x.size() && x[i].seq_num < y[0].seq_num; ++i) ;
            --i;
            tm = y[0].time;
        }

        int bid, ask;
        std::string tm_str;
        while (i < x.size() - 1 && j < y.size() - 1)
        {
            bid = x[i].bid - y[j].ask;
            ask = x[i].ask - y[j].bid;

//          ats::date_time::date_time dt(tm);
            tm_str = ats::date_time::date_time(tm).to_string("%Y%m%d %H%M%S.%f");
            tm_str = tm_str.substr(0, tm_str.size() - 3);
            out << x[i].symbol << "-" << y[j].symbol << "; " << tm_str << "; " << bid << "; " << ask << '\n';

            if (x[i + 1].seq_num < y[j + 1].seq_num)
            {
                tm = x[++i].time;
            }
            else if (x[i + 1].seq_num > y[j + 1].seq_num)
            {
                tm = y[++j].time;
            }
            else
            {
                tm = x[++i].time;
                ++j;
            }
        }

        out.close();
    }
```

5.1.3.4 tokenize()

```
template<size_t num_cols>
static size_t ats::tokenize (
            const std::string & line,
            std::array< std::string, num_cols > & result,
            char sep = ',' )  [static]
```

Referenced by ats::l2_message_reader::read().

```
    {
        size_t i = 0U;
        std::string::const_iterator begin = line.cbegin();
        for (auto it = line.cbegin(); it != line.cend() && i < num_cols; ++it)
        {
            if (*it == sep)
            {
                result[i++].assign(begin, it);
                begin = it + 1;
            }
        }

        if (i == num_cols)
            return num_cols + 1;

        if (i < num_cols)
            result[i++].assign(begin, line.cend());

        return i;
    }
```

5.2 ats::date_time Namespace Reference

Classes

- class date_time
- class stop_watch

5.3 ats::detail Namespace Reference

Classes

- struct basic_function
- struct function_base
- class member_function_handler
- class member_function_handler_base

5.4 ats::order_book_detail Namespace Reference

Classes

- struct price_level

  price level in an order book
- class price_levels

5.5 ats::sim Namespace Reference

Classes

- class fifo_exchange_order_book
- class price_level
- class price_levels
- class sim_book

Chapter 6

Class Documentation

6.1 ats::bar Struct Reference

```
#include <bar.hpp>
```

Collaboration diagram for ats::bar:

Public Attributes

- ats::price_t open
- ats::price_t high
- ats::price_t low
- ats::price_t close
- long quantity
- ats::timestamp_t time_open

6.1.1 Member Data Documentation

6.1.1.1 close

```
ats::price_t ats::bar::close
```

Referenced by ats::security_base::update_bars().

6.1.1.2 high

```
ats::price_t ats::bar::high
```

Referenced by ats::security_base::update_bars().

6.1.1.3 low

```
ats::price_t ats::bar::low
```

Referenced by ats::security_base::update_bars().

6.1.1.4 open

```
ats::price_t ats::bar::open
```

Referenced by ats::security_base::update_bars().

6.1.1.5 quantity

```
long ats::bar::quantity
```

Referenced by ats::security_base::update_bars().

6.1.1.6 time_open

```
ats::timestamp_t ats::bar::time_open
```

Referenced by ats::security_base::update_bars().

The documentation for this struct was generated from the following file:

- security/bar.hpp

6.2 ats::detail::basic_function< T > Struct Template Reference

```
#include <multievent_handler.hpp>
```

Inheritance diagram for ats::detail::basic_function< T >:

Collaboration diagram for ats::detail::basic_function< T >:

Public Types

- typedef std::function< T > function_type

Public Member Functions

- basic_function (const function_type &func)
- basic_function (function_type &&func)

Public Attributes

- function_type function

6.2.1 Member Typedef Documentation

6.2.1.1 function_type

```
template<typename T >
typedef std::function<T> ats::detail::basic_function< T >::function_type
```

6.2.2 Constructor & Destructor Documentation

6.2.2.1 basic_function() [1/2]

```
template<typename T >
ats::detail::basic_function< T >::basic_function (
             const function_type & func )  [inline]
```

```
: function(func) { }
```

6.2.2.2 basic_function() [2/2]

```
template<typename T >
ats::detail::basic_function< T >::basic_function (
             function_type && func )  [inline]
```

```
: function(std::move(func)) { }
```

6.2.3 Member Data Documentation

6.2.3.1 function

```
template<typename T >
function_type ats::detail::basic_function< T >::function
```

Referenced by ats::multievent_handler::invoke(), and ats::multievent_handler::operator()().

The documentation for this struct was generated from the following file:

- event_handler/multievent_handler.hpp

6.3 ats::bracket_order Class Reference

```
#include <bracket_order.hpp>
```

Inheritance diagram for ats::bracket_order:

Collaboration diagram for ats::bracket_order:

Public Member Functions

- bracket_order (const ats::orderid_t &id, ats::limit_order &lmt_order, ats::stop_order &stp_order)
- const ats::orderid_t & limit_order_id () const
- const ats::orderid_t & stop_order_id () const

Additional Inherited Members

6.3.1 Constructor & Destructor Documentation

6.3.1.1 bracket_order()

```
ats::bracket_order::bracket_order (
                const ats::orderid_t & id,
                ats::limit_order & lmt_order,
                ats::stop_order & stp_order )   [inline]
```

References ats::order::id(), and ats::order::parent_id.

```
            : order(id, lmt_order.symbol(), lmt_order.quantity(), lmt_order.side(), lmt_order.time_in_force())
        {
            lmt_order.parent_id = id;
            stp_order.parent_id = id;
        }
```

6.3.2 Member Function Documentation

6.3.2.1 limit_order_id()

```
const ats::orderid_t& ats::bracket_order::limit_order_id ( ) const  [inline]
```

```
{ return limit_order_id_; }
```

6.3.2.2 stop_order_id()

```
const ats::orderid_t& ats::bracket_order::stop_order_id ( ) const  [inline]
```

```
{ return stop_order_id_; }
```

The documentation for this class was generated from the following file:

- order/bracket_order.hpp

6.4 ats::csv_reader< num_columns > Class Template Reference

```
#include <csv_reader.hpp>
```

Inheritance diagram for ats::csv_reader< num_columns >:

Collaboration diagram for ats::csv_reader< num_columns >:

Public Member Functions

- csv_reader (std::ifstream ∗stream=nullptr, char delim=';', bool header=true)
- void set_stream (std::ifstream ∗stream)
- void set_delim (char delim)
- const std::array< std::string, num_columns > & header () const
- bool read (std::array< std::string, num_columns > &fields)

6.4.1 Constructor & Destructor Documentation

6.4.1.1 csv_reader()

```
template<size_t num_columns>
ats::csv_reader< num_columns >::csv_reader (
            std::ifstream * stream = nullptr,
            char delim = ';',
            bool header = true )  [inline]
```

```
            : stream_(stream), delim_(delim), header_flag_(header) { }
```

6.4.2 Member Function Documentation

6.4.2.1 header()

```
template<size_t num_columns>
const std::array<std::string, num_columns>& ats::csv_reader< num_columns >::header ( ) const
[inline]
```

```
{ return header_; }
```

6.4.2.2 read()

```
template<size_t num_columns>
bool ats::csv_reader< num_columns >::read (
            std::array< std::string, num_columns > & fields )  [inline]
```

Referenced by ats::level2_exchange_message_reader::read().

```
        {
            if (header_flag_)
            {
                std::getline(*stream_, line);
                parse(line, header_);
                header_flag_ = false;
            }

            if (std::getline(*stream_, line))
            {
                parse(line, fields);
                return true;
            }
            return false;
        }
```

6.4.2.3 set_delim()

```
template<size_t num_columns>
void ats::csv_reader< num_columns >::set_delim (
             char delim )  [inline]
```

```
18 { delim_ = delim; }
```

6.4.2.4 set_stream()

```
template<size_t num_columns>
void ats::csv_reader< num_columns >::set_stream (
             std::ifstream * stream )  [inline]
```

```
17 { stream_ = stream; }
```

The documentation for this class was generated from the following file:

- io/csv_reader.hpp

6.5 ats::csv_single_message_reader< MessageT, n_columns > Class Template Reference

```
#include <csv_message_reader.hpp>
```

Inheritance diagram for ats::csv_single_message_reader< MessageT, n_columns >:

Collaboration diagram for ats::csv_single_message_reader< MessageT, n_columns >:

Public Member Functions

- csv_single_message_reader (const std::string &filename)

Additional Inherited Members

6.5.1 Constructor & Destructor Documentation

6.5.1.1 csv_single_message_reader()

```
template<typename MessageT , size_t n_columns>
ats::csv_single_message_reader< MessageT, n_columns >::csv_single_message_reader (
            const std::string & filename )  [inline]
```

```
            : ats::single_message_reader<MessageT>(), stream_(filename)
     , reader_(&stream_) { }
```

The documentation for this class was generated from the following file:

- data_feed/historical/csv_message_reader.hpp

6.6 ats::data_feed Class Reference

```
#include <data_feed.hpp>
```

Inheritance diagram for ats::data_feed:

Collaboration diagram for ats::data_feed:

Public Member Functions

- data_feed (ats::portfolio_base ∗universe)
- void set_trading_universe (ats::portfolio_base ∗universe)

Protected Attributes

- ats::portfolio_base ∗ universe_

6.6.1 Constructor & Destructor Documentation

6.6.1.1 data_feed()

```
ats::data_feed::data_feed (
            ats::portfolio_base * universe )  [inline]
```

```
            : universe_(universe) { }
```

6.6.2 Member Function Documentation

6.6.2.1 set_trading_universe()

```
void ats::data_feed::set_trading_universe (
             ats::portfolio_base * universe )  [inline]
```

References universe_.

```
16 { universe_ = universe; }
```

6.6.3 Member Data Documentation

6.6.3.1 universe_

```
ats::portfolio_base* ats::data_feed::universe_  [protected]
```

Referenced by ats::historical_data_feed::send_message(), and set_trading_universe().

The documentation for this class was generated from the following file:

- data_feed/data_feed.hpp

6.7 ats::date_time::date_time Class Reference

```
#include <date_time.hpp>
```

Collaboration diagram for ats::date_time::date_time:

Public Types

- typedef boost::gregorian::date::year_type year_type
- typedef boost::gregorian::date::month_type month_type
- typedef boost::gregorian::date::day_type day_type
- typedef boost::gregorian::date::day_of_week_type day_of_week_type
- typedef boost::gregorian::date::day_of_year_type day_of_year_type
- typedef boost::posix_time::time_duration::hour_type hour_type
- typedef boost::posix_time::time_duration::min_type min_type
- typedef boost::posix_time::time_duration::sec_type sec_type
- typedef boost::posix_time::time_duration::fractional_seconds_type fractional_seconds_type

Public Member Functions

- date_time ()
- date_time (const boost::posix_time::ptime &datetime)
- date_time (const boost::gregorian::date &date)
- date_time (const date_time &datetime)
- date_time (year_type year, month_type month, day_type day)
- date_time (const boost::gregorian::date &date, const boost::posix_time::time_duration &time)
- date_time (boost::posix_time::special_values sv)
- date_time (const std::string &datetime, const std::string &fmt="%Y%m%d %H%M%S%F")
- bool is_not_a_date_time () const
- bool is_neg_infinity () const
- bool is_pos_infinity () const
- bool is_infinity () const
- bool is_special () const
- year_type year () const
- month_type month () const
- day_type day () const
- hour_type hour () const
- min_type minute () const
- sec_type second () const
- long long int millisecond () const
- boost::gregorian::date date () const
- day_of_week_type day_of_week () const
- day_of_year_type day_of_year () const
- boost::posix_time::time_duration time_of_day () const
- bool operator< (const date_time &other) const
- bool operator< (const boost::posix_time::ptime &other) const
- bool operator> (const date_time &other) const
- bool operator> (const boost::posix_time::ptime &other) const
- bool operator== (const date_time &other) const
- bool operator== (const boost::posix_time::ptime &other) const
- bool operator!= (const date_time &other) const
- bool operator!= (const boost::posix_time::ptime &other) const
- bool operator<= (const date_time &other) const
- bool operator>= (const date_time &other) const
- date_time operator+ (const boost::posix_time::time_duration &duration)
- date_time operator- (const boost::posix_time::time_duration &duration)
- const date_time operator+ (const boost::posix_time::time_duration &duration) const
- const date_time operator- (const boost::posix_time::time_duration &duration) const
- boost::posix_time::time_duration operator- (const date_time &other)
- boost::posix_time::time_duration operator- (const boost::posix_time::ptime &other)
- const boost::posix_time::time_duration operator- (const date_time &other) const
- const boost::posix_time::time_duration operator- (const boost::posix_time::ptime &other) const
- date_time & operator+= (const boost::posix_time::time_duration &duration)
- date_time & operator-= (const boost::posix_time::time_duration &duration)
- date_time & operator= (const date_time &other)
- date_time & operator= (const boost::posix_time::ptime &other)
- date_time & operator= (const boost::gregorian::date &other)
- operator boost::posix_time::ptime () const
- void parse_simple (const char ∗datetime)
- void parse_simple (const std::string &datetime)
- void parse (const char ∗datetime, const char ∗fmt="%Y%m%d %H%M%S%F")
- void parse (const std::string &datetime, const std::string &format="%Y%m%d %H%M%S%F")
- void parse (const char ∗datetime, std::stringstream &ss)
- void parse (const std::string &datetime, std::stringstream &ss)
- std::string to_string (const char ∗fmt="%Y-%m-%d %H:%M:%S.%f") const

Static Public Member Functions

- static date_time now ()

Friends

- std::ostream & operator<< (std::ostream &out, const date_time &datetime)

6.7.1 Member Typedef Documentation

6.7.1.1 day_of_week_type

```
typedef boost::gregorian::date::day_of_week_type ats::date_time::date_time::day_of_week_type
```

6.7.1.2 day_of_year_type

```
typedef boost::gregorian::date::day_of_year_type ats::date_time::date_time::day_of_year_type
```

6.7.1.3 day_type

```
typedef boost::gregorian::date::day_type ats::date_time::date_time::day_type
```

6.7.1.4 fractional_seconds_type

```
typedef boost::posix_time::time_duration::fractional_seconds_type ats::date_time::date_time←
::fractional_seconds_type
```

6.7.1.5 hour_type

```
typedef boost::posix_time::time_duration::hour_type ats::date_time::date_time::hour_type
```

6.7.1.6 min_type

```
typedef boost::posix_time::time_duration::min_type ats::date_time::date_time::min_type
```

6.7.1.7 month_type

```
typedef boost::gregorian::date::month_type ats::date_time::date_time::month_type
```

6.7.1.8 sec_type

```
typedef boost::posix_time::time_duration::sec_type ats::date_time::date_time::sec_type
```

6.7.1.9 year_type

```
typedef boost::gregorian::date::year_type ats::date_time::date_time::year_type
```

6.7.2 Constructor & Destructor Documentation

6.7.2.1 date_time() [1/8]

```
ats::date_time::date_time::date_time ( )  [inline]
```

Referenced by now(), operator+(), and operator-().

```
24 : datetime_() { }
```

6.7.2.2 date_time() [2/8]

```
ats::date_time::date_time::date_time (
             const boost::posix_time::ptime & datetime )  [inline]
```

```
25 : datetime_(datetime) { }
```

6.7.2.3 date_time() [3/8]

```
ats::date_time::date_time::date_time (
             const boost::gregorian::date & date )  [inline]
```

```
: datetime_(date) { }
```

6.7.2.4 date_time() [4/8]

```
ats::date_time::date_time::date_time (
             const date_time & datetime )  [inline]
```

```
: datetime_(datetime.datetime_) { }
```

6.7.2.5 date_time() [5/8]

```
ats::date_time::date_time::date_time (
             year_type year,
             month_type month,
             day_type day )  [inline]
```

```
            : datetime_(boost::gregorian::date(year, month, day)) { }
```

6.7.2.6 date_time() [6/8]

```
ats::date_time::date_time::date_time (
             const boost::gregorian::date & date,
             const boost::posix_time::time_duration & time )  [inline]
```

```
            : datetime_(date, time) { }
```

6.7.2.7 date_time() [7/8]

```
ats::date_time::date_time::date_time (
             boost::posix_time::special_values sv )  [inline]
```

```
: datetime_(sv) { }
```

6.7.2.8 date_time() [8/8]

```
ats::date_time::date_time::date_time (
            const std::string & datetime,
            const std::string & fmt = "%Y%m%d %H%M%S%F" )  [inline]
```

References parse().

```
        {
            this->parse(datetime, fmt);
        }
```

6.7.3 Member Function Documentation

6.7.3.1 date()

```
boost::gregorian::date ats::date_time::date_time::date ( ) const  [inline]
```

Referenced by parse_simple(), ats::recursive_timer::update(), and ats::security_base::update_bars().

```
{ return datetime_.date(); }
```

6.7.3.2 day()

```
day_type ats::date_time::date_time::day ( ) const  [inline]
```

Referenced by parse_simple().

```
{ return datetime_.date().day(); }
```

6.7.3.3 day_of_week()

```
day_of_week_type ats::date_time::date_time::day_of_week ( ) const  [inline]
```

```
{ return datetime_.date().day_of_week(); }
```

6.7.3.4 day_of_year()

```
day_of_year_type ats::date_time::date_time::day_of_year ( ) const  [inline]
```

```
{ return datetime_.date().day_of_year(); }
```

6.7.3.5 hour()

```
hour_type ats::date_time::date_time::hour ( ) const  [inline]
```

Referenced by parse_simple().

```
{ return datetime_.time_of_day().hours(); }
```

6.7.3.6 is_infinity()

```
bool ats::date_time::date_time::is_infinity ( ) const  [inline]
```

```
{ return datetime_.is_infinity(); }
```

6.7.3.7 is_neg_infinity()

```
bool ats::date_time::date_time::is_neg_infinity ( ) const  [inline]
```

```
{ return datetime_.is_neg_infinity(); }
```

6.7.3.8 is_not_a_date_time()

```
bool ats::date_time::date_time::is_not_a_date_time ( ) const  [inline]
```

Referenced by ats::recursive_timer::update().

```
{ return datetime_.is_not_a_date_time(); }
```

6.7.3.9 is_pos_infinity()

```
bool ats::date_time::date_time::is_pos_infinity ( ) const  [inline]
```

```
{ return datetime_.is_pos_infinity(); }
```

6.7.3.10 is_special()

```
bool ats::date_time::date_time::is_special ( ) const  [inline]
```

```
{ return datetime_.is_special(); }
```

6.7.3.11 millisecond()

```
long long int ats::date_time::date_time::millisecond ( ) const  [inline]
```

```
        {
            boost::posix_time::time_duration durat(datetime_.time_of_day());
            durat = durat - boost::posix_time::hours(durat.hours()) -
                    boost::posix_time::minutes(durat.minutes()) - boost::posix_time::seconds(durat.seconds(
   ));
            return durat.total_milliseconds();
        }
```

6.7.3.12 minute()

```
min_type ats::date_time::date_time::minute ( ) const  [inline]
```

```
{ return datetime_.time_of_day().minutes(); }
```

6.7.3.13 month()

```
month_type ats::date_time::date_time::month ( ) const  [inline]
```

Referenced by parse_simple().

```
{ return datetime_.date().month(); }
```

6.7.3.14 now()

```
static date_time ats::date_time::date_time::now ( ) [inline], [static]
```

References date_time().

```
{
    return date_time(boost::posix_time::microsec_clock::universal_time());
}
```

6.7.3.15 operator boost::posix_time::ptime()

```
ats::date_time::date_time::operator boost::posix_time::ptime ( ) const [inline]
```

```
{ return datetime_; }
```

6.7.3.16 operator"!=() [1/2]

```
bool ats::date_time::date_time::operator!= (
            const date_time & other ) const [inline]
```

```
{ return datetime_ != other.datetime_; }
```

6.7.3.17 operator"!=() [2/2]

```
bool ats::date_time::date_time::operator!= (
            const boost::posix_time::ptime & other ) const [inline]
```

```
{ return datetime_ != other; }
```

6.7.3.18 operator+() [1/2]

```
date_time ats::date_time::date_time::operator+ (
            const boost::posix_time::time_duration & duration ) [inline]
```

References date_time().

```
{ return date_time(datetime_ + duration); }
```

6.7.3.19 operator+() [2/2]

```
const date_time ats::date_time::date_time::operator+ (
             const boost::posix_time::time_duration & duration ) const  [inline]
```

References date_time().

```
85 { return date_time(datetime_ + duration); }
```

6.7.3.20 operator+=()

```
date_time& ats::date_time::date_time::operator+= (
             const boost::posix_time::time_duration & duration )  [inline]
```

```
94 { datetime_ += duration; return *this; }
```

6.7.3.21 operator-() [1/6]

```
date_time ats::date_time::date_time::operator- (
             const boost::posix_time::time_duration & duration )  [inline]
```

References date_time().

```
84 { return date_time(datetime_ - duration); }
```

6.7.3.22 operator-() [2/6]

```
const date_time ats::date_time::date_time::operator- (
             const boost::posix_time::time_duration & duration ) const  [inline]
```

References date_time().

```
86 { return date_time(datetime_ - duration); }
```

6.7.3.23 operator-() [3/6]

```
boost::posix_time::time_duration ats::date_time::date_time::operator- (
             const date_time & other )   [inline]
```

```
{ return datetime_ - other.datetime_; }
```

6.7.3.24 operator-() [4/6]

```
boost::posix_time::time_duration ats::date_time::date_time::operator- (
             const boost::posix_time::ptime & other )   [inline]
```

```
{ return datetime_ - other; }
```

6.7.3.25 operator-() [5/6]

```
const boost::posix_time::time_duration ats::date_time::date_time::operator- (
             const date_time & other ) const   [inline]
```

```
{ return datetime_ - other.datetime_; }
```

6.7.3.26 operator-() [6/6]

```
const boost::posix_time::time_duration ats::date_time::date_time::operator- (
             const boost::posix_time::ptime & other ) const   [inline]
```

```
{ return datetime_ - other; }
```

6.7.3.27 operator-=()

```
date_time& ats::date_time::date_time::operator-= (
             const boost::posix_time::time_duration & duration )   [inline]
```

```
{ datetime_ -= duration; return *this; }
```

6.7.3.28 operator<() [1/2]

```
bool ats::date_time::date_time::operator< (
             const date_time & other ) const  [inline]
```

```
{ return datetime_ < other.datetime_; }
```

6.7.3.29 operator<() [2/2]

```
bool ats::date_time::date_time::operator< (
             const boost::posix_time::ptime & other ) const  [inline]
```

```
{ return datetime_ < other; }
```

6.7.3.30 operator<=()

```
bool ats::date_time::date_time::operator<= (
             const date_time & other ) const  [inline]
```

```
{ return datetime_ <= other.datetime_; }
```

6.7.3.31 operator=() [1/3]

```
date_time& ats::date_time::date_time::operator= (
             const date_time & other )  [inline]
```

```
{ datetime_ = other.datetime_; return *this; }
```

6.7.3.32 operator=() [2/3]

```
date_time& ats::date_time::date_time::operator= (
             const boost::posix_time::ptime & other )  [inline]
```

```
{ datetime_ = other; return *this; }
```

6.7.3.33 operator=() [3/3]

```
date_time& ats::date_time::date_time::operator= (
             const boost::gregorian::date & other )  [inline]
```

```
{ datetime_ = boost::posix_time::ptime(other); return *this; }
```

6.7.3.34 operator==() [1/2]

```
bool ats::date_time::date_time::operator== (
             const date_time & other ) const  [inline]
```

```
{ return datetime_ == other.datetime_; }
```

6.7.3.35 operator==() [2/2]

```
bool ats::date_time::date_time::operator== (
             const boost::posix_time::ptime & other ) const  [inline]
```

```
{ return datetime_ == other; }
```

6.7.3.36 operator>() [1/2]

```
bool ats::date_time::date_time::operator> (
             const date_time & other ) const  [inline]
```

```
{ return datetime_ > other.datetime_; }
```

6.7.3.37 operator>() [2/2]

```
bool ats::date_time::date_time::operator> (
             const boost::posix_time::ptime & other ) const  [inline]
```

```
{ return datetime_ > other; }
```

6.7.3.38 operator>=()

```
bool ats::date_time::date_time::operator>= (
            const date_time & other ) const  [inline]
```

```
{ return datetime_ >= other.datetime_; }
```

6.7.3.39 parse() [1/4]

```
void ats::date_time::date_time::parse (
            const char * datetime,
            const char * fmt = "%Y%m%d %H%M%S%F" )  [inline]
```

Referenced by date_time(), ats::parse_fix_msg(), ats::level2_exchange_message_reader::read(), and ats::l2_message_reader::read().

```
{
    static boost::posix_time::time_input_facet* facet = nullptr;//new time_input_facet(format);
    static std::stringstream ss;//(date_str);
    if (facet == nullptr)
    {
        facet = new boost::posix_time::time_input_facet(1);
        ss.imbue(std::locale(std::locale(), facet));   // ss.imbue(locale(ss.getloc(), facet));
    }

    facet->format(fmt);//.c_str());
    ss.str(datetime);

    ss >> datetime_;
    ss.clear();
}
```

6.7.3.40 parse() [2/4]

```
void ats::date_time::date_time::parse (
            const std::string & datetime,
            const std::string & format = "%Y%m%d %H%M%S%F" )  [inline]
```

```
{
    static boost::posix_time::time_input_facet* facet = nullptr;//new time_input_facet(format);
    static std::stringstream ss;//(date_str);
    if (facet == nullptr)
    {
        facet = new boost::posix_time::time_input_facet(1);
        ss.imbue(std::locale(std::locale(), facet));   // ss.imbue(locale(ss.getloc(), facet));
    }

    facet->format(format.c_str());
    ss.str(datetime);

    ss >> datetime_;
    ss.clear();
}
```

6.7.3.41 parse() [3/4]

```
void ats::date_time::date_time::parse (
        const char * datetime,
        std::stringstream & ss ) [inline]
```

```
{
    ss.str(datetime);
    ss >> datetime_;
    ss.clear();
}
```

6.7.3.42 parse() [4/4]

```
void ats::date_time::date_time::parse (
        const std::string & datetime,
        std::stringstream & ss ) [inline]
```

```
{
    ss.str(datetime);
    ss >> datetime_;
    ss.clear();
}
```

6.7.3.43 parse_simple() [1/2]

```
void ats::date_time::date_time::parse_simple (
        const char * datetime ) [inline]
```

References date(), day(), hour(), month(), and year().

Referenced by parse_simple().

```
{
    char* stop;
    long date = std::strtol(datetime, &stop, 10);
    long day = date % 100;
    date = (date - day) / 100;
    long month = date % 100;
    long year = (date - month) / 100;
    boost::posix_time::ptime result(boost::gregorian::date(year, month, day));

    long time = std::strtol(stop + 1, &stop, 10);
    long sec = time % 100;
    time = (time - sec) / 100;
    long min = time % 100;
    long hour = (time - min) / 100;

    if (*stop == '.')
    {
        long frac_sec = std::strtol(stop + 1, nullptr, 10);
        double frac = std::strtod(stop, nullptr);
        long resolution = (double)frac_sec / frac;
        frac_sec *= boost::posix_time::time_duration::ticks_per_second() / resolution;
        result += boost::posix_time::time_duration(hour, min, sec, frac_sec);
    }
    else
        result += boost::posix_time::time_duration(hour, min, sec);

    datetime_ = result;
}
```

6.7.3.44 parse_simple() [2/2]

```
void ats::date_time::date_time::parse_simple (
              const std::string & datetime )  [inline]
```

References parse_simple().

```
        {
            parse_simple(datetime.c_str());
        }
```

6.7.3.45 second()

```
sec_type ats::date_time::date_time::second ( ) const  [inline]
```

```
55 { return datetime_.time_of_day().seconds(); }
```

6.7.3.46 time_of_day()

```
boost::posix_time::time_duration ats::date_time::date_time::time_of_day ( ) const  [inline]
```

```
67 { return datetime_.time_of_day(); }
```

6.7.3.47 to_string()

```
std::string ats::date_time::date_time::to_string (
              const char * fmt = "%Y-%m-%d %H:%M:%S.%f" ) const  [inline]
```

Referenced by ats::to_spread().

```
                                                           :%M:%S.%f") const
        {
            static boost::posix_time::time_facet* facet = nullptr;//new time_input_facet(format);
            static std::stringstream ss;
            if (facet == nullptr)
            {
                facet = new boost::posix_time::time_facet(fmt);
                ss.imbue(std::locale(std::locale(), facet));   // ss.imbue(locale(ss.getloc(), facet));
            }

            ss.str("");
            ss.clear();

            ss << datetime_;
            return ss.str();
        }
```

6.7.3.48 year()

```
year_type ats::date_time::date_time::year ( ) const  [inline]
```

Referenced by parse_simple().

```
{ return datetime_.date().year(); }
```

6.7.4 Friends And Related Function Documentation

6.7.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & out,
            const date_time & datetime )  [friend]
```

```
        {
            out << datetime.datetime_;
            return out;
        }
```

The documentation for this class was generated from the following file:

- date_time/date_time.hpp

6.8 ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash > Class Template Reference

```
#include <double_key_lookup.hpp>
```

Collaboration diagram for ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >:

Public Types

- typedef std::unordered_multimap< Key, Value, Hash > container_type
- typedef container_type::iterator iterator
- typedef container_type::const_iterator const_iterator
- typedef std::unordered_map< UniqueKey, iterator, UniqueHash > iterator_container_type

Public Member Functions

- iterator begin ()
- iterator end ()
- const_iterator cbegin () const
- const_iterator cend () const
- size_t size () const
- bool empty () const
- iterator find (const UniqueKey &unique_key)
- std::pair< iterator, iterator > equal_range (const Key &key)
- iterator insert (const UniqueKey &unique_key, const Key &key, const Value &value)
- iterator erase (const UniqueKey &unique_key)
- void clear ()

6.8.1 Member Typedef Documentation

6.8.1.1 const_iterator

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
typedef container_type::const_iterator ats::double_key_lookup< UniqueKey, Key, Value, Unique↩
Hash, Hash >::const_iterator
```

6.8.1.2 container_type

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
typedef std::unordered_multimap<Key, Value, Hash> ats::double_key_lookup< UniqueKey, Key,
Value, UniqueHash, Hash >::container_type
```

6.8.1.3 iterator

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
typedef container_type::iterator ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash,
Hash >::iterator
```

6.8.1.4 iterator_container_type

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
typedef std::unordered_map<UniqueKey, iterator, UniqueHash> ats::double_key_lookup< Unique↩
Key, Key, Value, UniqueHash, Hash >::iterator_container_type
```

6.8.2 Member Function Documentation

6.8.2.1 begin()

template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<UniqueKey>, class Hash = std::hash<Key>>
iterator ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::begin ( ) [inline]

```
{ return values_.begin(); }
```

6.8.2.2 cbegin()

template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<UniqueKey>, class Hash = std::hash<Key>>
const_iterator ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::cbegin ( ) const [inline]

```
{ return values_.cbegin(); }
```

6.8.2.3 cend()

template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<UniqueKey>, class Hash = std::hash<Key>>
const_iterator ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::cend ( ) const [inline]

```
{ return values_.cend(); }
```

6.8.2.4 clear()

template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<UniqueKey>, class Hash = std::hash<Key>>
void ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::clear ( ) [inline]

```
        {
            iterators_.clear();
            values_.clear();
        }
```

6.8.2.5 empty()

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
bool ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::empty ( ) const  [inline]
```

```
{ return values_.empty(); }
```

6.8.2.6 end()

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
iterator ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::end ( )  [inline]
```

```
{ return values_.end(); }
```

6.8.2.7 equal_range()

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
std::pair<iterator, iterator> ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash
>::equal_range (
             const Key & key )  [inline]
```

```
        {
            return values_.equal_range(key);
        }
```

6.8.2.8 erase()

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
iterator ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::erase (
             const UniqueKey & unique_key )  [inline]
```

References ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::find().

```
        {
            auto find = iterators_.find(unique_key);
            if (find != iterators_.end())
            {
                auto it = values_.erase(find->second);
                iterators_.erase(find);
                return it;
            }
            else
                return values_.end();
        }
```

6.8.2.9 find()

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
iterator ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::find (
            const UniqueKey & unique_key )  [inline]
```

Referenced by ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::erase().

```
        {
            auto find = iterators_.find(unique_key);
            return find != iterators_.end() ? find->second : values_.end();
        }
```

6.8.2.10 insert()

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
iterator ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::insert (
            const UniqueKey & unique_key,
            const Key & key,
            const Value & value )  [inline]
```

```
        {
            auto it = values_.insert(typename container_type::value_type(key, value));
            iterators_.insert(typename iterator_container_type::value_type(unique_key, it));
            return it;
        }
```

6.8.2.11 size()

```
template<typename UniqueKey , typename Key , typename Value , class UniqueHash = std::hash<↩
UniqueKey>, class Hash = std::hash<Key>>
size_t ats::double_key_lookup< UniqueKey, Key, Value, UniqueHash, Hash >::size ( ) const
[inline]
```

```
{ return values_.size(); }
```

The documentation for this class was generated from the following file:

- container/double_key_lookup.hpp

6.9 ats::event_handler< T > Class Template Reference

```
#include <event_handler.hpp>
```

Collaboration diagram for ats::event_handler< T >:

The documentation for this class was generated from the following file:

- event_handler/event_handler.hpp

6.10 ats::event_handler< ReturnT(Args...)> Class Template Reference

```
#include <event_handler.hpp>
```

Collaboration diagram for ats::event_handler< ReturnT(Args...)>:

Public Types

- typedef std::function< ReturnT(Args...)> connection_type

Public Member Functions

- event_handler & operator+= (const connection_type &handler)
- event_handler & operator+= (connection_type &&handler)
- template<class Object , ReturnT(Object::∗)(Args...) MethodPtr>
  void add_connection (Object ∗object_ptr)
- void operator() (Args &&... args) const

6.10.1 Member Typedef Documentation

6.10.1.1 connection_type

```
template<typename ReturnT , typename...  Args>
typedef std::function<ReturnT(Args...)> ats::event_handler< ReturnT(Args...)>::connection_type
```

6.10.2 Member Function Documentation

6.10.2.1 add_connection()

```
template<typename ReturnT , typename...  Args>
template<class Object , ReturnT(Object::*)(Args...)  MethodPtr>
void ats::event_handler< ReturnT(Args...)>::add_connection (
            Object * object_ptr )  [inline]
```

```
        {
            connections_.push_back([&object_ptr](Args... args) { return (object_ptr->*MethodPtr)(args...); });
        }
```

6.10.2.2 operator()()

```
template<typename ReturnT , typename... Args>
void ats::event_handler< ReturnT(Args...)>::operator() (
            Args &&... args ) const [inline]
```

```
        {
            for (const auto& h : connections_)
                h(std::forward<Args>(args)...);
        }
```

6.10.2.3 operator+=() [1/2]

```
template<typename ReturnT , typename... Args>
event_handler& ats::event_handler< ReturnT(Args...)>::operator+= (
            const connection_type & handler ) [inline]
```

```
        {
            connections_.push_back(handler);
            return *this;
        }
```

6.10.2.4 operator+=() [2/2]

```
template<typename ReturnT , typename... Args>
event_handler& ats::event_handler< ReturnT(Args...)>::operator+= (
            connection_type && handler ) [inline]
```

```
        {
            connections_.push_back(std::move(handler));
            return *this;
        }
```

The documentation for this class was generated from the following file:

- event_handler/event_handler.hpp

6.11 ats::exchange_message_reader_base< MessageT > Class Template Reference

```
#include <exchange_message_reader_base.hpp>
```

Inheritance diagram for ats::exchange_message_reader_base< MessageT >:

Collaboration diagram for ats::exchange_message_reader_base< MessageT >:

Public Member Functions

- virtual ∼exchange_message_reader_base ()
- virtual void send_message (ats::portfolio_base ∗universe) const override
- virtual const ats::message & get_last_message () const override
- const MessageT & get_last_true_message () const

Protected Attributes

- MessageT message_

6.11.1 Constructor & Destructor Documentation

6.11.1.1 ∼exchange_message_reader_base()

```
template<typename MessageT>
virtual ats::exchange_message_reader_base< MessageT >::∼exchange_message_reader_base ( )
[inline], [virtual]
```

```
{ }
```

6.11.2 Member Function Documentation

6.11.2.1 get_last_message()

```
template<typename MessageT>
virtual const ats::message& ats::exchange_message_reader_base< MessageT >::get_last_message (
) const [inline], [override], [virtual]
```

Implements ats::message_reader.

```
        {
            return static_cast<const ats::message&>(message_);
        }
```

6.11.2.2 get_last_true_message()

```
template<typename MessageT>
const MessageT& ats::exchange_message_reader_base< MessageT >::get_last_true_message ( ) const
[inline]
```

```
        {
            return message_;
        }
```

6.11.2.3 send_message()

```
template<typename MessageT>
virtual void ats::exchange_message_reader_base< MessageT >::send_message (
            ats::portfolio_base * universe ) const  [inline], [override], [virtual]
```

Implements ats::message_reader.

```
        {
            const ats::instrument_message& msg = static_cast<const ats::instrument_message&>(message_);
            ats::execution_engine* engine = universe->get_execution_engine(msg.exchange);
            if (engine != nullptr)
                engine->invoke(message_);//(static_cast<const MessageT&>(message_));
        }
```

6.11.3 Member Data Documentation

6.11.3.1 message_

```
template<typename MessageT>
MessageT ats::exchange_message_reader_base< MessageT >::message_  [protected]
```

Referenced by ats::exchange_message_reader_base< ats::level2_message_packet >::get_last_message(), ats::exchange_message_reader_base< ats::level2_message_packet >::get_last_true_message(), and ats::exchange_message_reader_base< ats::level2_message_packet >::send_message().

The documentation for this class was generated from the following file:

- data_feed/historical/exchange_message_reader_base.hpp

6.12 ats::exchange_order_book Class Reference

```
#include <exchange_order_book.hpp>
```

Collaboration diagram for ats::exchange_order_book:

Public Types

- typedef ats::order_book_detail::price_levels< std::greater< ats::price_t > > bids_type
- typedef ats::order_book_detail::price_levels< std::less< ats::price_t > > asks_type
- typedef bids_type::iterator bid_iterator
- typedef bids_type::const_iterator bid_const_iterator
- typedef asks_type::iterator ask_iterator
- typedef asks_type::const_iterator ask_const_iterator
- typedef ats::order_book_detail::price_level price_level_type

Public Member Functions

- exchange_order_book (const ats::symbol_key &symbol, const std::string &exchange, size_t book_depth)
- void update (const ats::level2_message &msg)
- void update2 (ats::level2_message &msg)
- bid_iterator begin_bid ()
- bid_const_iterator cbegin_bid () const
- ask_iterator begin_ask ()
- ask_const_iterator cbegin_ask () const
- bid_iterator end_bid ()
- bid_const_iterator cend_bid () const
- ask_iterator end_ask ()
- ask_const_iterator cend_ask () const
- size_t displayed_depth () const
- void clear ()
- int bid_ask_spread () const
- const price_level_type ∗ best_bid () const
- const price_level_type ∗ best_ask () const
- const price_level_type ∗ bid_at (ats::price_t price) const
- const price_level_type ∗ ask_at (ats::price_t price) const
- double midpoint () const
- bids_type & bids ()
- const bids_type & bids () const
- asks_type & asks ()
- const asks_type & asks () const
- const ats::symbol_key & symbol () const
- const std::string & exchange () const
- const ats::timestamp_t & last_update_time () const

Friends

- std::ostream & operator<< (std::ostream &os, const ats::exchange_order_book &book)

6.12.1 Member Typedef Documentation

6.12.1.1 ask_const_iterator

```
typedef asks_type::const_iterator ats::exchange_order_book::ask_const_iterator
```

6.12.1.2 ask_iterator

```
typedef asks_type::iterator ats::exchange_order_book::ask_iterator
```

6.12.1.3 asks_type

```
typedef ats::order_book_detail::price_levels<std::less<ats::price_t> > ats::exchange_order_book::asks_type
```

6.12.1.4 bid_const_iterator

```
typedef bids_type::const_iterator ats::exchange_order_book::bid_const_iterator
```

6.12.1.5 bid_iterator

```
typedef bids_type::iterator ats::exchange_order_book::bid_iterator
```

6.12.1.6 bids_type

```
typedef ats::order_book_detail::price_levels<std::greater<ats::price_t> > ats::exchange_order_book::bids_type
```

6.12.1.7 price_level_type

```
typedef ats::order_book_detail::price_level ats::exchange_order_book::price_level_type
```

6.12.2 Constructor & Destructor Documentation

6.12.2.1 exchange_order_book()

```
ats::exchange_order_book::exchange_order_book (
            const ats::symbol_key & symbol,
            const std::string & exchange,
            size_t book_depth )  [inline]
```

```
          : max_levels_(book_depth), bids_(book_depth), asks_(book_depth),
            symbol_(symbol), exchange_(exchange) { }
```

6.12.3 Member Function Documentation

6.12.3.1 ask_at()

```
const price_level_type* ats::exchange_order_book::ask_at (
            ats::price_t price ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cend(), and ats::order_book_detail::price_levels< compare >::find().

Referenced by ats::sim::fifo_exchange_order_book::add_order(), and ats::sim::fifo_exchange_order_book::update().

```
        {
            auto it = asks_.find(price);
            return it != asks_.cend() ? &it->second : nullptr;
        }
```

6.12.3.2 asks() [1/2]

```
asks_type& ats::exchange_order_book::asks ( )  [inline]
```

Referenced by ats::sim::fifo_exchange_order_book::add_order(), ats::level2_execution_engine::on_order_book_changed(), and ats::level2_execution_engine::send_order().

```
{ return asks_; }
```

6.12.3.3 asks() [2/2]

```
const asks_type& ats::exchange_order_book::asks ( ) const  [inline]
```

```
{ return asks_; }
```

6.12.3.4 begin_ask()

```
ask_iterator ats::exchange_order_book::begin_ask ( )  [inline]
```

References ats::order_book_detail::price_levels< compare >::begin().

```
{ return asks_.begin(); }
```

6.12.3.5 begin_bid()

```
bid_iterator ats::exchange_order_book::begin_bid ( )  [inline]
```

References ats::order_book_detail::price_levels< compare >::begin().

```
{ return bids_.begin(); }
```

6.12.3.6 best_ask()

```
const price_level_type* ats::exchange_order_book::best_ask ( ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cbegin(), and ats::order_book_detail::price_levels< compare >::empty().

Referenced by ats::sim::fifo_exchange_order_book::add_order(), ats::level2_execution_engine::on_order_book_changed(), ats::level2_execution_engine::send_order(), and ats::sim::fifo_exchange_order_book::update().

```
        {
            return !asks_.empty() ? &asks_.cbegin()->second : nullptr;
        }
```

6.12.3.7 best_bid()

```
const price_level_type* ats::exchange_order_book::best_bid ( ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cbegin(), and ats::order_book_detail::price_levels< compare >::empty().

Referenced by ats::sim::fifo_exchange_order_book::add_order(), ats::level2_execution_engine::on_order_book_changed(), ats::level2_execution_engine::send_order(), and ats::sim::fifo_exchange_order_book::update().

```
        {
            return !bids_.empty() ? &bids_.cbegin()->second : nullptr;
        }
```

6.12.3.8 bid_ask_spread()

```
int ats::exchange_order_book::bid_ask_spread ( ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cbegin().

```
{ return asks_.cbegin()->first - bids_.cbegin()->first; }
```

6.12.3.9 bid_at()

```
const price_level_type* ats::exchange_order_book::bid_at (
            ats::price_t price ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cend(), and ats::order_book_detail::price_levels< compare >::find().

Referenced by ats::sim::fifo_exchange_order_book::add_order(), and ats::sim::fifo_exchange_order_book::update().

```
        {
            auto it = bids_.find(price);
            return it != bids_.cend() ? &it->second : nullptr;
        }
```

6.12.3.10 bids() [1/2]

```
bids_type& ats::exchange_order_book::bids ( )  [inline]
```

Referenced by ats::sim::fifo_exchange_order_book::add_order(), ats::level2_execution_engine::on_order_book_changed(), and ats::level2_execution_engine::send_order().

```
{ return bids_; }
```

6.12.3.11 bids() [2/2]

```
const bids_type& ats::exchange_order_book::bids ( ) const  [inline]
```

```
{ return bids_; }
```

6.12.3.12 cbegin_ask()

```
ask_const_iterator ats::exchange_order_book::cbegin_ask ( ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cbegin().

```
{ return asks_.cbegin(); }
```

6.12.3.13 cbegin_bid()

```
bid_const_iterator ats::exchange_order_book::cbegin_bid ( ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cbegin().

```
{ return bids_.cbegin(); }
```

6.12.3.14 cend_ask()

```
ask_const_iterator ats::exchange_order_book::cend_ask ( ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cend().

```
{ return asks_.cend(); }
```

6.12.3.15 cend_bid()

```
bid_const_iterator ats::exchange_order_book::cend_bid ( ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cend().

```
{ return bids_.cend(); }
```

6.12.3.16 clear()

```
void ats::exchange_order_book::clear ( )  [inline]
```

References ats::order_book_detail::price_levels< compare >::clear().

```
{ bids_.clear(); asks_.clear(); }
```

6.12.3.17 displayed_depth()

```
size_t ats::exchange_order_book::displayed_depth ( ) const  [inline]
```

```
{ return max_levels_; }
```

6.12.3.18 end_ask()

```
ask_iterator ats::exchange_order_book::end_ask ( )  [inline]
```

References ats::order_book_detail::price_levels< compare >::end().

```
{ return asks_.end(); }
```

6.12.3.19 end_bid()

```
bid_iterator ats::exchange_order_book::end_bid ( )  [inline]
```

References ats::order_book_detail::price_levels< compare >::end().

```
{ return bids_.end(); }
```

6.12.3.20 exchange()

```
const std::string& ats::exchange_order_book::exchange ( ) const  [inline]
```

```
{ return exchange_; }
```

6.12.3.21 last_update_time()

```
const ats::timestamp_t& ats::exchange_order_book::last_update_time ( ) const  [inline]
```

```
{ return last_update_time_; }
```

6.12.3.22 midpoint()

```
double ats::exchange_order_book::midpoint ( ) const  [inline]
```

References ats::order_book_detail::price_levels< compare >::cbegin(), and ats::order_book_detail::price_levels< compare >::empty().

```
        {
            return !bids_.empty() && !asks_.empty() ? (double)(bids_.
cbegin()->first + asks_.cbegin()->first) / 2.0 : 0.0;
        }
```

6.12.3.23 symbol()

```
const ats::symbol_key& ats::exchange_order_book::symbol ( ) const  [inline]
```

```
{ return symbol_; }
```

6.12.3.24 update()

```
void ats::exchange_order_book::update (
             const ats::level2_message & msg )  [inline]
```

References ats::Ask, ats::Bid, ats::level2_message::entry_type, ats::message::time, and ats::order_book_detail::price_levels< compare >::update().

Referenced by ats::sim::fifo_exchange_order_book::update(), and ats::order_book::update().

```
        {
            last_update_time_ = msg.time;

            if (msg.entry_type == ats::entry_type::Bid)
                bids_.update(msg);
            else if (msg.entry_type == ats::entry_type::Ask)
                asks_.update(msg);
        }
```

6.12.3.25 update2()

```
void ats::exchange_order_book::update2 (
             ats::level2_message & msg )  [inline]
```

References ats::level2_message::aggressor_side, ats::Ask, ats::Bid, ats::order_book_detail::price_levels< compare >::cbegin(), ats::order_book_detail::price_levels< compare >::empty(), ats::level2_message::entry_type, ats::level2_message::price, ats::message::time, ats::Trade, and ats::order_book_detail::price_levels< compare >::update2().

Referenced by ats::order_book::update2().

```
        {
            last_update_time_ = msg.time;

            if (msg.entry_type == ats::entry_type::Bid)
                bids_.update2(msg);
            else if (msg.entry_type == ats::entry_type::Ask)
                asks_.update2(msg);
            else if (msg.entry_type == ats::entry_type::Trade && msg.aggressor_side == 0)
            {
                if (!asks_.empty() && msg.price >= asks_.cbegin()->first)
                    msg.aggressor_side = 1;
                else if (!bids_.empty() && msg.price <= bids_.cbegin()->first)
                    msg.aggressor_side = -1;
            }
        }
```

6.12.4 Friends And Related Function Documentation

6.12.4.1 operator<<

```
std::ostream& operator<< (
               std::ostream & os,
               const ats::exchange_order_book & book )  [friend]
```

```
    {
        std::vector<std::string> bids;
        std::vector<std::string> asks;

        std::stringstream ss;
        size_t max_bid_len = 0;
        for (auto it = book.bids().cbegin(); it != book.bids().cend(); ++it)
        {
            ss << it->first << "(" << it->second.quantity << ")";
            std::string text = ss.str();
            if (text.length() > max_bid_len)
                max_bid_len = text.length();
            bids.push_back(text);
            ss.str("");
            ss.clear();
        }

        for (auto it = book.asks().cbegin(); it != book.asks().cend(); ++it)
        {
            ss << it->first << "(" << it->second.quantity << ")";
            asks.push_back(ss.str());
            ss.str("");
            ss.clear();
        }

        auto it_b = bids.begin();
        auto it_a = asks.begin();
        for (; it_b != bids.end() && it_a != asks.end(); ++it_b, ++it_a)
        {
            os << *it_b;
            size_t dl = max_bid_len - it_b->length();
            for (size_t i = 0; i < dl; ++i)
                os << " ";
            os << " | " << *it_a << '\n';
        }

        for (; it_b != bids.end(); ++it_b)
        {
            os << *it_b;
            size_t dl = max_bid_len - it_b->length();
            for (size_t i = 0; i < dl; ++i)
                os << " ";
            os << " |\n";
        }

        for (; it_a != asks.end(); ++it_a)
        {
            for (size_t i = 0; i < max_bid_len; ++i)
                os << " ";
            os << " | " << *it_a << '\n';
        }

        return os;
    }
```

The documentation for this class was generated from the following file:

- order_book/exchange_order_book.hpp

6.13 ats::execution Struct Reference

```
#include <execution.hpp>
```

Collaboration diagram for ats::execution:

Public Attributes

- std::string symbol
- timestamp_t time
- price_t price
- long quantity
- double commission = 0.0

6.13.1 Member Data Documentation

6.13.1.1 commission

```
double ats::execution::commission = 0.0
```

6.13.1.2 price

```
price_t ats::execution::price
```

6.13.1.3 quantity

```
long ats::execution::quantity
```

6.13.1.4 symbol

```
std::string ats::execution::symbol
```

6.13.1.5 time

`timestamp_t ats::execution::time`

The documentation for this struct was generated from the following file:

- order_processor/execution.hpp

6.14 ats::execution_engine Class Reference

`#include <execution_engine.hpp>`

Inheritance diagram for ats::execution_engine:

Collaboration diagram for ats::execution_engine:

Public Member Functions

- execution_engine (const std::string &name, const ats::subscription &subscription)
- virtual ∼execution_engine ()
- virtual void subscribe (const ats::symbol_key &)=0
- virtual void send_order (const ats::market_order &)=0
- virtual void send_order (const ats::limit_order &)=0
- virtual void send_order (const ats::stop_order &)=0
- virtual void cancel_order (const ats::orderid_t &order_id)=0
- const std::string & name () const
- const ats::subscription & subscription () const
- void add_order_status_listener (const ats::order_status_handler &handler)
- void on_order_status_changed (const ats::order_status_message &msg)

6.14.1 Constructor & Destructor Documentation

6.14.1.1 execution_engine()

```
ats::execution_engine::execution_engine (
            const std::string & name,
            const ats::subscription & subscription ) [inline]
```

References ats::multievent_handler::add_event_handler(), and on_order_status_changed().

```
            : name_(name), subscription_(subscription)
        {
            add_event_handler(&
    execution_engine::on_order_status_changed, this);
        }
```

6.14.1.2 ∼execution_engine()

```
virtual ats::execution_engine::∼execution_engine ( )  [inline], [virtual]
```

References cancel_order(), send_order(), and subscribe().

```
{ }
```

6.14.2 Member Function Documentation

6.14.2.1 add_order_status_listener()

```
void ats::execution_engine::add_order_status_listener (
             const ats::order_status_handler & handler )  [inline]
```

Referenced by ats::portfolio_base::add_connection().

```
        {
            order_status_handler_ = handler;
        }
```

6.14.2.2 cancel_order()

```
virtual void ats::execution_engine::cancel_order (
             const ats::orderid_t & order_id )  [pure virtual]
```

Implemented in ats::level2_execution_engine.

Referenced by ats::portfolio_base::cancel_order(), and ∼execution_engine().

6.14.2.3 name()

```
const std::string& ats::execution_engine::name ( ) const  [inline]
```

Referenced by ats::portfolio_base::add_connection(), and ats::level2_execution_engine::subscribe().

```
{ return name_; }
```

6.14.2.4 on_order_status_changed()

```
void ats::execution_engine::on_order_status_changed (
            const ats::order_status_message & msg )  [inline]
```

Referenced by ats::level2_execution_engine::cancel_order(), execution_engine(), ats::level2_execution_engine::on_order_book_changed(), ats::level2_execution_engine::send_order(), and ats::level2_execution_engine::subscribe().

```
        {
            if (order_status_handler_ != nullptr)
                order_status_handler_(msg);
        }
```

6.14.2.5 send_order() [1/3]

```
virtual void ats::execution_engine::send_order (
            const ats::market_order &  )  [pure virtual]
```

Implemented in ats::level2_execution_engine.

Referenced by ats::portfolio_base::send_order(), and ∼execution_engine().

6.14.2.6 send_order() [2/3]

```
virtual void ats::execution_engine::send_order (
            const ats::limit_order &  )  [pure virtual]
```

Implemented in ats::level2_execution_engine.

6.14.2.7 send_order() [3/3]

```
virtual void ats::execution_engine::send_order (
            const ats::stop_order &  )  [pure virtual]
```

Implemented in ats::level2_execution_engine.

6.14.2.8 subscribe()

```
virtual void ats::execution_engine::subscribe (
            const ats::symbol_key &  )  [pure virtual]
```

Implemented in ats::level2_execution_engine.

Referenced by ats::portfolio_base::add_connection(), and ∼execution_engine().

6.14.2.9 subscription()

```
const ats::subscription& ats::execution_engine::subscription ( ) const  [inline]
```

Referenced by ats::portfolio_base::add_connection().

```
35 { return subscription_; }
```

The documentation for this class was generated from the following file:

- execution_engine/execution_engine.hpp

6.15 ats::sim::fifo_exchange_order_book Class Reference

```
#include <fifo_exchange_order_book.hpp>
```

Collaboration diagram for ats::sim::fifo_exchange_order_book:

Public Types

- typedef std::unordered_map< ats::orderid_t, price_level::iterator > order_container

Public Member Functions

- fifo_exchange_order_book (const ats::symbol_key &symbol, const std::string &exchange, size_t book_depth)
- void add_order (const ats::limit_order &order)
- void cancel_order (const ats::orderid_t &id, const ats::timestamp_t &time)
- void update (const ats::level2_message &msg)
- void add_order_status_listener (const ats::order_status_handler &listener)
- const ats::sim::price_level ∗ get_level (ats::price_t price, bool is_bid) const
- const ats::exchange_order_book & get_order_book () const
- const order_container & get_sim_orders () const
- void process_change_msg (const ats::level2_message &msg)
- void process_delete_msg (const ats::level2_message &msg)
- void process_trade_msg (const ats::level2_message &msg)
- void process_insert_msg (const ats::level2_message &msg)

6.15.1 Member Typedef Documentation

6.15.1.1 order_container

```
typedef std::unordered_map<ats::orderid_t, price_level::iterator> ats::sim::fifo_exchange_order_book::order_container
```

6.15.2 Constructor & Destructor Documentation

6.15.2.1 fifo_exchange_order_book()

```
ats::sim::fifo_exchange_order_book::fifo_exchange_order_book (
               const ats::symbol_key & symbol,
               const std::string & exchange,
               size_t book_depth )   [inline]
```

References add_order().

```
         : book_(symbol, exchange, book_depth) { }
```

6.15.3 Member Function Documentation

6.15.3.1 add_order()

```
void ats::sim::fifo_exchange_order_book::add_order (
               const ats::limit_order & order )   [inline]
```

References ats::sim::sim_book::add_order(), ats::exchange_order_book::ask_at(), ats::exchange_order_book::asks(), ats::exchange_order_book::best_ask(), ats::exchange_order_book::best_bid(), ats::exchange_order_book::bid_at(), ats::exchange_order_book::bids(), ats::Buy, ats::BuyCover, ats::order_book_detail::price_levels< compare >::crbegin(), ats::order_book_detail::price_levels< compare >::empty(), ats::sim::sim_book::get_level(), ats::GTC, ats::order::id(), ats::sim::sim_book::insert_order(), ats::sim::price_level::is_defined(), ats::order_book_detail::price_level::price, ats::limit_order::price(), ats::order::quantity(), ats::order::side(), ats::order::symbol(), and ats::order::transact_time.

Referenced by fifo_exchange_order_book().

```
 {
     if (order.side() == ats::order_side::Buy || order.
side() == ats::order_side::BuyCover)
     {
         if (book_.best_ask() != nullptr && order.price() >= book_.
best_ask()->price)
         {
             ats::order_status_filled_message msg(order.
id(), order.transact_time,
                     book_.best_ask()->price, order.quantity());
             order_status_listener_(msg);
         }
         else
         {
             const auto* true_l = book_.bid_at(order.price());
             const ats::sim::price_level* l = sim_book_.
get_level(order.price(), true);
             if (true_l == nullptr && !book_.bids().empty() && order.
price() <= book_.best_bid()->price
                     && order.price() >= book_.bids().crbegin()->first)
                 sim_book_.insert_order(order);
             else
             {
                 if (true_l != nullptr && (l == nullptr || !l->is_defined()))
                 {
                     ats::limit_order ord(0, order.symbol(), true_l->quantity,
order.side(),
                             ats::order_time_in_force::GTC, order.
price());
                     sim_book_.insert_order(ord);
                 }
                 sim_book_.add_order(order);
             }
         }
     }
     else
     {
         if (book_.best_bid() != nullptr && order.price() <= book_.
best_bid()->price)
         {
             ats::order_status_filled_message msg(order.
id(), order.transact_time,
                     book_.best_bid()->price, order.quantity());
             order_status_listener_(msg);
         }
         else
         {
             const auto* true_l = book_.ask_at(order.price());
             const ats::sim::price_level* l = sim_book_.
get_level(order.price(), false);
             if (true_l == nullptr && !book_.asks().empty() && order.
price() >= book_.best_ask()->price
                     && order.price() <= book_.asks().crbegin()->first)
                 sim_book_.insert_order(order);
             else
             {
                 if (true_l != nullptr && (l == nullptr || !l->is_defined()))
                 {
                     ats::limit_order ord(0, order.symbol(), true_l->quantity,
order.side(),
                             ats::order_time_in_force::GTC, order.
price());
                     sim_book_.insert_order(ord);
                 }
                 sim_book_.add_order(order);
             }
         }
     }
 }
```

6.15.3.2 add_order_status_listener()

```
void ats::sim::fifo_exchange_order_book::add_order_status_listener (
               const ats::order_status_handler & listener )   [inline]
```

References ats::sim::sim_book::add_order_status_listener().

Referenced by ats::level2_execution_engine::subscribe().

```
     {
         order_status_listener_ = listener;
         sim_book_.add_order_status_listener(listener);
     }
```

6.15.3.3 cancel_order()

```
void ats::sim::fifo_exchange_order_book::cancel_order (
               const ats::orderid_t & id,
               const ats::timestamp_t & time )   [inline]
```

References ats::sim::sim_book::cancel_order(), and update().

```
     { sim_book_.cancel_order(id, time); }
```

6.15.3.4 get_level()

```
const ats::sim::price_level* ats::sim::fifo_exchange_order_book::get_level (
            ats::price_t price,
            bool is_bid ) const  [inline]
```

References ats::sim::sim_book::get_level().

```
        { return sim_book_.get_level(price, is_bid); }
```

6.15.3.5 get_order_book()

```
const ats::exchange_order_book& ats::sim::fifo_exchange_order_book::get_order_book ( ) const
[inline]
```

```
{ return book_; }
```

6.15.3.6 get_sim_orders()

```
const order_container& ats::sim::fifo_exchange_order_book::get_sim_orders ( ) const  [inline]
```

References ats::sim::sim_book::get_sim_orders(), process_change_msg(), and process_delete_msg().

```
{ return sim_book_.get_sim_orders(); }
```

6.15.3.7 process_change_msg()

```
void ats::sim::fifo_exchange_order_book::process_change_msg (
            const ats::level2_message & msg )  [inline]
```

References ats::sim::sim_book::process_change_msg().

Referenced by get_sim_orders().

```
    {
        sim_book_.process_change_msg(msg);
    }
```

6.15.3.8 process_delete_msg()

```
void ats::sim::fifo_exchange_order_book::process_delete_msg (
             const ats::level2_message & msg )  [inline]
```

References ats::sim::sim_book::process_delete_msg().

Referenced by get_sim_orders().

```
    {
        sim_book_.process_delete_msg(msg);
    }
```

6.15.3.9 process_insert_msg()

```
void ats::sim::fifo_exchange_order_book::process_insert_msg (
             const ats::level2_message & msg )  [inline]
```

References ats::sim::sim_book::process_insert_msg().

```
{ sim_book_.process_insert_msg(msg); }
```

6.15.3.10 process_trade_msg()

```
void ats::sim::fifo_exchange_order_book::process_trade_msg (
             const ats::level2_message & msg )  [inline]
```

References ats::sim::sim_book::process_trade_msg().

```
{ sim_book_.process_trade_msg(msg); }
```

6.15.3.11 update()

```
void ats::sim::fifo_exchange_order_book::update (
             const ats::level2_message & msg )  [inline]
```

References ats::level2_message::aggressor_side, ats::exchange_order_book::ask_at(), ats::exchange_order_book::best_ask(), ats::exchange_order_book::best_bid(), ats::Bid, ats::exchange_order_book::bid_at(), ats::Change, ats::level2_message::entry_type, ats::sim::sim_book::execute_crosses(), ats::order_book_detail::price_level::price, ats::level2_message::price, ats::sim::sim_book::process_level2_msg(), ats::order_book_detail::price_level::quantity, ats::level2_message::quantity, ats::message::time, ats::Trade, ats::exchange_order_book::update(), and ats::level2_message::update_action.

Referenced by cancel_order().

```
  {
      // Create a "delta" message
      ats::level2_message msg_delta = msg;
      if (msg.update_action == ats::update_action::Change)
      {
          const ats::exchange_order_book::price_level_type*
lptr = msg.entry_type == ats::entry_type::Bid ?
                  book_.bid_at(msg.price) : book_.ask_at(msg.
price);

          if (lptr != nullptr)
              msg_delta.quantity = msg.quantity - lptr->
quantity;
      }
      // this is not used (at least for now):
      else if (msg.entry_type == ats::entry_type::Trade)
      {
          if (book_.best_ask() != nullptr && msg.price >= book_.
best_ask()->price)
              msg_delta.aggressor_side = 1;
          else if (book_.best_bid() != nullptr && msg.price <= book_.
best_bid()->price)
              msg_delta.aggressor_side = -1;
          else
              msg_delta.aggressor_side = 0;
      }

      book_.update(msg);

      const ats::price_t* bid = book_.best_bid() == nullptr ? nullptr : &book_.
best_bid()->price;
      const ats::price_t* ask = book_.best_ask() == nullptr ? nullptr : &book_.
best_ask()->price;
      sim_book_.execute_crosses(bid, ask, msg.time);

      sim_book_.process_level2_msg(msg_delta);
  }
```

The documentation for this class was generated from the following file:

- order_book/simulation/fifo_exchange_order_book.hpp

6.16 ats::detail::function_base Struct Reference

```
#include <multievent_handler.hpp>
```

Inheritance diagram for ats::detail::function_base:

Collaboration diagram for ats::detail::function_base:

The documentation for this struct was generated from the following file:

- event_handler/multievent_handler.hpp

6.17 ats::historical_data_feed Class Reference

```
#include <historical_data_feed.hpp>
```

Inheritance diagram for ats::historical_data_feed:

Collaboration diagram for ats::historical_data_feed:

Public Member Functions

- historical_data_feed (ats::portfolio_base ∗universe)
- virtual ∼historical_data_feed ()
- void add_message_reader (const msg_reader_ptr &reader)
- void add_message_reader (msg_reader_ptr &&reader)
- bool read ()
- void send_message ()
- void run ()

Additional Inherited Members

6.17.1 Constructor & Destructor Documentation

6.17.1.1 historical_data_feed()

```
ats::historical_data_feed::historical_data_feed (
            ats::portfolio_base ∗ universe )  [inline]
```

```
            : ats::data_feed(universe) { }
```

6.17.1.2 ∼historical_data_feed()

```
virtual ats::historical_data_feed::∼historical_data_feed ( )  [inline], [virtual]
```

```
{ }
```

6.17.2 Member Function Documentation

6.17.2.1 add_message_reader() [1/2]

```
void ats::historical_data_feed::add_message_reader (
            const msg_reader_ptr & reader )  [inline]
```

```
{ readers_.push_back(reader); }
```

6.17.2.2 add_message_reader() [2/2]

```
void ats::historical_data_feed::add_message_reader (
            msg_reader_ptr && reader )  [inline]
```

```
{ readers_.push_back(std::move(reader)); }
```

6.17.2.3 read()

```
bool ats::historical_data_feed::read ( )  [inline]
```

References ats::message::time.

Referenced by run().

```
        {
            if (indices_.empty())
            {
                for (size_t i = 0; i < readers_.size(); ++i)
                {
                    if (readers_[i]->read())
                    {
                        const ats::message& msg = readers_[i]->get_last_message();
                        indices_.insert(std::make_pair(msg.time, i));
                    }
                }
            }
            else
            {
                size_t i = indices_.begin()->second;
                indices_.erase(indices_.begin());
                if (readers_[i]->read())
                {
                    const ats::message& msg = readers_[i]->get_last_message();
                    indices_.insert(std::make_pair(msg.time, i));
                }
            }

            return !indices_.empty();
        }
```

6.17.2.4 run()

```
void ats::historical_data_feed::run ( )  [inline]
```

References read(), and send_message().

```
{
    while (read())
    {
        send_message();
    }
}
```

6.17.2.5 send_message()

```
void ats::historical_data_feed::send_message ( )  [inline]
```

References ats::data_feed::universe_.

Referenced by run().

```
{
    if (!indices_.empty())
    {
        size_t i = indices_.begin()->second;
        readers_[i]->send_message(this->universe_);
    }
}
```

The documentation for this class was generated from the following file:

- data_feed/historical/historical_data_feed.hpp

6.18 ats::indicator< ValueT > Class Template Reference

```
#include <indicator.hpp>
```

Collaboration diagram for ats::indicator< ValueT >:

Public Member Functions

- virtual ∼indicator ()
- const ValueT & value () const

Protected Attributes

- ValueT value_

6.18.1 Constructor & Destructor Documentation

6.18.1.1 ∼indicator()

```
template<typename ValueT = double>
virtual ats::indicator< ValueT >::∼indicator ( )  [inline], [virtual]
```

```
11 { }
```

6.18.2 Member Function Documentation

6.18.2.1 value()

```
template<typename ValueT = double>
const ValueT& ats::indicator< ValueT >::value ( ) const  [inline]
```

References ats::indicator< ValueT >::value_.

```
13 { return value_; }
```

6.18.3 Member Data Documentation

6.18.3.1 value_

```
template<typename ValueT = double>
ValueT ats::indicator< ValueT >::value_  [protected]
```

Referenced by ats::indicator< ValueT >::value().

The documentation for this class was generated from the following file:

- indicator/indicator.hpp

6.19 ats::instrument_message Struct Reference

```
#include <message.hpp>
```

Inheritance diagram for ats::instrument_message:

Collaboration diagram for ats::instrument_message:

Public Member Functions

- virtual ∼instrument_message ()

Public Attributes

- std::string symbol
- std::string exchange

6.19.1 Constructor & Destructor Documentation

6.19.1.1 ∼instrument_message()

```
virtual ats::instrument_message::∼instrument_message ( ) [inline], [virtual]
```

```
24 { }
```

6.19.2 Member Data Documentation

6.19.2.1 exchange

```
std::string ats::instrument_message::exchange
```

Referenced by ats::fix_to_csv(), ats::l2_message_reader::l2_message_reader(), ats::parse_fix_msg(), ats::level2_exchange_message_reader::read(), ats::l2_message_reader::read(), ats::exchange_message_reader_base< ats::level2_message_packet >::send_message(), ats::order_book::update(), and ats::order_book::update2().

6.19.2.2 symbol

```
std::string ats::instrument_message::symbol
```

Referenced by ats::l2_message_reader::l2_message_reader(), ats::level2_execution_engine::on_order_book_changed(), ats::parse_fix_msg(), ats::sim::price_level::process_change_msg(), ats::sim::price_levels< std::greater< ats::price_t > >::process_insert_msg(), ats::portfolio_base::process_message(), ats::level2_exchange_message_reader::read(), and ats::l2_message_reader::read().

The documentation for this struct was generated from the following file:

- message/message.hpp

6.20 ats::instrument_message_packet< MessageT > Struct Template Reference

```
#include <message.hpp>
```

Inheritance diagram for ats::instrument_message_packet< MessageT >:

Collaboration diagram for ats::instrument_message_packet< MessageT >:

Public Types

- typedef std::vector< MessageT >::iterator iterator
- typedef std::vector< MessageT >::const_iterator const_iterator

Public Member Functions

- iterator begin ()
- const_iterator cbegin () const
- iterator end ()
- const_iterator cend () const
- MessageT & operator[ ] (size_t index)
- const MessageT & operator[ ] (size_t index) const

Public Attributes

- std::vector< MessageT > messages

6.20.1 Member Typedef Documentation

6.20.1.1 const_iterator

```
template<typename MessageT >
typedef std::vector<MessageT>::const_iterator ats::instrument_message_packet< MessageT >↩
::const_iterator
```

6.20.1.2 iterator

```
template<typename MessageT >
typedef std::vector<MessageT>::iterator ats::instrument_message_packet< MessageT >::iterator
```

6.20.2 Member Function Documentation

6.20.2.1 begin()

```
template<typename MessageT >
iterator ats::instrument_message_packet< MessageT >::begin ( )  [inline]
```

```
{ return messages.begin(); }
```

6.20.2.2 cbegin()

```
template<typename MessageT >
const_iterator ats::instrument_message_packet< MessageT >::cbegin ( ) const  [inline]
```

```
{ return messages.cbegin(); }
```

6.20.2.3 cend()

```
template<typename MessageT >
const_iterator ats::instrument_message_packet< MessageT >::cend ( ) const  [inline]
```

```
{ return messages.cend(); }
```

6.20.2.4 end()

```
template<typename MessageT >
iterator ats::instrument_message_packet< MessageT >::end ( )  [inline]
```

```
{ return messages.end(); }
```

6.20.2.5 operator[]() [1/2]

```
template<typename MessageT >
MessageT& ats::instrument_message_packet< MessageT >::operator[] (
             size_t index )  [inline]
```

```
{ return messages[index]; }
```

6.20.2.6 operator[]() [2/2]

```
template<typename MessageT >
const MessageT& ats::instrument_message_packet< MessageT >::operator[] (
            size_t index ) const  [inline]
```

```
{ return messages[index]; }
```

6.20.3 Member Data Documentation

6.20.3.1 messages

```
template<typename MessageT >
std::vector<MessageT> ats::instrument_message_packet< MessageT >::messages
```

Referenced by ats::fix_to_csv(), ats::level2_execution_engine::on_order_book_changed(), ats::parse_fix_msg(), ats::security_base::process_message(), and ats::l2_message_reader::read().

The documentation for this struct was generated from the following file:

- message/message.hpp

6.21 ats::l2_message_reader Class Reference

```
#include <level2_message_reader.hpp>
```

Inheritance diagram for ats::l2_message_reader:

Collaboration diagram for ats::l2_message_reader:

Public Member Functions

- l2_message_reader (const std::string &filename, const std::string &symbol, const std::string &exchange)
- virtual bool read () override

Additional Inherited Members

6.21.1 Constructor & Destructor Documentation

6.21.1.1 l2_message_reader()

```
ats::l2_message_reader::l2_message_reader (
            const std::string & filename,
            const std::string & symbol,
            const std::string & exchange )  [inline]
```

References ats::instrument_message::exchange, ats::exchange_message_reader_base< ats::level2_message_packet >::message_, and ats::instrument_message::symbol.

```
            : ats::exchange_message_reader_base<ats::level2_message_packet>(),
            stream_(filename)
        {
                message_.symbol = symbol;
                message_.exchange = exchange;
            }
```

6.21.2 Member Function Documentation

6.21.2.1 read()

```
virtual bool ats::l2_message_reader::read ( )  [inline], [override], [virtual]
```

Implements ats::message_reader.

References ats::Ask, ats::Bid, ats::Change, ats::Delete, ats::level2_message::entry_type, ats::instrument_message::exchange, ats::level2_message::level, ats::exchange_message_reader_base< ats::level2_message_packet >::message_, ats::instrument_message_packet< MessageT >::messages, ats::New, ats::level2_message::order_count, ats::date_time::date_time::parse(), ats::level2_message::price, ats::level2_message::quantity, ats::instrument_message::symbol, ats::message::time, ats::tokenize(), ats::Trade, and ats::level2_message::update_action.

```
        {
            message_.messages.clear();
            ats::level2_message msg;

            bool is_message = false;
            while (std::getline(stream_, line_))
            {
                // Windows uses CRLF new lines as opposed to Unix's LF:
/*              if (line_.length() > 0 && *line_.crbegin() == '\r')
                    line_.erase(line_.length() - 1);*/

                if (ats::tokenize(line_, fields_, ',') != fields_.size())
                    break;

                if (!is_message)
                {
                    message_.time.parse(fields_[0], "%Y%m%d %H%M%S%F");
                    msg.time = message_.time;
                    msg.symbol = message_.symbol;
                    msg.exchange = message_.exchange;
                    is_message = true;
                }

                if (fields_[1] == "N")
                    msg.update_action = ats::update_action::New;
                else if (fields_[1] == "C")
                    msg.update_action = ats::update_action::Change;
                else if (fields_[1] == "D")
                    msg.update_action = ats::update_action::Delete;

                if (fields_[2] == "B")
                    msg.entry_type = ats::entry_type::Bid;
                else if (fields_[2] == "A")
                    msg.entry_type = ats::entry_type::Ask;
                else if (fields_[2] == "T")
                    msg.entry_type = ats::entry_type::Trade;

                msg.level = std::stol(fields_[3]); //std::strtol(fields_[3].c_str(), nullptr, 10);
                msg.price = std::stol(fields_[4]);//std::strtol(fields_[4].c_str(), nullptr, 10);
                msg.quantity = std::stol(fields_[5]);//std::strtol(fields_[5].c_str(), nullptr, 10);
                msg.order_count = std::stol(fields_[6]);//std::strtol(fields_[6].c_str(), nullptr, 10);

                message_.messages.push_back(msg);

                //std::cout << msg.time << ',' << msg.price << ',' << msg.quantity << ',' << msg.order_count << '\n';
            }

            return is_message;
        }
```

The documentation for this class was generated from the following file:

- custom_message_readers/level2_message_reader.hpp

6.22 ats::level2_exchange_message_reader Class Reference

`#include <level2_exchange_message_reader.hpp>`

Inheritance diagram for ats::level2_exchange_message_reader:

Collaboration diagram for ats::level2_exchange_message_reader:

Public Member Functions

- level2_exchange_message_reader (const std::string &filename)
- virtual bool read () override

Additional Inherited Members

6.22.1 Constructor & Destructor Documentation

6.22.1.1 level2_exchange_message_reader()

```
ats::level2_exchange_message_reader::level2_exchange_message_reader (
              const std::string & filename )  [inline]
```

```
    : ats::exchange_message_reader_base<ats::level2_message>(), stream_(filename), reader_(&stream_) { }
```

6.22.2 Member Function Documentation

6.22.2.1 read()

`virtual bool ats::level2_exchange_message_reader::read ( ) [inline], [override], [virtual]`

Implements ats::message_reader.

References ats::level2_message::entry_type, ats::instrument_message::exchange, ats::level2_message::level, ats::exchange_message_reader_base< ats::level2_message >::message_, ats::level2_message::order_count, ats::date_time::date_time::parse(), ats::level2_message::price, ats::level2_message::quantity, ats::csv_reader< num_columns >::read(), ats::level2_message::seq_number, ats::instrument_message::symbol, ats::message::time, and ats::level2_message::update_action.

```
        {
            if (reader_.read(fields_))
            {
                message_.symbol = fields_[0];
                message_.exchange = fields_[1];
                message_.seq_number = std::strtol(fields_[2].c_str(), nullptr, 10);
                message_.time.parse(fields_[3], "%Y%m%d %H%M%S%F"); //= 0; //
    Time!!!!!!!!!!!!!!
                message_.update_action = static_cast<
   ats::update_action>(std::strtol(fields_[4].c_str(), nullptr, 10));
                message_.entry_type = static_cast<
   ats::entry_type>(std::strtol(fields_[5].c_str(), nullptr, 10));
                message_.price = std::strtol(fields_[6].c_str(), nullptr, 10);
                message_.quantity = std::strtol(fields_[7].c_str(), nullptr, 10);
                message_.order_count = std::strtol(fields_[8].c_str(), nullptr, 10);
                message_.level = std::strtol(fields_[9].c_str(), nullptr, 10);
                return true;
            }
            else
                return false;
        }
```

The documentation for this class was generated from the following file:

- custom_message_readers/level2_exchange_message_reader.hpp

6.23 ats::level2_execution_engine Class Reference

`#include <level2_execution_engine.hpp>`

Inheritance diagram for ats::level2_execution_engine:

Collaboration diagram for ats::level2_execution_engine:

Public Member Functions

- level2_execution_engine (const std::string &name, size_t book_depth=10U)
- void add_order_book_changed_listener (const ats::order_book_changed_handler &handler)
- void subscribe (const ats::symbol_key &symbol)
- void on_order_book_changed (const ats::level2_message_packet &msg)
- const ats::timestamp_t & current_time () const
- virtual void send_order (const ats::market_order &order) override
- virtual void send_order (const ats::limit_order &order) override
- virtual void send_order (const ats::stop_order &order) override
- void cancel_order (const ats::limit_order &order)
- virtual void cancel_order (const ats::orderid_t &order_id) override

6.23.1 Constructor & Destructor Documentation

6.23.1.1 level2_execution_engine()

```
ats::level2_execution_engine::level2_execution_engine (
            const std::string & name,
            size_t book_depth = 10U )  [inline]
```

References ats::multievent_handler::add_event_handler(), and on_order_book_changed().

```
            : ats::execution_engine(name,
     ats::subscription::Level2), book_depth_(book_depth)
        {
            add_event_handler(&
     level2_execution_engine::on_order_book_changed, this);
        }
```

6.23.2 Member Function Documentation

6.23.2.1 add_order_book_changed_listener()

```
void ats::level2_execution_engine::add_order_book_changed_listener (
            const ats::order_book_changed_handler & handler )  [inline]
```

Referenced by ats::portfolio_base::add_connection().

```
        {
            order_book_changed_handler_ = handler;
        }
```

6.23.2.2 cancel_order() [1/2]

```
void ats::level2_execution_engine::cancel_order (
            const ats::limit_order & order )
```

Referenced by current_time().

6.23.2.3 cancel_order() [2/2]

```
virtual void ats::level2_execution_engine::cancel_order (
            const ats::orderid_t & order_id )  [inline], [override], [virtual]
```

Implements ats::execution_engine.

References ats::Buy, ats::BuyCover, current_time(), ats::execution_engine::on_order_status_changed(), ats::order::side(), and ats::symbol().

```
    {
        auto it = orders_.find(order_id);
        if (it != orders_.cend())
        {
            auto& order = it->second;
            std::type_index order_type(typeid(*order.get()));
            if (order_type == typeid(ats::limit_order))
            {
                auto book_it = sim_books_.find(order->symbol());
                book_it->second.cancel_order(order_id, current_time());
            }
            else if (order_type == typeid(ats::stop_order))
            {
                ats::price_t price = std::static_pointer_cast<ats::stop_order>(order)->price();
                if (order->side() == ats::order_side::Buy || order->side() == ats::order_side::BuyCover)
                {
                    auto range = stops_buy_.equal_range(price);
                    stops_buy_.erase(range.first, range.second);
                }
                else
                {
                    auto range = stops_sell_.equal_range(price);
                    stops_sell_.erase(range.first, range.second);
                }
            }

            orders_.erase(it);
            // For a limit order, on_order_status_changed will be called from inside the fifo_exchange_order_book
            if (order_type == typeid(ats::stop_order))
                on_order_status_changed(ats::order_status_cancelled_message(order_id, current_time()));
        }
        else
            std::cout << "ERROR (level2_execution_engine): Cannot cancel order id=" << order_id << '\n';
    }
```

6.23.2.4 current_time()

```
const ats::timestamp_t& ats::level2_execution_engine::current_time ( ) const  [inline]
```

References cancel_order(), and send_order().

Referenced by cancel_order(), on_order_book_changed(), and send_order().

```
{ return time_; }
```

6.23.2.5 on_order_book_changed()

```
void ats::level2_execution_engine::on_order_book_changed (
            const ats::level2_message_packet & msg )  [inline]
```

References ats::exchange_order_book::asks(), ats::exchange_order_book::best_ask(), ats::exchange_order_book::best_bid(), ats::exchange_order_book::bids(), current_time(), ats::order_book_detail::price_levels< compare >::empty(), ats::order::id(), ats::instrument_message_packet< MessageT >::messages, ats::execution_engine::on_order_status_changed(), ats::order_book_detail::price_level::price, ats::order::quantity(), ats::instrument_message::symbol, and ats::message::time.

Referenced by level2_execution_engine().

```
    {
        time_ = msg.time;

        auto book_it = sim_books_.find(msg.symbol);
        if (book_it == sim_books_.cend()) return;

        for (const auto& m : msg.messages)
            book_it->second.update(m);

        // Check if stop orders must be executed
        const ats::exchange_order_book& book = book_it->second.get_order_book()
;
        if (!stops_buy_.empty() && !book.asks().empty())
        {
            ats::price_t ask = book.best_ask()->price;
            for (auto it = stops_buy_.begin(); it != stops_buy_.end() && it->second.price() <= ask;)
            {
                const ats::stop_order& order = it->second;
                on_order_status_changed(
ats::order_status_filled_message(order.id(),
current_time(), ask, order.quantity()));
                stops_buy_.erase(it++);
            }
        }
        else if (!stops_sell_.empty() && !book.bids().empty())
        {
            ats::price_t bid = book.best_bid()->price;
            for (auto it = stops_sell_.begin(); it != stops_sell_.end() && it->second.price() > bid;)
            {
                const ats::stop_order& order = it->second;
                on_order_status_changed(
ats::order_status_filled_message(order.id(),
current_time(), bid, order.quantity()));
                stops_sell_.erase(it++);
            }
        }

        if (order_book_changed_handler_ != nullptr)
            order_book_changed_handler_(msg);
    }
```

6.23.2.6 send_order() [1/3]

```
void ats::level2_execution_engine::send_order (
            const ats::market_order & order )  [override], [virtual]
```

Implements ats::execution_engine.

References ats::exchange_order_book::asks(), ats::exchange_order_book::best_ask(), ats::exchange_order_book::best_bid(), ats::exchange_order_book::bids(), ats::Buy, ats::BuyCover, current_time(), ats::order_book_detail::price_levels< compare >::empty(), ats::market_order::fill_price, ats::order::id(), ats::execution_engine::on_order_status_changed(), ats::order_book_detail::price_level::price, ats::order::quantity(), ats::order::side(), and ats::order::symbol().

Referenced by current_time().

```
 {
     auto it = sim_books_.find(order.symbol());
     const ats::exchange_order_book& book = it->second.get_order_book();

     on_order_status_changed(
ats::order_status_pending_new_message(order.
id(), current_time()));

     if (order.fill_price != 0)
     {
         on_order_status_changed(
ats::order_status_filled_message(order.id(),
current_time(), order.fill_price, order.quantity()));
         return;
     }

     if (order.side() == ats::order_side::Buy || order.
side() == ats::order_side::BuyCover)
     {
         if (!book.asks().empty())
         {
             ats::price_t price = book.best_ask()->price;
             on_order_status_changed(
ats::order_status_filled_message(order.id(),
current_time(), price, order.quantity()));
         }
         else
             on_order_status_changed(
ats::order_status_rejected_message(order.id(),
current_time(),
                     "level2_execution_engine: No asks in order book"));
     }
     else
     {
         if (!book.bids().empty())
         {
             ats::price_t price = book.best_bid()->price;
             on_order_status_changed(
ats::order_status_filled_message(order.id(),
current_time(), price, order.quantity()));
         }
         else
             on_order_status_changed(
ats::order_status_rejected_message(order.id(),
current_time(),
                     "level2_execution_engine: No bids in order book"));
     }
 }
```

6.23.2.7 send_order() [2/3]

```
void ats::level2_execution_engine::send_order (
             const ats::limit_order & order )   [override], [virtual]
```

Implements ats::execution_engine.

References ats::order::symbol().

```
{
    auto it = sim_books_.find(order.symbol());
    if (it != sim_books_.cend())
    {
         add_order(order);
         it->second.add_order(order);
     }
 }
```

6.23.2.8 send_order() [3/3]

```
void ats::level2_execution_engine::send_order (
            const ats::stop_order & order )  [override], [virtual]
```

Implements ats::execution_engine.

References ats::exchange_order_book::asks(), ats::exchange_order_book::best_ask(), ats::exchange_order_book::best_bid(), ats::exchange_order_book::bids(), ats::Buy, ats::BuyCover, current_time(), ats::order_book_detail::price_levels< compare >::empty(), ats::order::id(), ats::execution_engine::on_order_status_changed(), ats::order_book_detail::price_level::price(), ats::stop_order::price(), ats::order::quantity(), ats::order::side(), and ats::order::symbol().

```
    {
        auto it = sim_books_.find(order.symbol());
        const ats::exchange_order_book& book = it->second.get_order_book();

        on_order_status_changed(ats::order_status_pending_new_message(order.id(), current_time()));

        if (order.side() == ats::order_side::Buy || order.side() == ats::order_side::BuyCover)
        {
            if (!book.asks().empty() && order.price() <= book.best_ask()->price)
                on_order_status_changed(ats::order_status_filled_message(order.id(), current_time(), book.best_ask()->price, order.quantity()));
            else
            {
                stops_buy_.insert(std::make_pair(order.price(), order));
                add_order(order);
//              orders_.insert(std::make_pair(order.id(), std::make_shared<ats::stop_order>(order)));
            }
        }
        else
        {
            if (!book.bids().empty() && order.price() >= book.best_bid()->price)
                on_order_status_changed(ats::order_status_filled_message(order.id(), current_time(), book.best_bid()->price, order.quantity()));
            else
            {
                stops_sell_.insert(std::make_pair(order.price(), order));
                add_order(order);
//              orders_.insert(std::make_pair(order.id(), std::make_shared<ats::stop_order>(order)));
            }
        }
    }
```

6.23.2.9 subscribe()

```
void ats::level2_execution_engine::subscribe (
            const ats::symbol_key & symbol )  [inline], [virtual]
```

Implements ats::execution_engine.

References ats::sim::fifo_exchange_order_book::add_order_status_listener(), ats::execution_engine::name(), ats::execution_engine::on_order_status_changed(), and ats::symbol_key::to_string().

```
    {
        auto it = sim_books_.find(symbol.to_string());
        if (it == sim_books_.cend())
        {
            ats::sim::fifo_exchange_order_book sim_book(symbol,
name(), book_depth_);
            sim_book.add_order_status_listener(std::bind(&
ats::level2_execution_engine::on_order_status_changed,
 this, std::placeholders::_1));
            sim_books_.insert(std::make_pair(symbol.to_string(), std::move(sim_book)));
        }
        else
        {
            std::string text = "level2_execution_engine: Symbol '" + symbol.
to_string() + "' already exists";
            throw std::invalid_argument(text);
        }
    }
```

The documentation for this class was generated from the following files:

- execution_engine/level2/level2_execution_engine.hpp
- execution_engine/level2/level2_execution_engine.cpp

6.24 ats::level2_historical_data_feed Class Reference

`#include <level2_historical_data_feed.hpp>`

Inheritance diagram for ats::level2_historical_data_feed:

Collaboration diagram for ats::level2_historical_data_feed:

Public Member Functions

- level2_historical_data_feed (ats::portfolio_base ∗universe)

Additional Inherited Members

6.24.1 Constructor & Destructor Documentation

6.24.1.1 level2_historical_data_feed()

```
ats::level2_historical_data_feed::level2_historical_data_feed (
            ats::portfolio_base * universe )  [inline]
```

References ats::multievent_handler::add_event_handler().

```
        : ats::historical_data_feed(universe)
    {
        add_event_handler(&level2_historical_data_feed::handler, this);
    }
```

The documentation for this class was generated from the following file:

- custom_data_feeds/level2_historical_data_feed.hpp

6.25 ats::level2_message Struct Reference

```
#include <level2_message.hpp>
```

Inheritance diagram for ats::level2_message:

Collaboration diagram for ats::level2_message:

Public Attributes

- price_t price
- long quantity
- long order_count = 0
- size_t level
- size_t seq_number = 0
- ats::update_action update_action
- ats::entry_type entry_type
- ats::market_state state
- int aggressor_side = 0

Additional Inherited Members

6.25.1 Member Data Documentation

6.25.1.1 aggressor_side

```
int ats::level2_message::aggressor_side = 0
```

Referenced by ats::parse_fix_msg(), ats::sim::fifo_exchange_order_book::update(), and ats::exchange_order_book::update2().

6.25.1.2 entry_type

```
ats::entry_type ats::level2_message::entry_type
```

Referenced by ats::parse_fix_msg(), ats::sim::price_level::process_change_msg(), ats::sim::sim_book::process_change_msg(), ats::sim::sim_book::process_delete_msg(), ats::sim::sim_book::process_insert_msg(), ats::sim::price_levels< std::greater< ats::price_t > >::process_insert_msg(), ats::sim::sim_book::process_level2_msg(), ats::level2_exchange_message_reader::read(), ats::l2_message_reader::read(), ats::sim::fifo_exchange_order_book::update(), ats::exchange_order_book::update(), ats::order_book_detail::price_levels< std::greater< ats::price_t > >::update(), ats::exchange_order_book::update2(), and ats::order_book_detail::price_levels< std::greater< ats::price_t > >::update2().

6.25.1.3 level

```
size_t ats::level2_message::level
```

Referenced by ats::parse_fix_msg(), ats::level2_exchange_message_reader::read(), and ats::l2_message_reader::read().

6.25.1.4 order_count

```
long ats::level2_message::order_count = 0
```

Referenced by ats::parse_fix_msg(), ats::level2_exchange_message_reader::read(), ats::l2_message_reader::read(), ats::order_book_detail::price_levels< std::greater< ats::price_t > >::update(), and ats::order_book_detail::price_levels< std::greater< ats::price_t > >::update2().

6.25.1.5 price

```
price_t ats::level2_message::price
```

Referenced by ats::parse_fix_msg(), ats::sim::price_level::process_change_msg(), ats::sim::sim_book::process_change_msg(), ats::sim::price_levels< std::greater< ats::price_t > >::process_change_msg(), ats::sim::price_levels< std::greater< ats::price_t > >::process_delete_msg(), ats::sim::price_levels< std::greater< ats::price_t > >::process_insert_msg(), ats::sim::sim_book::process_trade_msg(), ats::level2_exchange_message_reader::read(), ats::l2_message_reader::read(), ats::sim::fifo_exchange_order_book::update(), ats::order_book_detail::price_levels< std::greater< ats::price_t > >::update(), ats::exchange_order_book::update2(), and ats::order_book_detail::price_levels< std::greater< ats::price_t > >::update2().

6.25.1.6 quantity

```
long ats::level2_message::quantity
```

Referenced by ats::parse_fix_msg(), ats::sim::price_level::process_change_msg(), ats::sim::price_levels< std::greater< ats::price_t > >::process_delete_msg(), ats::sim::price_levels< std::greater< ats::price_t > >::process_insert_msg(), ats::sim::sim_book::process_trade_msg(), ats::level2_exchange_message_reader::read(), ats::l2_message_reader::read(), ats::sim::fifo_exchange_order_book::update(), ats::order_book_detail::price_levels< std::greater< ats::price_t > >::update(), and ats::order_book_detail::price_levels< std::greater< ats::price_t > >::update2().

6.25.1.7 seq_number

```
size_t ats::level2_message::seq_number = 0
```

Referenced by ats::parse_fix_msg(), and ats::level2_exchange_message_reader::read().

6.25.1.8 state

```
ats::market_state ats::level2_message::state
```

6.25.1.9 update_action

```
ats::update_action ats::level2_message::update_action
```

Referenced by ats::parse_fix_msg(), ats::sim::sim_book::process_level2_msg(), ats::level2_exchange_message↩_reader::read(), ats::l2_message_reader::read(), ats::sim::fifo_exchange_order_book::update(), ats::order_book↩_detail::price_levels< std::greater< ats::price_t > >::update(), and ats::order_book_detail::price_levels< std↩::greater< ats::price_t > >::update2().

The documentation for this struct was generated from the following file:

- message/level2_message.hpp

6.26 ats::limit_order Class Reference

```
#include <limit_order.hpp>
```

Inheritance diagram for ats::limit_order:

Collaboration diagram for ats::limit_order:

Public Member Functions

- limit_order (const ats::orderid_t &id, const std::string &symbol, long quantity, ats::order_side side, ats↩::order_time_in_force time_in_force, ats::price_t price)
- ats::price_t price () const

Additional Inherited Members

6.26.1 Constructor & Destructor Documentation

6.26.1.1 limit_order()

```
ats::limit_order::limit_order (
            const ats::orderid_t & id,
            const std::string & symbol,
            long quantity,
            ats::order_side side,
            ats::order_time_in_force time_in_force,
            ats::price_t price )  [inline]
```

```
15              : ats::order(id, symbol, quantity, side,
     time_in_force), price_(price) { }
```

6.26.2 Member Function Documentation

6.26.2.1 price()

```
ats::price_t ats::limit_order::price ( ) const  [inline]
```

Referenced by ats::sim::sim_book::add_order(), ats::sim::fifo_exchange_order_book::add_order(), ats::sim::price_levels< std::greater< ats::price_t > >::add_order(), ats::limit_order_container1::add_order(), and ats::sim::price_levels< std::greater< ats::price_t > >::insert_order().

```
{ return price_; }
```

The documentation for this class was generated from the following file:

- order/limit_order.hpp

6.27 ats::limit_order_container Class Reference

```
#include <limit_order_container.hpp>
```

Collaboration diagram for ats::limit_order_container:

Public Member Functions

- limit_order_container & operator+= (const order_ptr &order)
- limit_order_container & operator+= (order_ptr &&order)
- limit_order_container & operator+= (const ats::limit_order &order)
- void delete_order (iterator it)
- void delete_order (const ats::orderid_t &id)
- iterator get_order (const ats::orderid_t &id)
- order_ptr get_order_ptr (const ats::orderid_t &id)
- std::pair< iterator, iterator > get_orders (ats::price_t price)
- void print_all_orders () const
- void print_all_orders (ats::price_t price)
- iterator begin ()
- const_iterator cbegin () const
- iterator end ()
- const_iterator cend () const

6.27.1 Member Function Documentation

6.27.1.1 begin()

```
iterator ats::limit_order_container::begin ( )  [inline]
```

```
{ return orders_.begin(); }
```

6.27.1.2 cbegin()

```
const_iterator ats::limit_order_container::cbegin ( ) const  [inline]
```

```
{ return orders_.cbegin(); }
```

6.27.1.3 cend()

```
const_iterator ats::limit_order_container::cend ( ) const  [inline]
```

```
{ return orders_.cend(); }
```

6.27.1.4 delete_order() [1/2]

```
void ats::limit_order_container::delete_order (
             iterator it )  [inline]
```

```
        {
            auto find = iterators_.find(it->second->id());
            iterators_.erase(find);
            orders_.erase(it);
        }
```

6.27.1.5 delete_order() [2/2]

```
void ats::limit_order_container::delete_order (
             const ats::orderid_t & id )  [inline]
```

```
        {
            auto find = iterators_.find(id);
            if (find != iterators_.end())
            {
                auto it = find->second;
                iterators_.erase(find);
                orders_.erase(it);
            }
        }
```

6.27.1.6 end()

```
iterator ats::limit_order_container::end ( )  [inline]
```

```
{ return orders_.end(); }
```

6.27.1.7 get_order()

```
iterator ats::limit_order_container::get_order (
             const ats::orderid_t & id )  [inline]
```

```
        {
            auto find = iterators_.find(id);
            return find != iterators_.end() ? find->second : orders_.end();
        }
```

6.27.1.8 get_order_ptr()

```
order_ptr ats::limit_order_container::get_order_ptr (
             const ats::orderid_t & id )  [inline]
```

```
        {
            auto find = iterators_.find(id);
            return find != iterators_.end() ? find->second->second : nullptr;
        }
```

6.27.1.9 get_orders()

```
std::pair<iterator, iterator> ats::limit_order_container::get_orders (
             ats::price_t price )  [inline]
```

Referenced by print_all_orders().

```
        {
            return orders_.equal_range(price);
        }
```

6.27.1.10 operator+=() [1/3]

```
limit_order_container& ats::limit_order_container::operator+= (
            const order_ptr & order )  [inline]
```

Referenced by operator+=().

```
{
    auto it = orders_.insert(std::make_pair(order->price(), order));
    iterators_.insert(std::make_pair(order->id(), it));
    return *this;
}
```

6.27.1.11 operator+=() [2/3]

```
limit_order_container& ats::limit_order_container::operator+= (
            order_ptr && order )  [inline]
```

```
{
    auto it = orders_.insert(std::make_pair(order->price(), std::move(order)));
    iterators_.insert(std::make_pair(it->second->id(), it));
    return *this;
}
```

6.27.1.12 operator+=() [3/3]

```
limit_order_container& ats::limit_order_container::operator+= (
            const ats::limit_order & order )  [inline]
```

References operator+=().

```
{
    return this->operator +=(std::make_shared<ats::limit_order>(order));
}
```

6.27.1.13 print_all_orders() [1/2]

```
void ats::limit_order_container::print_all_orders ( ) const  [inline]
```

```
{
    for (auto it = orders_.cbegin(); it != orders_.cend(); ++it)
    {
        std::cout << "Order @" << it->second->price() << " id=" << it->second->id() << '\n';
    }
}
```

6.27.1.14 print_all_orders() [2/2]

```
void ats::limit_order_container::print_all_orders (
              ats::price_t price )  [inline]
```

References get_orders().

```
    {
        auto range = get_orders(price);
        for (auto it = range.first; it != range.second; ++it)
        {
            std::cout << "Order @" << it->second->price() << " id=" << it->second->id() << '\n';
        }
    }
```

The documentation for this class was generated from the following file:

- container/limit_order_container.hpp

6.28 ats::limit_order_container1 Class Reference

```
#include <limit_order_container.hpp>
```

Collaboration diagram for ats::limit_order_container1:

Public Member Functions

- void add_order (const ats::limit_order &order)
- void delete_order (iterator it)
- void delete_order (const ats::orderid_t &id)
- iterator get_order (const ats::orderid_t &id)
- iterator_container_type ∗ get_orders (ats::price_t price)
- void print_all_orders () const
- void print_all_orders (ats::price_t price)
- iterator begin ()
- iterator end ()

6.28.1 Member Function Documentation

6.28.1.1 add_order()

```
void ats::limit_order_container1::add_order (
              const ats::limit_order & order )  [inline]
```

References ats::order::id(), and ats::limit_order::price().

```
    {
        auto it = orders_.insert(std::make_pair(order.id(), order)).first;
        auto find = iterators_.find(order.price());
        if (find != iterators_.end())
            find->second.insert(it);
        else
            iterators_.insert(std::make_pair(order.price(), iterator_container_type{ it }));
    }
```

6.28.1.2 begin()

```
iterator ats::limit_order_container1::begin ( )  [inline]
```

```
{ return orders_.begin(); }
```

6.28.1.3 delete_order() [1/2]

```
void ats::limit_order_container1::delete_order (
               iterator it )  [inline]
```

```
    {
        auto& level = iterators_.find(it->second.id())->second;
        level.erase(it);
        orders_.erase(it);
    }
```

6.28.1.4 delete_order() [2/2]

```
void ats::limit_order_container1::delete_order (
               const ats::orderid_t & id )  [inline]
```

```
    {
        auto it = orders_.find(id);
        if (it != orders_.end())
        {
            auto& level = iterators_.find(it->second.id())->second;
            level.erase(level.find(it));
            orders_.erase(it);
            //delete_order(it);
        }
    }
```

6.28.1.5 end()

```
iterator ats::limit_order_container1::end ( )  [inline]
```

```
{ return orders_.end(); }
```

6.28.1.6 get_order()

```
iterator ats::limit_order_container1::get_order (
            const ats::orderid_t & id )  [inline]
```

```
    {
        return orders_.find(id);
        //          return find != orders_.end() ? &find->second : nullptr;
    }
```

6.28.1.7 get_orders()

```
iterator_container_type* ats::limit_order_container1::get_orders (
            ats::price_t price )  [inline]
```

```
    {
        auto find = iterators_.find(price);
        return find != iterators_.end() ? &find->second : nullptr;
    }
```

6.28.1.8 print_all_orders() [1/2]

```
void ats::limit_order_container1::print_all_orders ( ) const  [inline]
```

```
    {
        for (auto it = orders_.cbegin(); it != orders_.cend(); ++it)
        {
            std::cout << "Order @" << it->second.price() << " id=" << it->second.id() << '\n';
        }
    }
```

6.28.1.9 print_all_orders() [2/2]

```
void ats::limit_order_container1::print_all_orders (
            ats::price_t price )  [inline]
```

```
    {
        auto find = iterators_.find(price);
        if (find == iterators_.end()) return;

        for (auto it = find->second.cbegin(); it != find->second.cend(); ++it)
        {
            //std::cout << "Order @" << (*it)->second->price() << " id=" << (*it)->second->id() << '\n';
        }
    }
```

The documentation for this class was generated from the following file:

- container/limit_order_container.hpp

6.29 ats::market_order Class Reference

```
#include <market_order.hpp>
```

Inheritance diagram for ats::market_order:

Collaboration diagram for ats::market_order:

Public Member Functions

- market_order (const ats::orderid_t &id, const std::string &symbol, long quantity, ats::order_side side, ats::order_time_in_force time_in_force)

Public Attributes

- ats::price_t fill_price

6.29.1 Constructor & Destructor Documentation

6.29.1.1 market_order()

```
ats::market_order::market_order (
            const ats::orderid_t & id,
            const std::string & symbol,
            long quantity,
            ats::order_side side,
            ats::order_time_in_force time_in_force )   [inline]
```

```
15           : ats::order(id, symbol, quantity, side,
    time_in_force), fill_price(0) { }
```

6.29.2 Member Data Documentation

6.29.2.1 fill_price

```
ats::price_t ats::market_order::fill_price
```

Referenced by ats::level2_execution_engine::send_order().

The documentation for this class was generated from the following file:

- order/market_order.hpp

6.30 ats::detail::member_function_handler< Object, EventArgsT > Class Template Reference

```
#include <multievent_memfunc_handler.hpp>
```

Inheritance diagram for ats::detail::member_function_handler< Object, EventArgsT >:

Collaboration diagram for ats::detail::member_function_handler< Object, EventArgsT >:

Public Types

- typedef void(Object::∗ member_function_type) (const EventArgsT &)

Public Member Functions

- member_function_handler (member_function_type memfunc, Object ∗instance)
- virtual void operator() (const basic_message &e) override

6.30.1 Member Typedef Documentation

6.30.1.1 member_function_type

```
template<class Object , typename EventArgsT >
typedef void(Object::* ats::detail::member_function_handler< Object, EventArgsT >::member_←-
function_type) (const EventArgsT &)
```

6.30.2 Constructor & Destructor Documentation

6.30.2.1 member_function_handler()

```
template<class Object , typename EventArgsT >
ats::detail::member_function_handler< Object, EventArgsT >::member_function_handler (
            member_function_type memfunc,
            Object * instance ) [inline]
```

```
29                  : instance_(instance), memfunc_(memfunc) { }
```

6.30.3 Member Function Documentation

6.30.3.1 operator()()

```
template<class Object , typename EventArgsT >
virtual void ats::detail::member_function_handler< Object, EventArgsT >::operator() (
             const basic_message & e )  [inline], [override], [virtual]
```

Implements ats::detail::member_function_handler_base.

```
            {
                (instance_->*memfunc_)(static_cast<const EventArgsT&>(e));
            }
```

The documentation for this class was generated from the following file:

- event_handler/multievent_memfunc_handler.hpp

6.31 ats::detail::member_function_handler_base Class Reference

```
#include <multievent_memfunc_handler.hpp>
```

Inheritance diagram for ats::detail::member_function_handler_base:

Collaboration diagram for ats::detail::member_function_handler_base:

Public Member Functions

- virtual ∼member_function_handler_base ()
- virtual void operator() (const basic_message &e)=0

6.31.1 Constructor & Destructor Documentation

6.31.1.1 ∼member_function_handler_base()

```
virtual ats::detail::member_function_handler_base::∼member_function_handler_base ( )  [inline],
[virtual]
```

References operator()().

```
{ }
```

6.31.2 Member Function Documentation

6.31.2.1 operator()()

```
virtual void ats::detail::member_function_handler_base::operator() (
             const basic_message & e )  [pure virtual]
```

Implemented in ats::detail::member_function_handler< Object, EventArgsT >.

Referenced by ∼member_function_handler_base().

The documentation for this class was generated from the following file:

- event_handler/multievent_memfunc_handler.hpp

6.32 ats::message Struct Reference

```
#include <message.hpp>
```

Inheritance diagram for ats::message:

Collaboration diagram for ats::message:

Public Member Functions

- virtual ∼message ()
- message ()=default
- message (const ats::timestamp_t &time)
- message (ats::timestamp_t &&time)

Public Attributes

- ats::timestamp_t time

6.32.1 Constructor & Destructor Documentation

6.32.1.1 ∼message()

```
virtual ats::message::∼message ( )  [inline], [virtual]
```

References message().

```
15 { }
```

6.32.1.2 message() [1/3]

```
ats::message::message ( )  [default]
```

Referenced by ∼message().

6.32.1.3 message() [2/3]

```
ats::message::message (
             const ats::timestamp_t & time )  [inline]
```

```
: time(time) { }
```

6.32.1.4 message() [3/3]

```
ats::message::message (
             ats::timestamp_t && time )  [inline]
```

```
: time(std::move(time)) { }
```

6.32.2 Member Data Documentation

6.32.2.1 time

```
ats::timestamp_t ats::message::time
```

Referenced by ats::level2_execution_engine::on_order_book_changed(), ats::parse_fix_msg(), ats::sim::price_level::process_change_msg(), ats::sim::sim_book::process_change_msg(), ats::sim::price_levels< std::greater< ats::price_t > >::process_delete_msg(), ats::security_base::process_message(), ats::portfolio_base::process_message(), ats::portfolio_base::process_order_status_message(), ats::sim::sim_book::process_trade_msg(), ats::level2_exchange_message_reader::read(), ats::l2_message_reader::read(), ats::historical_data_feed::read(), ats::sim::fifo_exchange_order_book::update(), ats::exchange_order_book::update(), and ats::exchange_order_book::update2().

The documentation for this struct was generated from the following file:

- message/message.hpp

6.33 ats::message_reader Class Reference

```
#include <message_reader.hpp>
```

Inheritance diagram for ats::message_reader:

Collaboration diagram for ats::message_reader:

Public Member Functions

- virtual ∼message_reader ()
- virtual bool read ()=0
- virtual void send_message (ats::portfolio_base ∗universe) const =0
- template<typename CallableT >
  void send (CallableT &f)
- virtual const ats::message & get_last_message () const =0

6.33.1 Constructor & Destructor Documentation

6.33.1.1 ∼message_reader()

```
virtual ats::message_reader::∼message_reader ( )  [inline], [virtual]
```

References read(), and send_message().

```
12 { }
```

6.33.2 Member Function Documentation

6.33.2.1 get_last_message()

```
virtual const ats::message& ats::message_reader::get_last_message ( ) const  [pure virtual]
```

Implemented in ats::single_message_reader< MessageT >, ats::exchange_message_reader_base< MessageT >, ats::exchange_message_reader_base< ats::level2_message >, and ats::exchange_message_reader_base< ats::level2_message_packet >.

Referenced by send().

6.33.2.2 read()

```
virtual bool ats::message_reader::read ( )  [pure virtual]
```

Implemented in ats::l2_message_reader, and ats::level2_exchange_message_reader.

Referenced by ∼message_reader().

6.33.2.3 send()

```
template<typename CallableT >
void ats::message_reader::send (
             CallableT & f )  [inline]
```

References get_last_message().

```
        {
            f(static_cast<decltype(get_last_message())>(
     get_last_message()));
        }
```

6.33.2.4 send_message()

```
virtual void ats::message_reader::send_message (
             ats::portfolio_base * universe ) const  [pure virtual]
```

Implemented in ats::single_message_reader< MessageT >, ats::exchange_message_reader_base< MessageT >, ats::exchange_message_reader_base< ats::level2_message >, and ats::exchange_message_reader_base< ats::level2_message_packet >.

Referenced by ∼message_reader().

The documentation for this class was generated from the following file:

- data_feed/historical/message_reader.hpp

6.34 ats::metainfo Struct Reference

```
#include <metainfo.hpp>
```

Collaboration diagram for ats::metainfo:

Public Member Functions

- metainfo (ats::price_t ticksize=10, double tickvalue=10.0)

Public Attributes

- ats::price_t tick_size = 10
- double tick_value = 10.0

6.34.1 Constructor & Destructor Documentation

6.34.1.1 metainfo()

```
ats::metainfo::metainfo (
            ats::price_t ticksize = 10,
            double tickvalue = 10.0 )  [inline]
```

```
14            : tick_size(ticksize), tick_value(tickvalue) { }
```

6.34.2 Member Data Documentation

6.34.2.1 tick_size

```
ats::price_t ats::metainfo::tick_size = 10
```

6.34.2.2 tick_value

```
double ats::metainfo::tick_value = 10.0
```

The documentation for this struct was generated from the following file:

- security/metainfo.hpp

6.35 ats::multievent_handler Class Reference

```
#include <multievent_handler.hpp>
```

Inheritance diagram for ats::multievent_handler:

Collaboration diagram for ats::multievent_handler:

Public Member Functions

- template<typename FuncT >
  void add_event_handler (FuncT &handler)
- template<class Object , typename... Args>
  void add_event_handler (void(Object::∗MemFuncPtr)(Args...), Object ∗instance)
- template<typename... Args>
  void invoke (Args &&... args)
- template<typename... Args>
  void operator() (Args &&... args)

6.35.1 Member Function Documentation

6.35.1.1 add_event_handler() [1/2]

```
template<typename FuncT >
void ats::multievent_handler::add_event_handler (
             FuncT & handler )  [inline]
```

Referenced by ats::execution_engine::execution_engine(), ats::level2_execution_engine::level2_execution_engine(), and ats::level2_historical_data_feed::level2_historical_data_feed().

```
        {
            // define a basic function of signature FuncT from the passed handler
            func_ptr f_ptr(new ats::detail::basic_function<FuncT>(
     std::function<FuncT>(handler)));

            // insert the function above into the hash map
            std::type_index index(typeid(FuncT));
            auto it = handlers_.find(index);
            if (it == handlers_.end())
                handlers_[index] = std::list<func_ptr>{ std::move(f_ptr) };
            else
                it->second.push_back(std::move(f_ptr));
        }
```

6.35.1.2 add_event_handler() [2/2]

```
template<class Object , typename... Args>
void ats::multievent_handler::add_event_handler (
             void(Object::∗)(Args...)  MemFuncPtr,
             Object ∗ instance )  [inline]
```

```
        {
            // define a basic function from the passed member function handler
            typedef void FuncT(Args...);
            func_ptr f_ptr(new ats::detail::basic_function<FuncT>(
                    [MemFuncPtr, instance](Args... args) { (instance->*MemFuncPtr)(args...); }
            ));

            // insert the function above into the hash map
            std::type_index index(typeid(FuncT));
            auto it = handlers_.find(index);
            if (it == handlers_.end())
                handlers_[index] = std::list<func_ptr>{ std::move(f_ptr) };
            else
                it->second.push_back(std::move(f_ptr));
        }
```

6.35.1.3 invoke()

```
template<typename... Args>
void ats::multievent_handler::invoke (
            Args &&... args ) [inline]
```

References ats::detail::basic_function< T >::function.

Referenced by ats::exchange_message_reader_base< ats::level2_message_packet >::send_message(), and ats::single_message_reader< MessageT >::send_message().

```
    {
        typedef void FuncT(Args...);
        auto it = handlers_.find(typeid(FuncT));
        if (it != handlers_.end())
        {
            for (const auto& f_ptr : it->second)
            {
                std::function<FuncT> func = static_cast<const ats::detail::basic_function<FuncT>&>(*f_ptr).function;
                func(std::forward<Args>(args)...);
            }
        }
    }
```

6.35.1.4 operator()()

```
template<typename... Args>
void ats::multievent_handler::operator() (
            Args &&... args ) [inline]
```

References ats::detail::basic_function< T >::function.

```
    {
        typedef void FuncT(Args...);
        auto it = handlers_.find(typeid(FuncT));
        if (it != handlers_.end())
        {
            for (const auto& f_ptr : it->second)
            {
                std::function<FuncT> func = static_cast<const ats::detail::basic_function<FuncT>&>(*f_ptr).function;
                func(std::forward<Args>(args)...);
            }
        }
    }
```

The documentation for this class was generated from the following file:

- event_handler/multievent_handler.hpp

6.36 ats::multievent_memfunc_handler Class Reference

```
#include <multievent_memfunc_handler.hpp>
```

Collaboration diagram for ats::multievent_memfunc_handler:

Public Member Functions

- void invoke (const basic_message &e)
- template<class Object , typename EventArgsT >
  void add_event_handler (void(Object::∗MemFuncPtr)(const EventArgsT &), Object ∗instance)

6.36.1 Member Function Documentation

6.36.1.1 add_event_handler()

```
template<class Object , typename EventArgsT >
void ats::multievent_memfunc_handler::add_event_handler (
            void(Object::∗)(const EventArgsT &) MemFuncPtr,
            Object ∗ instance )  [inline]
```

```
        {
            handlers_.insert(handler_container::value_type(typeid(EventArgsT),
                    func_ptr(new ats::detail::member_function_handler<Object, EventArgsT>(MemFuncPtr, instance))));
        }
```

6.36.1.2 invoke()

```
void ats::multievent_memfunc_handler::invoke (
            const basic_message & e )  [inline]
```

```
        {
            auto handler_range = handlers_.equal_range(typeid(e));
            for (auto it = handler_range.first; it != handler_range.second; ++it)
                it->second->operator()(e);
        }
```

The documentation for this class was generated from the following file:

- event_handler/multievent_memfunc_handler.hpp

6.37 ats::order Class Reference

```
#include <order.hpp>
```

Inheritance diagram for ats::order:

Collaboration diagram for ats::order:

Public Member Functions

- order (const ats::orderid_t &id, const std::string &symbol, long quantity, ats::order_side side, ats::order_time_in_force time_in_force)
- virtual ∼order ()
- const ats::orderid_t & id () const
- const std::string & symbol () const
- long quantity () const
- ats::order_side side () const
- ats::order_time_in_force time_in_force () const
- ats::order_status status () const
- void set_quantity (long new_quantity)
- void set_status (const ats::order_status &new_status)
- bool is_pending () const
- bool compare (const order &another_order) const
- bool compare (const std::type_index &another_order) const

Public Attributes

- long parent_id = -1
- std::string exchange = ""
- ats::timestamp_t transact_time
- long executed_quantity = 0

6.37.1 Constructor & Destructor Documentation

6.37.1.1 order()

```
ats::order::order (
            const ats::orderid_t & id,
            const std::string & symbol,
            long quantity,
            ats::order_side side,
            ats::order_time_in_force time_in_force )  [inline]
```

```
            : id_(id), symbol_(symbol), quantity_(quantity),
              side_(side), time_in_force_(time_in_force),
    transact_time(ats::timestamp_t::now())
        { }
```

6.37.1.2 ∼order()

```
virtual ats::order::∼order ( )  [inline], [virtual]
```

```
{ }
```

6.37.2 Member Function Documentation

6.37.2.1 compare() [1/2]

```
bool ats::order::compare (
             const order & another_order ) const  [inline]
```

```
        {
            std::type_index index(typeid(*this));
            return index == typeid(another_order);
        }
```

6.37.2.2 compare() [2/2]

```
bool ats::order::compare (
             const std::type_index & another_order ) const  [inline]
```

```
        {
            std::type_index index(typeid(*this));
            return index == another_order;
        }
```

6.37.2.3 id()

```
const ats::orderid_t& ats::order::id ( ) const  [inline]
```

Referenced by ats::sim::sim_book::add_order(), ats::sim::fifo_exchange_order_book::add_order(), ats::sim::price_level::add_order(), ats::limit_order_container1::add_order(), ats::bracket_order::bracket_order(), ats::sim::sim_book::insert_order(), ats::sim::price_level::insert_order(), ats::level2_execution_engine::on_order_book_changed(), ats::order_container1::operator+=(), ats::order_container::operator+=(), ats::level2_execution_engine::send_order(), and ats::portfolio_base::send_order().

```
{ return id_; }
```

6.37.2.4 is_pending()

```
bool ats::order::is_pending ( ) const  [inline]
```

References ats::Canceled, ats::Filled, and ats::Rejected.

```
        {
            return !(status_ == ats::order_status::Filled ||
                     status_ == ats::order_status::Canceled ||
                     status_ == ats::order_status::Rejected);
        }
```

6.37.2.5 quantity()

```
long ats::order::quantity ( ) const  [inline]
```

Referenced by ats::sim::sim_book::add_order(), ats::sim::fifo_exchange_order_book::add_order(), ats::sim::price_level::add_order(), ats::sim::price_level::insert_order(), ats::level2_execution_engine::on_order_book_changed(), and ats::level2_execution_engine::send_order().

```
28 { return quantity_; }
```

6.37.2.6 set_quantity()

```
void ats::order::set_quantity (
            long new_quantity )  [inline]
```

```
33 { quantity_ = new_quantity; }
```

6.37.2.7 set_status()

```
void ats::order::set_status (
            const ats::order_status & new_status )  [inline]
```

```
34 { status_ = new_status; }
```

6.37.2.8 side()

```
ats::order_side ats::order::side ( ) const  [inline]
```

Referenced by ats::sim::sim_book::add_order(), ats::sim::fifo_exchange_order_book::add_order(), ats::level2_execution_engine::cancel_order(), ats::sim::sim_book::insert_order(), and ats::level2_execution_engine::send_order().

```
29 { return side_; }
```

6.37.2.9 status()

```
ats::order_status ats::order::status ( ) const  [inline]
```

```
31 { return status_; }
```

6.37.2.10 symbol()

```
const std::string& ats::order::symbol ( ) const  [inline]
```

Referenced by ats::sim::sim_book::add_order(), ats::sim::fifo_exchange_order_book::add_order(), ats::level2_↩ execution_engine::cancel_order(), ats::order_container::operator+=(), ats::level2_execution_engine::send_order(), and ats::portfolio_base::send_order().

```
27 { return symbol_; }
```

6.37.2.11 time_in_force()

```
ats::order_time_in_force ats::order::time_in_force ( ) const  [inline]
```

Referenced by ats::sim::sim_book::add_order().

```
30 { return time_in_force_; }
```

6.37.3 Member Data Documentation

6.37.3.1 exchange

```
std::string ats::order::exchange = ""
```

Referenced by ats::order_container::operator+=(), and ats::portfolio_base::send_order().

6.37.3.2 executed_quantity

```
long ats::order::executed_quantity = 0
```

6.37.3.3 parent_id

```
long ats::order::parent_id = -1
```

Referenced by ats::bracket_order::bracket_order().

6.37.3.4 transact_time

```
ats::timestamp_t ats::order::transact_time
```

Referenced by ats::sim::sim_book::add_order(), and ats::sim::fifo_exchange_order_book::add_order().

The documentation for this class was generated from the following file:

- order/order.hpp

6.38 ats::order_book Class Reference

```
#include <order_book.hpp>
```

Collaboration diagram for ats::order_book:

Public Member Functions

- order_book (const ats::symbol_key &symbol)
- const ats::symbol_key & symbol () const
- void add_order_book (const std::string &exchange, size_t book_depth)
- ats::exchange_order_book ∗ get (const std::string &exchange)
- const ats::exchange_order_book ∗ get (const std::string &exchange) const
- void update (const ats::level2_message &msg)
- void update2 (ats::level2_message &msg)

6.38.1 Constructor & Destructor Documentation

6.38.1.1 order_book()

```
ats::order_book::order_book (
            const ats::symbol_key & symbol )  [inline]
```

Referenced by ats::portfolio_base::get_order_book().

```
14 : symbol_(symbol) { }
```

6.38.2 Member Function Documentation

6.38.2.1 add_order_book()

```
void ats::order_book::add_order_book (
            const std::string & exchange,
            size_t book_depth )  [inline]
```

Referenced by ats::security_base::create_order_book().

```
{
    orderbooks_.insert(std::make_pair(exchange, ats::exchange_order_book(
symbol_, exchange, book_depth)));
}
```

6.38.2.2 get() [1/2]

```
ats::exchange_order_book* ats::order_book::get (
            const std::string & exchange )  [inline]
```

Referenced by ats::security_base::exchange_order_book().

```
{
    auto it = orderbooks_.find(exchange);
    return it != orderbooks_.cend() ? &it->second : nullptr;
}
```

6.38.2.3 get() [2/2]

```
const ats::exchange_order_book* ats::order_book::get (
            const std::string & exchange ) const  [inline]
```

```
{
    auto it = orderbooks_.find(exchange);
    return it != orderbooks_.cend() ? &it->second : nullptr;
}
```

6.38.2.4 symbol()

```
const ats::symbol_key& ats::order_book::symbol ( ) const  [inline]
```

```
{ return symbol_; }
```

6.38.2.5 update()

```
void ats::order_book::update (
            const ats::level2_message & msg ) [inline]
```

References ats::instrument_message::exchange, and ats::exchange_order_book::update().

```
        {
            ats::exchange_order_book* book = get(msg.exchange);
            if (book != nullptr)
                book->update(msg);
        }
```

6.38.2.6 update2()

```
void ats::order_book::update2 (
            ats::level2_message & msg ) [inline]
```

References ats::instrument_message::exchange, and ats::exchange_order_book::update2().

Referenced by ats::security_base::process_message().

```
        {
            ats::exchange_order_book* book = get(msg.exchange);
            if (book != nullptr)
                book->update2(msg);
        }
```

The documentation for this class was generated from the following file:

- order_book/order_book.hpp

6.39 ats::order_book_manager Class Reference

```
#include <order_book_manager.hpp>
```

Collaboration diagram for ats::order_book_manager:

Public Member Functions

- const std::vector< ats::order_book > & get_order_books () const
- std::vector< ats::order_book > & get_order_books ()
- const ats::order_book & get_order_book (const ats::symbol_key &symbol) const
- ats::order_book & get_order_book (const ats::symbol_key &symbol)
- void create_order_book (const ats::symbol_key &symbol)
- void update_order_book (const ats::symbol_key &symbol, const ats::level2_message &msg)

6.39.1 Member Function Documentation

6.39.1.1 create_order_book()

```
void ats::order_book_manager::create_order_book (
            const ats::symbol_key & symbol ) [inline]
```

References ats::symbol_key::index.

```
        {
            if (symbol.index == books_.size())
                books_.push_back(ats::order_book(symbol));
            else
                throw std::invalid_argument("Cannot create an order book");
        }
```

6.39.1.2 get_order_book() [1/2]

```
const ats::order_book& ats::order_book_manager::get_order_book (
            const ats::symbol_key & symbol ) const [inline]
```

References ats::symbol_key::index.

```
{ return books_[symbol.index]; }
```

6.39.1.3 get_order_book() [2/2]

```
ats::order_book& ats::order_book_manager::get_order_book (
            const ats::symbol_key & symbol ) [inline]
```

References ats::symbol_key::index.

```
{ return books_[symbol.index]; }
```

6.39.1.4 get_order_books() [1/2]

```
const std::vector<ats::order_book>& ats::order_book_manager::get_order_books ( ) const [inline]
```

```
{ return books_; }
```

6.39.1.5 get_order_books() [2/2]

```
std::vector<ats::order_book>& ats::order_book_manager::get_order_books ( )  [inline]
```

```
{ return books_; }
```

6.39.1.6 update_order_book()

```
void ats::order_book_manager::update_order_book (
             const ats::symbol_key & symbol,
             const ats::level2_message & msg )  [inline]
```

References ats::symbol_key::index.

```
        {
            books_[symbol.index].update(msg);
        }
```

The documentation for this class was generated from the following file:

- order_book/order_book_manager.hpp

6.40 ats::order_container Class Reference

```
#include <order_container.hpp>
```

Collaboration diagram for ats::order_container:

Public Member Functions

- order_container & operator+= (const ats::order &order)
- order_container & operator+= (const order_ptr &order)
- order_container & operator+= (order_ptr &&order)
- iterator get (const ats::orderid_t &order_id)
- const_iterator get (const ats::orderid_t &order_id) const
- std::pair< iterator, iterator > get (const std::string &symbol, const std::string &exchange)
- std::pair< iterator, iterator > get (const key_type &key)
- void remove (const ats::orderid_t &order_id)
- void remove (iterator it)
- bool empty () const
- iterator begin ()
- const_iterator cbegin () const
- iterator end ()
- const_iterator cend () const

6.40.1 Member Function Documentation

6.40.1.1 begin()

```
iterator ats::order_container::begin ( ) [inline]
```

```
{ return orders_.begin(); }
```

6.40.1.2 cbegin()

```
const_iterator ats::order_container::cbegin ( ) const [inline]
```

```
{ return orders_.cbegin(); }
```

6.40.1.3 cend()

```
const_iterator ats::order_container::cend ( ) const [inline]
```

```
{ return orders_.cend(); }
```

6.40.1.4 empty()

```
bool ats::order_container::empty ( ) const [inline]
```

```
{ return orders_.empty(); }
```

6.40.1.5 end()

```
iterator ats::order_container::end ( ) [inline]
```

Referenced by get().

```
{ return orders_.end(); }
```

6.40.1.6 get() [1/4]

```
iterator ats::order_container::get (
            const ats::orderid_t & order_id )  [inline]
```

References end().

```
{
    auto find = iterators_.find(order_id);
    return find != iterators_.end() ? find->second : orders_.end();
}
```

6.40.1.7 get() [2/4]

```
const_iterator ats::order_container::get (
            const ats::orderid_t & order_id ) const  [inline]
```

```
{
    auto find = iterators_.find(order_id);
    return find != iterators_.cend() ? find->second : orders_.cend();
}
```

6.40.1.8 get() [3/4]

```
std::pair<iterator, iterator> ats::order_container::get (
            const std::string & symbol,
            const std::string & exchange )  [inline]
```

```
{
    return orders_.equal_range(std::make_pair(symbol, exchange));
}
```

6.40.1.9 get() [4/4]

```
std::pair<iterator, iterator> ats::order_container::get (
            const key_type & key )  [inline]
```

```
{
    return orders_.equal_range(key);
}
```

6.40.1.10 operator+=() [1/3]

```
order_container& ats::order_container::operator+= (
              const ats::order & order )  [inline]
```

References ats::order::exchange, ats::order::id(), and ats::order::symbol().

```
        {
            key_type key(order.symbol(), order.exchange);
            auto it = orders_.insert(std::make_pair(key, std::make_shared<ats::order>(order)));
            iterators_.insert(std::make_pair(order.id(), it));
            return *this;
        }
```

6.40.1.11 operator+=() [2/3]

```
order_container& ats::order_container::operator+= (
              const order_ptr & order )  [inline]
```

```
        {
            key_type key(order->symbol(), order->exchange);
            auto it = orders_.insert(std::make_pair(key, order));
            iterators_.insert(std::make_pair(order->id(), it));
            return *this;
        }
```

6.40.1.12 operator+=() [3/3]

```
order_container& ats::order_container::operator+= (
              order_ptr && order )  [inline]
```

References ats::order::exchange, and ats::order::symbol().

```
        {
            key_type key(order->symbol(), order->exchange);
            auto it = orders_.insert(std::make_pair(key, std::move(order)));
            iterators_.insert(std::make_pair(it->second->id(), it));
            return *this;
        }
```

6.40.1.13 remove() [1/2]

```
void ats::order_container::remove (
              const ats::orderid_t & order_id )  [inline]
```

```
        {
            auto find = iterators_.find(order_id);
            if (find != iterators_.end())
            {
                auto it = find->second;
                iterators_.erase(find);
                orders_.erase(it);
            }
        }
```

6.40.1.14 remove() [2/2]

```
void ats::order_container::remove (
             iterator it )  [inline]
```

```
{
    auto find = iterators_.find(it->second->id());
    if (find != iterators_.end())
    {
        iterators_.erase(find);
        orders_.erase(it);
    }
 }
```

The documentation for this class was generated from the following file:

- container/order_container.hpp

6.41 ats::order_container1 Class Reference

```
#include <order_container1.hpp>
```

Collaboration diagram for ats::order_container1:

Public Member Functions

- order_container1 & operator+= (const ats::order &order)
- iterator get (const ats::orderid_t &order_id)
- std::pair< iterator, iterator > get (const std::type_index &index)
- void remove (const ats::orderid_t &order_id)
- void remove (iterator it)
- bool empty () const
- iterator begin ()
- const_iterator cbegin () const
- iterator end ()
- const_iterator cend () const

6.41.1 Member Function Documentation

6.41.1.1 begin()

```
iterator ats::order_container1::begin ( )  [inline]
```

```
{ return orders_.begin(); }
```

6.41.1.2 cbegin()

```
const_iterator ats::order_container1::cbegin ( ) const  [inline]
```

```
{ return orders_.cbegin(); }
```

6.41.1.3 cend()

```
const_iterator ats::order_container1::cend ( ) const  [inline]
```

```
{ return orders_.cend(); }
```

6.41.1.4 empty()

```
bool ats::order_container1::empty ( ) const  [inline]
```

```
{ return orders_.empty(); }
```

6.41.1.5 end()

```
iterator ats::order_container1::end ( )  [inline]
```

```
{ return orders_.end(); }
```

6.41.1.6 get() [1/2]

```
iterator ats::order_container1::get (
             const ats::orderid_t & order_id )  [inline]
```

```
        {
            auto find = iterators_.find(order_id);
            return find != iterators_.end() ? find->second : orders_.end();
        }
```

6.41.1.7 get() [2/2]

```
std::pair<iterator, iterator> ats::order_container1::get (
             const std::type_index & index ) [inline]
```

```
{
    return orders_.equal_range(index);
}
```

6.41.1.8 operator+=()

```
order_container1& ats::order_container1::operator+= (
             const ats::order & order ) [inline]
```

References ats::order::id().

```
{
    std::type_index index(typeid(order));
    auto it = orders_.insert(order_container_type::value_type(index, std::make_shared<ats::order>(
order)));
    iterators_.insert(std::make_pair(order.id(), it));
    return *this;
}
```

6.41.1.9 remove() [1/2]

```
void ats::order_container1::remove (
             const ats::orderid_t & order_id ) [inline]
```

```
{
    auto find = iterators_.find(order_id);
    if (find != iterators_.end())
    {
        auto it = find->second;
        iterators_.erase(find);
        orders_.erase(it);
    }
}
```

6.41.1.10 remove() [2/2]

```
void ats::order_container1::remove (
             iterator it ) [inline]
```

```
{
    auto find = iterators_.find(it->second->id());
    if (find != iterators_.end())
    {
        iterators_.erase(find);
        orders_.erase(it);
    }
}
```

The documentation for this class was generated from the following file:

- container/order_container1.hpp

6.42 ats::order_processor Class Reference

```
#include <order_processor.hpp>
```

Collaboration diagram for ats::order_processor:

Public Member Functions

- const ats::order_book & get_order_book (const ats::symbol_key &symbol) const
- const ats::position & get_position (const ats::symbol_key &symbol) const
- const std::vector< ats::position > & get_all_positions () const
- void create_order_book (const ats::symbol_key &symbol)
- void on_add_security (const ats::symbol_key &symbol, const std::string &exchange, size_t book_depth)

6.42.1 Member Function Documentation

6.42.1.1 create_order_book()

```
void ats::order_processor::create_order_book (
             const ats::symbol_key & symbol )  [inline]
```

```
        {
            order_books_.push_back(ats::order_book(symbol));
        }
```

6.42.1.2 get_all_positions()

```
const std::vector<ats::position>& ats::order_processor::get_all_positions ( ) const  [inline]
```

```
{ return positions_; }
```

6.42.1.3 get_order_book()

```
const ats::order_book& ats::order_processor::get_order_book (
             const ats::symbol_key & symbol ) const  [inline]
```

References ats::symbol_key::index.

```
        {
            return order_books_[symbol.index];
        }
```

6.42.1.4 get_position()

```
const ats::position& ats::order_processor::get_position (
            const ats::symbol_key & symbol ) const  [inline]
```

References ats::symbol_key::index.

```
{
    return positions_[symbol.index];
}
```

6.42.1.5 on_add_security()

```
void ats::order_processor::on_add_security (
            const ats::symbol_key & symbol,
            const std::string & exchange,
            size_t book_depth )  [inline]
```

```
{

}
```

The documentation for this class was generated from the following file:

- portfolio/order_processor.hpp

6.43 ats::order_status_cancelled_message Struct Reference

```
#include <order_status_message.hpp>
```

Inheritance diagram for ats::order_status_cancelled_message:

Collaboration diagram for ats::order_status_cancelled_message:

Public Member Functions

- order_status_cancelled_message (const ats::orderid_t &id, const ats::timestamp_t &time, const std::string &cancel_reason="")

Public Attributes

- std::string cancel_reason

6.43.1 Constructor & Destructor Documentation

6.43.1.1 order_status_cancelled_message()

```
ats::order_status_cancelled_message::order_status_cancelled_message (
            const ats::orderid_t & id,
            const ats::timestamp_t & time,
            const std::string & cancel_reason = "" )  [inline]
```

```
            : ats::order_status_message(id,
    ats::order_status::Canceled, time), cancel_reason(
    cancel_reason) { }
```

6.43.2 Member Data Documentation

6.43.2.1 cancel_reason

```
std::string ats::order_status_cancelled_message::cancel_reason
```

The documentation for this struct was generated from the following file:

- message/order_status_message.hpp

6.44 ats::order_status_filled_message Struct Reference

```
#include <order_status_message.hpp>
```

Inheritance diagram for ats::order_status_filled_message:

Collaboration diagram for ats::order_status_filled_message:

Public Member Functions

- order_status_filled_message (const ats::orderid_t &id, const ats::timestamp_t &time, ats::price_t price, long quantity)

Public Attributes

- ats::price_t price
- long quantity

6.44.1 Constructor & Destructor Documentation

6.44.1.1 order_status_filled_message()

```
ats::order_status_filled_message::order_status_filled_message (
            const ats::orderid_t & id,
            const ats::timestamp_t & time,
            ats::price_t price,
            long quantity )  [inline]
```

```
            : ats::order_status_message(id,
     ats::order_status::Filled, time), price(price),
     quantity(quantity) { }
```

6.44.2 Member Data Documentation

6.44.2.1 price

```
ats::price_t ats::order_status_filled_message::price
```

6.44.2.2 quantity

```
long ats::order_status_filled_message::quantity
```

The documentation for this struct was generated from the following file:

- message/order_status_message.hpp

6.45 ats::order_status_message Struct Reference

```
#include <order_status_message.hpp>
```

Inheritance diagram for ats::order_status_message:

Collaboration diagram for ats::order_status_message:

Public Member Functions

- order_status_message (const ats::orderid_t &id, ats::order_status status, const ats::timestamp_t &time)

Public Attributes

- ats::orderid_t order_id
- ats::order_status order_status
- long replaces_order_id = -1
- long replaced_by_order_id = -1
- long parent_id = -1

6.45.1 Constructor & Destructor Documentation

6.45.1.1 order_status_message()

```
ats::order_status_message::order_status_message (
            const ats::orderid_t & id,
            ats::order_status status,
            const ats::timestamp_t & time )  [inline]
```

```
            : ats::message(time), order_id(id), order_status(status) { }
```

6.45.2 Member Data Documentation

6.45.2.1 order_id

```
ats::orderid_t ats::order_status_message::order_id
```

Referenced by ats::portfolio_base::process_order_status_message().

6.45.2.2 order_status

```
ats::order_status ats::order_status_message::order_status
```

Referenced by ats::portfolio_base::process_order_status_message().

6.45.2.3 parent_id

```
long ats::order_status_message::parent_id = -1
```

6.45.2.4 replaced_by_order_id

```
long ats::order_status_message::replaced_by_order_id = -1
```

6.45.2.5 replaces_order_id

```
long ats::order_status_message::replaces_order_id = -1
```

The documentation for this struct was generated from the following file:

- message/order_status_message.hpp

6.46 ats::order_status_new_message Struct Reference

```
#include <order_status_message.hpp>
```

Inheritance diagram for ats::order_status_new_message:

Collaboration diagram for ats::order_status_new_message:

Public Member Functions

- order_status_new_message (const ats::orderid_t &id, const ats::timestamp_t &time, const ats::timestamp_t &time_accepted)

Public Attributes

- ats::timestamp_t time_accepted

6.46.1 Constructor & Destructor Documentation

6.46.1.1 order_status_new_message()

```
ats::order_status_new_message::order_status_new_message (
            const ats::orderid_t & id,
            const ats::timestamp_t & time,
            const ats::timestamp_t & time_accepted )  [inline]
```

```
            : ats::order_status_message(id,
    ats::order_status::New, time), time_accepted(time_accepted) { }
```

6.46.2 Member Data Documentation

6.46.2.1 time_accepted

```
ats::timestamp_t ats::order_status_new_message::time_accepted
```

The documentation for this struct was generated from the following file:

- message/order_status_message.hpp

6.47 ats::order_status_partially_filled_message Struct Reference

```
#include <order_status_message.hpp>
```

Inheritance diagram for ats::order_status_partially_filled_message:

Collaboration diagram for ats::order_status_partially_filled_message:

Public Member Functions

- order_status_partially_filled_message (const ats::orderid_t &id, const ats::timestamp_t &time, ats::price_↩ t price, long quantity)

Public Attributes

- ats::price_t price
- long quantity

6.47.1 Constructor & Destructor Documentation

6.47.1.1 order_status_partially_filled_message()

```
ats::order_status_partially_filled_message::order_status_partially_filled_message (
            const ats::orderid_t & id,
            const ats::timestamp_t & time,
            ats::price_t price,
            long quantity )  [inline]
```

```
         : ats::order_status_message(id,
    ats::order_status::PartiallyFilled, time),
    price(price), quantity(quantity) { }
```

6.47.2 Member Data Documentation

6.47.2.1 price

```
ats::price_t ats::order_status_partially_filled_message::price
```

6.47.2.2 quantity

```
long ats::order_status_partially_filled_message::quantity
```

The documentation for this struct was generated from the following file:

- message/order_status_message.hpp

6.48 ats::order_status_pending_new_message Struct Reference

```
#include <order_status_message.hpp>
```

Inheritance diagram for ats::order_status_pending_new_message:

Collaboration diagram for ats::order_status_pending_new_message:

Public Member Functions

- order_status_pending_new_message (const ats::orderid_t &id, const ats::timestamp_t &time)

Additional Inherited Members

6.48.1 Constructor & Destructor Documentation

6.48.1.1 order_status_pending_new_message()

```
ats::order_status_pending_new_message::order_status_pending_new_message (
            const ats::orderid_t & id,
            const ats::timestamp_t & time )  [inline]
```

```
66         : ats::order_status_message(id,
      ats::order_status::PendingNew, time) { }
```

The documentation for this struct was generated from the following file:

- message/order_status_message.hpp

6.49 ats::order_status_rejected_message Struct Reference

```
#include <order_status_message.hpp>
```

Inheritance diagram for ats::order_status_rejected_message:

Collaboration diagram for ats::order_status_rejected_message:

Public Member Functions

- order_status_rejected_message (const ats::orderid_t &id, const ats::timestamp_t &time, const std::string &rejection_reason="")

Public Attributes

- std::string rejection_reason

6.49.1 Constructor & Destructor Documentation

6.49.1.1 order_status_rejected_message()

```
ats::order_status_rejected_message::order_status_rejected_message (
            const ats::orderid_t & id,
            const ats::timestamp_t & time,
            const std::string & rejection_reason = "" )  [inline]
```

```
58            : ats::order_status_message(id,
      ats::order_status::Rejected, time), rejection_reason(
      rejection_reason) { }
```

6.49.2 Member Data Documentation

6.49.2.1 rejection_reason

```
std::string ats::order_status_rejected_message::rejection_reason
```

The documentation for this struct was generated from the following file:

- message/order_status_message.hpp

6.50 ats::performance_item Struct Reference

```
#include <report_engine.hpp>
```

Inheritance diagram for ats::performance_item:

Collaboration diagram for ats::performance_item:

Public Attributes

- ats::price_t price
- long profit_ticks
- long quantity

6.50.1 Member Data Documentation

6.50.1.1 price

```
ats::price_t ats::performance_item::price
```

6.50.1.2 profit_ticks

```
long ats::performance_item::profit_ticks
```

6.50.1.3 quantity

```
long ats::performance_item::quantity
```

The documentation for this struct was generated from the following file:

- report/report_engine.hpp

6.51 ats::performance_item_basic Struct Reference

```
#include <report_engine.hpp>
```

Inheritance diagram for ats::performance_item_basic:

Collaboration diagram for ats::performance_item_basic:

Public Attributes

- ats::timestamp_t time_open
- ats::timestamp_t time_close
- double profit

6.51.1 Member Data Documentation

6.51.1.1 profit

```
double ats::performance_item_basic::profit
```

6.51.1.2 time_close

```
ats::timestamp_t ats::performance_item_basic::time_close
```

6.51.1.3 time_open

```
ats::timestamp_t ats::performance_item_basic::time_open
```

The documentation for this struct was generated from the following file:

- report/report_engine.hpp

6.52 ats::pnl_item Struct Reference

```
#include <report_engine.hpp>
```

Collaboration diagram for ats::pnl_item:

Public Attributes

- ats::timestamp_t time
- double profit

6.52.1 Member Data Documentation

6.52.1.1 profit

```
double ats::pnl_item::profit
```

Referenced by ats::portfolio_base::process_execution().

6.52.1.2 time

```
ats::timestamp_t ats::pnl_item::time
```

Referenced by ats::portfolio_base::process_execution().

The documentation for this struct was generated from the following file:

- report/report_engine.hpp

6.53 ats::portfolio_base Class Reference

```
#include <portfolio_base.hpp>
```

Inheritance diagram for ats::portfolio_base:

Collaboration diagram for ats::portfolio_base:

Public Member Functions

- portfolio_base ()
- virtual ∼portfolio_base ()
- virtual void on_init ()
- virtual void on_exit ()
- const ats::timestamp_t & current_time () const
- void set_bar_parameters (const boost::posix_time::time_duration &bar_periodicity, size_t bars_to_store)
- void on_time_update (const ats::timestamp_t &time)
- void process_order_status_message (const ats::order_status_message &msg)
- void process_message (const ats::level2_message_packet &msg)
- void send_order (const ats::order &order)
- void cancel_order (const ats::orderid_t &order_id)
- void cancel_pending_orders ()
- void add_connection (ats::execution_engine ∗engine)
- std::ofstream & LOG ()
- void add_security (ats::security_base ∗sec)
- void add_security (const security_base_ptr &sec)
- void create_order_book (const std::string &symbol, const std::string &exchange, size_t book_depth=10U)
- ats::execution_engine ∗ get_execution_engine (const std::string &engine_name)
- const std::vector< ats::symbol_key > & get_symbols () const
- security_base_ptr & get_security (const ats::symbol_key &symbol)
- template<typename SecurityT >
  std::shared_ptr< SecurityT > get_casted_security (const ats::symbol_key &symbol)
- security_base_ptr ∗ get_security (const std::string &symbol)
- std::shared_ptr< ats::order > ∗ get_order (const ats::orderid_t &order_id)
- const ats::symbol_key ∗ get_symbol_key (const std::string &symbol) const
- const ats::order_book & get_order_book (const ats::symbol_key &symbol) const
- const ats::position & get_position (const ats::symbol_key &symbol) const
- const ats::report_engine & get_report () const
- size_t get_next_order_id () const
- size_t get_next_symbol_key () const
- void process_execution (const ats::symbol_key &symbol, ats::order_side side, long quantity, ats::price_t price, const ats::timestamp_t &time)

Protected Attributes

- std::unordered_map< std::string, ats::symbol_key > symbol_keys_

6.53.1 Constructor & Destructor Documentation

6.53.1.1 portfolio_base()

```
ats::portfolio_base::portfolio_base ( )  [inline]
```

References process_message(), and process_order_status_message().

```
                               : log_("LOG.txt")
          {
               // register "global" event handlers (i.e., not having info about the symbol and exchange)
               this->add_event_handler(&
    portfolio_base::process_order_status_message, this);
               this->add_event_handler(&
    portfolio_base::process_message, this);
//             this->add_event_handler(&portfolio_base::on_order_book_changed, this);

               on_init();
          }
```

6.53.1.2 ∼portfolio_base()

```
virtual ats::portfolio_base::~portfolio_base ( )  [inline], [virtual]
```

```
{    }
```

6.53.2 Member Function Documentation

6.53.2.1 add_connection()

```
void ats::portfolio_base::add_connection (
            ats::execution_engine * engine )  [inline]
```

References ats::level2_execution_engine::add_order_book_changed_listener(), ats::execution_engine::add_order_status_listener(), ats::Level2, ats::execution_engine::name(), ats::execution_engine::subscribe(), ats::execution_engine::subscription(), and ats::TimeAndSales.

```
      {
          // subscribe securities that trade on the venue
          for (const auto& sec : securities_)
          {
              const auto& book = sec->order_book();
              if (book.get(engine->name()) != nullptr)
                  engine->subscribe(sec->symbol());
          }

          engine->add_order_status_listener([=](const
ats::order_status_message& msg) {
process_order_status_message(msg); });

          if (engine->subscription() == ats::subscription::Level2)
          {
              ats::level2_execution_engine* l2_engine = static_cast<
ats::level2_execution_engine*>(engine);
              l2_engine->add_order_book_changed_listener([=](const
ats::level2_message_packet& msg) { process_message(msg); });
          }
          else if (engine->subscription() ==
ats::subscription::TimeAndSales)
          {
          }

          execution_engines_.insert(std::make_pair(engine->name(), engine));
      }
```

6.53.2.2 add_security() [1/2]

```
void ats::portfolio_base::add_security (
            ats::security_base * sec )  [inline]
```

References ats::symbol_key::name, and ats::security_base::symbol().

```
      {
          const ats::symbol_key& symbol = sec->symbol();
          symbols_.push_back(symbol);
          symbol_keys_.insert(std::make_pair(symbol.name, symbol));
          positions_.push_back(ats::position(symbol));

          securities_.push_back(security_base_ptr(dynamic_cast<decltype(sec)>(sec)));
      }
```

6.53.2.3 add_security() [2/2]

```
void ats::portfolio_base::add_security (
            const security_base_ptr & sec )  [inline]
```

References ats::symbol_key::name.

```
      {
          securities_.push_back(sec);

          const ats::symbol_key& symbol = sec->symbol();
          symbols_.push_back(symbol);
          symbol_keys_.insert(std::make_pair(symbol.name, symbol));
          positions_.push_back(ats::position(symbol));
      }
```

6.53.2.4 cancel_order()

```
void ats::portfolio_base::cancel_order (
            const ats::orderid_t & order_id )  [inline]
```

References ats::execution_engine::cancel_order().

```
        {
/*          auto it = orders_.get(order_id);
            if (it != orders_.cend())
            {
                ats::execution_engine* engine = execution_engines_[it->second->exchange];
                engine->cancel_order(order_id);
            }*/
            auto it = orders_.find(order_id);
            if (it != orders_.cend())
            {
                ats::execution_engine* engine = execution_engines_[it->second->exchange];
                engine->cancel_order(order_id);
            }
        }
```

6.53.2.5 cancel_pending_orders()

```
void ats::portfolio_base::cancel_pending_orders ( )  [inline]
```

```
        {
/*          for (auto it = orders_.begin(); it != orders_.end(); ++it)
            {
                auto& order = it->second;
                if (order->is_pending())
                {
                    std::cout << "Trying to cancel order id=" << it->second->id() << '\n';
                    cancel_order(order->id());
                }
            }*/
            for (auto it = orders_.begin(); it != orders_.end();)
            {
                if (it->second->is_pending())
                    cancel_order((it++)->second->id());
                else
                    ++it;
            }
        }
```

6.53.2.6 create_order_book()

```
void ats::portfolio_base::create_order_book (
            const std::string & symbol,
            const std::string & exchange,
            size_t book_depth = 10U )  [inline]
```

References ats::symbol_key::index.

```
        {
            const ats::symbol_key* key = get_symbol_key(symbol);
            if (key != nullptr)
                securities_[key->index]->create_order_book(exchange, book_depth);
        }
```

6.53.2.7 current_time()

```
const ats::timestamp_t& ats::portfolio_base::current_time ( ) const  [inline]
```

Referenced by ats::security_base::current_time().

```
{ return time_; }
```

6.53.2.8 get_casted_security()

```
template<typename SecurityT >
std::shared_ptr<SecurityT> ats::portfolio_base::get_casted_security (
            const ats::symbol_key & symbol )  [inline]
```

References ats::symbol_key::index.

```
        {
            security_base_ptr& sec = securities_[symbol.index];
            return std::static_pointer_cast<SecurityT>(sec);
        }
```

6.53.2.9 get_execution_engine()

```
ats::execution_engine* ats::portfolio_base::get_execution_engine (
            const std::string & engine_name )  [inline]
```

Referenced by ats::exchange_message_reader_base< ats::level2_message_packet >::send_message().

```
        {
            auto find = execution_engines_.find(engine_name);
            return find != execution_engines_.end() ? find->second : nullptr;
        }
```

6.53.2.10 get_next_order_id()

```
size_t ats::portfolio_base::get_next_order_id ( ) const  [inline]
```

```
        {
            static size_t id = 1;
            return id++;
        }
```

6.53.2.11 get_next_symbol_key()

```
size_t ats::portfolio_base::get_next_symbol_key ( ) const  [inline]
```

```
        {
            static size_t index = 0;
            return index++;
        }
```

6.53.2.12 get_order()

```
std::shared_ptr<ats::order>* ats::portfolio_base::get_order (
            const ats::orderid_t & order_id )  [inline]
```

```
        {
//          auto it = orders_.get(order_id);
            auto it = orders_.find(order_id);
            return it != orders_.end() ? &it->second : nullptr;
        }
```

6.53.2.13 get_order_book()

```
const ats::order_book& ats::portfolio_base::get_order_book (
            const ats::symbol_key & symbol ) const  [inline]
```

References ats::symbol_key::index, and ats::order_book::order_book().

Referenced by ats::security_base::get_order_book().

```
        {
//          return order_books_[symbol.index];

            return securities_[symbol.index]->order_book();
        }
```

6.53.2.14 get_position()

```
const ats::position& ats::portfolio_base::get_position (
            const ats::symbol_key & symbol ) const  [inline]
```

References ats::symbol_key::index.

Referenced by ats::security_base::get_position().

```
        {
            return positions_[symbol.index];
        }
```

6.53.2.15 get_report()

```
const ats::report_engine& ats::portfolio_base::get_report ( ) const  [inline]
```

```
{ return report_; }
```

6.53.2.16 get_security() [1/2]

```
security_base_ptr& ats::portfolio_base::get_security (
            const ats::symbol_key & symbol )  [inline]
```

References ats::symbol_key::index.

```
        {
            return securities_[symbol.index];
        }
```

6.53.2.17 get_security() [2/2]

```
security_base_ptr* ats::portfolio_base::get_security (
            const std::string & symbol )  [inline]
```

References ats::symbol_key::index.

```
        {
            const ats::symbol_key* key = get_symbol_key(symbol);
            return key != nullptr ? &securities_[key->index] : nullptr;
        }
```

6.53.2.18 get_symbol_key()

```
const ats::symbol_key* ats::portfolio_base::get_symbol_key (
            const std::string & symbol ) const  [inline]
```

Referenced by process_order_status_message(), and send_order().

```
        {
            auto it = symbol_keys_.find(symbol);
            return it != symbol_keys_.cend() ? &it->second : nullptr;
        }
```

6.53.2.19 get_symbols()

```
const std::vector<ats::symbol_key>& ats::portfolio_base::get_symbols ( ) const  [inline]
```

```
{ return symbols_; }
```

6.53.2.20 LOG()

```
std::ofstream& ats::portfolio_base::LOG ( )  [inline]
```

Referenced by ats::security_base::LOG().

```
{ return log_;  }
```

6.53.2.21 on_exit()

```
virtual void ats::portfolio_base::on_exit ( )  [inline], [virtual]
```

```
{ }
```

6.53.2.22 on_init()

```
virtual void ats::portfolio_base::on_init ( )  [inline], [virtual]
```

```
{ }
```

6.53.2.23 on_time_update()

```
void ats::portfolio_base::on_time_update (
              const ats::timestamp_t & time )  [inline]
```

Referenced by process_order_status_message().

```
        {
            time_ = time;

/*          for (auto& sec: securities_)
                sec->process_time_update(time);*/
        }
```

6.53.2.24 process_execution()

```
void ats::portfolio_base::process_execution (
            const ats::symbol_key & symbol,
            ats::order_side side,
            long quantity,
            ats::price_t price,
            const ats::timestamp_t & time )  [inline]
```

References ats::position::add_execution(), ats::Buy, ats::BuyCover, ats::symbol_key::index, ats::position::is_long(), ats::pnl_item::profit, ats::position::quantity(), ats::position::realized_pnl(), and ats::pnl_item::time.

Referenced by process_order_status_message().

```
        {
            ats::position& pos = positions_[symbol.index];
            double previous_pnl = pos.realized_pnl();
            bool is_long = side == ats::order_side::Buy || side ==
     ats::order_side::BuyCover;
            bool is_opposite_dir = pos.quantity() != 0 && is_long != pos.
     is_long();

            // Add execution to position
            pos.add_execution(side, quantity, price, time);

            if (is_opposite_dir)
            {
                ats::pnl_item item;
                item.time = time;
                item.profit = pos.realized_pnl() - previous_pnl;
                report_.add_pnl_item(item);
            }
        }
```

6.53.2.25 process_message()

```
void ats::portfolio_base::process_message (
            const ats::level2_message_packet & msg )  [inline]
```

References ats::symbol_key::index, ats::instrument_message::symbol, and ats::message::time.

Referenced by portfolio_base().

```
        {
            on_time_update(msg.time);

            const ats::symbol_key* symbol = get_symbol_key(msg.
     symbol);
            if (symbol != nullptr)
                securities_[symbol->index]->process_message(msg);
        }
```

6.53.2.26 process_order_status_message()

```
void ats::portfolio_base::process_order_status_message (
              const ats::order_status_message & msg )
```

References ats::Canceled, ats::Filled, get_symbol_key(), ats::symbol_key::index, on_time_update(), ats::order_status_message::order_id, ats::order_status_message::order_status, ats::PartiallyFilled, process_execution(), ats::Rejected, and ats::message::time.

Referenced by portfolio_base().

```
    {
        on_time_update(msg.time);

//      auto it = orders_.get(msg.order_id);
        auto it = orders_.find(msg.order_id);
        if (it == orders_.end()) return;
        auto& ord_ptr = it->second;

        const ats::symbol_key* symbol = get_symbol_key(ord_ptr->symbol());
        if (symbol == nullptr)
        {
            std::cout << "ERROR: (Security, Exchange)=(" << ord_ptr->symbol() << "," << ord_ptr->exchange
                << ") doesn’t exist\n";
            return;
        }

        security_base_ptr& sec = securities_[symbol->index];
        ats::position& pos = positions_[symbol->index]; //sec->get_position();

        if (msg.order_status == ats::order_status::Filled)
        {
            // Modify the security position and remove the order
            const auto& m = static_cast<const ats::order_status_filled_message
   &>(msg);
            process_execution(*symbol, ord_ptr->side(), m.quantity, m.price, m.time);
            ord_ptr->executed_quantity = ord_ptr->quantity();
            ord_ptr->set_status(ats::order_status::Filled);
            //orders_.remove(it);
            orders_.erase(it);
        }
        else if (msg.order_status ==
   ats::order_status::PartiallyFilled)
        {
            // Modify the security position and the order quantity
            const auto& m = static_cast<const ats::order_status_filled_message
   &>(msg);
            process_execution(*symbol, ord_ptr->side(), m.quantity, m.price, m.time);
            ord_ptr->set_quantity(ord_ptr->quantity() - m.quantity);
            ord_ptr->executed_quantity += m.quantity;
            ord_ptr->set_status(ats::order_status::PartiallyFilled);
        }
        else if (msg.order_status == ats::order_status::Canceled ||
            msg.order_status == ats::order_status::Rejected)
        {
            orders_.erase(it);
            if (msg.order_status == ats::order_status::Rejected)
                std::cout << "Order cancelled: symbol=" << sec->symbol().to_string() << ’\n’;
            if (msg.order_status == ats::order_status::Rejected)
                std::cout << "Rejected: symbol=" << sec->symbol().to_string()
                << ", reason=" << static_cast<const
   ats::order_status_rejected_message&>(msg).rejection_reason << ’\n’;
        }

        // Pass the message to the related security
        sec->on_order_status_changed(msg);
    }
```

6.53.2.27 send_order()

```
void ats::portfolio_base::send_order (
            const ats::order & order )
```

References ats::order::exchange, get_symbol_key(), ats::order::id(), ats::symbol_key::index, ats::symbol_key::name, ats::execution_engine::send_order(), and ats::order::symbol().

```
{
    auto engine_it = execution_engines_.find(order.exchange);
    ats::execution_engine* engine;

     if (engine_it != execution_engines_.cend())
     {
         const ats::symbol_key* symbol = get_symbol_key(order.symbol());
         const std::string& venue_name = engine_it->second->name();
         //if (symbol != nullptr && order_books_[symbol->index].get(venue_name) != nullptr)
         if (symbol != nullptr && securities_[symbol->index]->order_book().get(venue_name) != nullptr)
         {
             //orders_ += order;
             orders_.insert(std::make_pair(order.id(), std::make_shared<ats::order>(order)));
             engine = engine_it->second;
             //                  venue_it->second->send_order(order);
         }
         else
         {
             std::cout << "ERROR: Exchange '" << venue_name
                     << "' has no subscription for security '" << order.symbol() << '\n';
             return;
         }
     }
     else
     {
         std::cout << "ERROR: Cannot find exchange '" << order.exchange << "'\n";
         return;
     }

     //          venue->send_order(static_cast<decltype(order)>(order));

     std::type_index order_type(typeid(order));
     if (order_type == typeid(ats::market_order))
         engine->send_order(static_cast<const ats::market_order&>(order));
     else if (order_type == typeid(ats::limit_order))
         engine->send_order(static_cast<const ats::limit_order&>(order));
     else if (order_type == typeid(ats::stop_order))
         engine->send_order(static_cast<const ats::stop_order&>(order));
 }
```

6.53.2.28 set_bar_parameters()

```
void ats::portfolio_base::set_bar_parameters (
            const boost::posix_time::time_duration & bar_periodicity,
            size_t bars_to_store )   [inline]
```

```
{
    bar_periodicity_ = bar_periodicity;
    bars_to_store_ = bars_to_store;

    for (auto& sec : securities_)
        sec->set_bar_parameters(bar_periodicity, bars_to_store);
}
```

6.53.3 Member Data Documentation

6.53.3.1 symbol_keys_

```
std::unordered_map<std::string, ats::symbol_key> ats::portfolio_base::symbol_keys_  [protected]
```

The documentation for this class was generated from the following files:

- portfolio/portfolio_base.hpp
- portfolio/portfolio_base.cpp

6.54 ats::position Class Reference

```
#include <position.hpp>
```

Collaboration diagram for ats::position:

Public Member Functions

- position (const ats::symbol_key &symbol)
- long quantity () const
- const ats::timestamp_t & time () const
- bool is_long () const
- double price () const
- long realized_pnl () const
- long inventory () const
- void add_execution (ats::order_side side, long quantity, ats::price_t price, const ats::timestamp_t &time)

6.54.1 Constructor & Destructor Documentation

6.54.1.1 position()

```
ats::position::position (
            const ats::symbol_key & symbol )  [inline]
```

```
            : quantity_(0), is_long_(true), avg_price_(0.0), symbol_(symbol) { }
```

6.54.2 Member Function Documentation

6.54.2.1 add_execution()

```
void ats::position::add_execution (
              ats::order_side side,
              long quantity,
              ats::price_t price,
              const ats::timestamp_t & time )  [inline]
```

References ats::Buy, ats::BuyCover, ats::position_offset_message::is_long, ats::position_offset_message::price, and ats::position_offset_message::quantity.

Referenced by ats::portfolio_base::process_execution().

```
     {
         bool is_long = side == ats::order_side::Buy || side ==
ats::order_side::BuyCover;
         add(price, quantity, is_long);
     }
```

6.54.2.2 inventory()

```
long ats::position::inventory ( ) const  [inline]
```

Referenced by ats::security_base::get_inventory().

```
     {
         if (quantity_ == 0)
             return 0;
         else
             return is_long_ ? quantity_ : -quantity_;
     }
```

6.54.2.3 is_long()

```
bool ats::position::is_long ( ) const  [inline]
```

Referenced by ats::portfolio_base::process_execution().

```
{ return is_long_; }
```

6.54.2.4 price()

```
double ats::position::price ( ) const  [inline]
```

```
{ return avg_price_; }
```

6.54.2.5 quantity()

```
long ats::position::quantity ( ) const  [inline]
```

Referenced by ats::portfolio_base::process_execution().

```
{ return quantity_; }
```

6.54.2.6 realized_pnl()

```
long ats::position::realized_pnl ( ) const  [inline]
```

Referenced by ats::portfolio_base::process_execution().

```
{ return realized_pnl_; }
```

6.54.2.7 time()

```
const ats::timestamp_t& ats::position::time ( ) const  [inline]
```

```
{ return time_; }
```

The documentation for this class was generated from the following file:

- position/position.hpp

6.55 ats::position_offset_message Struct Reference

```
#include <position.hpp>
```

Inheritance diagram for ats::position_offset_message:

Collaboration diagram for ats::position_offset_message:

Public Attributes

- std::string symbol
- ats::price_t price
- long quantity
- ats::price_t profit_ticks
- ats::timestamp_t close_time
- bool is_long

Additional Inherited Members

6.55.1 Member Data Documentation

6.55.1.1 close_time

```
ats::timestamp_t ats::position_offset_message::close_time
```

6.55.1.2 is_long

```
bool ats::position_offset_message::is_long
```

Referenced by ats::position::add_execution().

6.55.1.3 price

```
ats::price_t ats::position_offset_message::price
```

Referenced by ats::position::add_execution().

6.55.1.4 profit_ticks

```
ats::price_t ats::position_offset_message::profit_ticks
```

6.55.1.5 quantity

```
long ats::position_offset_message::quantity
```

Referenced by ats::position::add_execution().

6.55.1.6 symbol

```
std::string ats::position_offset_message::symbol
```

The documentation for this struct was generated from the following file:

- position/position.hpp

6.56 ats::order_book_detail::price_level Struct Reference

price level in an order book

```
#include <price_level.hpp>
```

Collaboration diagram for ats::order_book_detail::price_level:

Public Member Functions

- price_level ()
- price_level (ats::price_t price, long quantity, unsigned int order_count)
- void set (ats::price_t price, long quantity, unsigned int order_count)

Public Attributes

- ats::price_t price
- long quantity
- unsigned int order_count

6.56.1 Detailed Description

price level in an order book

6.56.2 Constructor & Destructor Documentation

6.56.2.1 price_level() [1/2]

```
ats::order_book_detail::price_level::price_level ( )  [inline]
```

```
            : price(0), quantity(0), order_count(0) { }
```

6.56.2.2 price_level() [2/2]

```
ats::order_book_detail::price_level::price_level (
            ats::price_t price,
            long quantity,
            unsigned int order_count )  [inline]
```

```
            : price(price), quantity(quantity), order_count(order_count) { }
```

6.56.3 Member Function Documentation

6.56.3.1 set()

```
void ats::order_book_detail::price_level::set (
            ats::price_t price,
            long quantity,
            unsigned int order_count )  [inline]
```

References order_count, price, and quantity.

```
        {
            this->price = price;
            this->quantity = quantity;
            this->order_count = order_count;
        }
```

6.56.4 Member Data Documentation

6.56.4.1 order_count

```
unsigned int ats::order_book_detail::price_level::order_count
```

Referenced by set().

6.56.4.2 price

```
ats::price_t ats::order_book_detail::price_level::price
```

Referenced by ats::sim::fifo_exchange_order_book::add_order(), ats::level2_execution_engine::on_order_book_changed(), ats::level2_execution_engine::send_order(), set(), and ats::sim::fifo_exchange_order_book::update().

6.56.4.3 quantity

```
long ats::order_book_detail::price_level::quantity
```

Referenced by set(), and ats::sim::fifo_exchange_order_book::update().

The documentation for this struct was generated from the following file:

- order_book/detail/price_level.hpp

6.57 ats::sim::price_level Class Reference

```
#include <sim_book_price_level.hpp>
```

Collaboration diagram for ats::sim::price_level:

Public Types

- typedef std::list< ats::limit_order > orderqueue_type
- typedef orderqueue_type::iterator iterator
- typedef std::unordered_map< ats::orderid_t, iterator > order_container

Public Member Functions

- price_level (ats::price_t price)
- iterator begin ()
- iterator end ()
- ats::price_t price () const
- bool is_defined () const
- iterator add_order (const ats::limit_order &order)
- iterator insert_order (const ats::limit_order &order)
- bool erase_order (iterator position)
- void clean ()
- void clean (long quantity)
- void execute_all_orders (const ats::timestamp_t &time, ats::order_status_handler &listener, order_container &orders)
- void execute_orders (long quantity, const ats::timestamp_t &time, ats::order_status_handler &listener, order_container &orders)
- std::string to_string () const
- void process_change_msg (const ats::level2_message &msg, ats::order_status_handler &listener, order_container &orders)

Public Attributes

- long quantity = 0
- long sim_quantity = 0
- long traded_quantity = 0

6.57.1 Member Typedef Documentation

6.57.1.1 iterator

```
typedef orderqueue_type::iterator ats::sim::price_level::iterator
```

6.57.1.2 order_container

```
typedef std::unordered_map<ats::orderid_t, iterator> ats::sim::price_level::order_container
```

6.57.1.3 orderqueue_type

```
typedef std::list<ats::limit_order> ats::sim::price_level::orderqueue_type
```

6.57.2 Constructor & Destructor Documentation

6.57.2.1 price_level()

```
ats::sim::price_level::price_level (
             ats::price_t price )  [inline]
```

```
: price_(price) { }
```

6.57.3 Member Function Documentation

6.57.3.1 add_order()

```
price_level::iterator ats::sim::price_level::add_order (
               const ats::limit_order & order )  [inline]
```

References ats::order::id(), ats::order::quantity(), quantity, and sim_quantity.

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::add_order(), is_defined(), process_change_msg(), and ats::sim::price_levels< std::greater< ats::price_t > >::process_insert_msg().

```
        {
            queue_.push_back(order);
            if (order.id() != 0)
                sim_quantity += order.quantity();
            quantity += order.quantity();

            return std::prev(queue_.end());
        }
```

6.57.3.2 begin()

iterator ats::sim::price_level::begin ( ) [inline]

```
{ return queue_.begin(); }
```

6.57.3.3 clean() [1/2]

void ats::sim::price_level::clean ( ) [inline]

References quantity.

Referenced by is_defined(), process_change_msg(), and ats::sim::price_levels< std::greater< ats::price_t > >↩ ::process_insert_msg().

```
{
    for (auto it = queue_.begin(); it != queue_.end();)
    {
        if (it->id() == 0)
            queue_.erase(it++);
        else
            ++it;
    }

    quantity = 0;
}
```

6.57.3.4 clean() [2/2]

void ats::sim::price_level::clean (
long *quantity*) [inline]

References quantity.

```
{
    long remained_qty = qty;
    for (auto it = queue_.rbegin(); it != queue_.rend() && remained_qty > 0;)
    {
        if (it->id() != 0)
            ++it;
        else if (it->quantity() > remained_qty)
        {
            it->set_quantity(it->quantity() - remained_qty);
            quantity -= remained_qty;
            break;
        }
        else
        {
            quantity -= it->quantity();
            remained_qty -= it->quantity();
            auto rm = --it.base();
            ++it;
            queue_.erase(rm);
        }
    }
}
```

6.57.3.5 end()

```
iterator ats::sim::price_level::end ( )  [inline]
```

```
{ return queue_.end(); }
```

6.57.3.6 erase_order()

```
bool ats::sim::price_level::erase_order (
              iterator position )  [inline]
```

References quantity, and sim_quantity.

Referenced by is_defined().

```
        {
            if (position != queue_.end())
            {
                quantity -= position->quantity();
                if (position->id() != 0)
                    sim_quantity -= position->quantity();
                queue_.erase(position);
                return true;
            }
            else
                return false;
        }
```

6.57.3.7 execute_all_orders()

```
void ats::sim::price_level::execute_all_orders (
              const ats::timestamp_t & time,
              ats::order_status_handler & listener,
              order_container & orders )  [inline]
```

References quantity, and sim_quantity.

Referenced by is_defined().

```
        {
            for (auto it = queue_.begin(); it != queue_.end();)
            {
                if (it->id() != 0)
                {
                    auto ord_it = orders.find(it->id());
                    if (ord_it != orders.end())
                        orders.erase(ord_it);

                    ats::order_status_filled_message msg(it->id(), time, price_, it->quantity());
                    listener(msg);
                }
                queue_.erase(it++);
            }

            quantity = 0;
            sim_quantity = 0;
        }
```

6.57.3.8 execute_orders()

```
void ats::sim::price_level::execute_orders (
            long quantity,
            const ats::timestamp_t & time,
            ats::order_status_handler & listener,
            order_container & orders )  [inline]
```

References quantity, and sim_quantity.

Referenced by is_defined(), and process_change_msg().

```
        {
            long unexecuted_qty = quantity;
            for (auto it = queue_.begin(); it != queue_.end() && unexecuted_qty > 0;)
            {
                if (it->quantity() > unexecuted_qty)
                {
                    it->set_quantity(it->quantity() - unexecuted_qty);
                    quantity -= unexecuted_qty;
                    if (it->id() != 0)
                    {
                        sim_quantity -= unexecuted_qty;
                        ats::order_status_partially_filled_message
       msg(it->id(), time, it->price(), unexecuted_qty);
                        listener(msg);
                    }
                    break;
                }
                else
                {
                    quantity -= it->quantity();
                    if (it->id() != 0)
                    {
                        auto ord_it = orders.find(it->id());
                        if (ord_it != orders.end())
                            orders.erase(ord_it);

                        sim_quantity -= it->quantity();
                        ats::order_status_filled_message msg(it->id(), time
      , it->price(), it->quantity());
                        listener(msg);
                    }
                    unexecuted_qty -= it->quantity();
                    queue_.erase(it++);
                }
            }
        }
```

6.57.3.9 insert_order()

```
price_level::iterator ats::sim::price_level::insert_order (
            const ats::limit_order & order )  [inline]
```

References ats::order::id(), ats::order::quantity(), quantity, and sim_quantity.

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::insert_order(), is_defined(), process_change_msg(), and ats::sim::price_levels< std::greater< ats::price_t > >::process_insert_msg().

```
        {
            queue_.push_front(order);
            if (order.id() != 0)
                sim_quantity += order.quantity();
            quantity += order.quantity();
            is_defined_ = true;
            return queue_.begin();
        }
```

6.57.3.10 is_defined()

```
bool ats::sim::price_level::is_defined ( ) const  [inline]
```

References add_order(), clean(), erase_order(), execute_all_orders(), execute_orders(), insert_order(), process_change_msg(), quantity, and to_string().

Referenced by ats::sim::fifo_exchange_order_book::add_order(), and ats::sim::price_levels< std::greater< ats::price_t > >::process_insert_msg().

```
{ return is_defined_; }
```

6.57.3.11 price()

```
ats::price_t ats::sim::price_level::price ( ) const  [inline]
```

Referenced by ats::sim::sim_book::add_order().

```
{ return price_; }
```

6.57.3.12 process_change_msg()

```
void ats::sim::price_level::process_change_msg (
             const ats::level2_message & msg,
             ats::order_status_handler & listener,
             order_container & orders )  [inline]
```

References add_order(), ats::Bid, ats::Buy, clean(), ats::level2_message::entry_type, execute_orders(), ats::GTC, insert_order(), ats::level2_message::price, ats::level2_message::quantity, ats::SellShort, ats::instrument_message::symbol, ats::message::time, and traded_quantity.

Referenced by is_defined().

```
{
    if (msg.quantity > 0)
    {
        ats::order_side side = msg.entry_type == ats::entry_type::Bid ?
            ats::order_side::Buy : ats::order_side::SellShort;
        ats::limit_order order(0, msg.symbol, msg.quantity, side, ats::order_time_in_force::GTC, msg.price);

        if (!is_defined_)
            insert_order(order);
        else
            add_order(order);
    }
    else if (traded_quantity == 0)
        clean(-msg.quantity);
    else
        execute_orders(-msg.quantity, msg.time, listener, orders);

    traded_quantity = 0;
}
```

6.57.3.13 to_string()

```
std::string ats::sim::price_level::to_string ( ) const  [inline]
```

References quantity, sim_quantity, and traded_quantity.

Referenced by is_defined().

```
        {
            std::stringstream ss;
            ss << "Price level " << price_ << "(qty=" << quantity << ",sim_qty=" <<
    sim_quantity
                << ",trd_qty=" << traded_quantity << ";" << is_defined_ << "): ";
            for (const auto& x : queue_)
                ss << "(id=" << x.id() << ",qty=" << x.quantity() << "), ";

            return ss.str();
        }
```

6.57.4 Member Data Documentation

6.57.4.1 quantity

```
long ats::sim::price_level::quantity = 0
```

Referenced by ats::sim::sim_book::add_order(), add_order(), clean(), erase_order(), execute_all_orders(), execute_orders(), insert_order(), is_defined(), ats::sim::price_levels< std::greater< ats::price_t > >::process_↩insert_msg(), and to_string().

6.57.4.2 sim_quantity

```
long ats::sim::price_level::sim_quantity = 0
```

Referenced by add_order(), erase_order(), execute_all_orders(), execute_orders(), insert_order(), ats::sim::price↩_levels< std::greater< ats::price_t > >::process_insert_msg(), and to_string().

6.57.4.3 traded_quantity

```
long ats::sim::price_level::traded_quantity = 0
```

Referenced by process_change_msg(), ats::sim::sim_book::process_trade_msg(), and to_string().

The documentation for this class was generated from the following file:

- order_book/simulation/sim_book_price_level.hpp

6.58 ats::sim::price_levels< comp > Class Template Reference

```
#include <sim_book_price_levels.hpp>
```

Inheritance diagram for ats::sim::price_levels< comp >:

Collaboration diagram for ats::sim::price_levels< comp >:

Public Types

- typedef std::map< ats::price_t, ats::sim::price_level, comp > container_type
- typedef container_type::iterator iterator

Public Member Functions

- void add_order_status_listener (const ats::order_status_handler &listener)
- price_level::iterator add_order (const ats::limit_order &order)
- price_level::iterator insert_order (const ats::limit_order &order)
- void cancel_order (price_level::iterator position, const ats::timestamp_t &time)
- bool erase_order (price_level::iterator position)
- void erase_level (ats::price_t price)
- const price_level ∗ top_level () const
- price_level ∗ top_level ()
- const price_level ∗ get_level (ats::price_t price) const
- iterator begin ()
- iterator end ()
- bool empty () const
- void clean (ats::price_t price)
- void execute_all_orders (ats::price_t price, const ats::timestamp_t &time, order_container &orders)
- void execute_orders (ats::price_t price, long quantity, const ats::timestamp_t &time, order_container &orders)
- void process_change_msg (const ats::level2_message &msg, order_container &orders)
- void process_delete_msg (const ats::level2_message &msg, order_container &orders)
- void process_insert_msg (const ats::level2_message &msg)

6.58.1 Member Typedef Documentation

6.58.1.1 container_type

```
template<typename comp = std::less<ats::price_t>>
typedef std::map<ats::price_t, ats::sim::price_level, comp> ats::sim::price_levels< comp >↩
::container_type
```

6.58.1.2 iterator

```
template<typename comp = std::less<ats::price_t>>
typedef container_type::iterator ats::sim::price_levels< comp >::iterator
```

6.58.2 Member Function Documentation

6.58.2.1 add_order()

```
template<typename comp >
price_level::iterator ats::sim::price_levels< comp >::add_order (
            const ats::limit_order & order )
```

Referenced by ats::sim::sim_book::add_order(), and ats::sim::price_levels< std::greater< ats::price_t > >::add_order_status_listener().

```
    {
        auto it = levels_.find(order.price());
        if (it != levels_.end())
            return it->second.add_order(order);
        else
        {
            price_level l(order.price());
            l.add_order(order);
            auto ins = levels_.insert(std::make_pair(order.price(), l)).first;
            return std::prev(ins->second.end());
        }
    }
```

6.58.2.2 add_order_status_listener()

```
template<typename comp = std::less<ats::price_t>>
void ats::sim::price_levels< comp >::add_order_status_listener (
            const ats::order_status_handler & listener )   [inline]
```

Referenced by ats::sim::sim_book::add_order_status_listener().

```
        {
            order_status_listener_ = listener;
        }
```

6.58.2.3 begin()

```
template<typename comp = std::less<ats::price_t>>
iterator ats::sim::price_levels< comp >::begin ( )   [inline]
```

```
{ return levels_.begin(); }
```

6.58.2.4 cancel_order()

```
template<typename comp >
void ats::sim::price_levels< comp >::cancel_order (
            price_level::iterator position,
            const ats::timestamp_t & time )
```

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::add_order_status_listener(), and ats::sim::sim_book::cancel_order().

```
{
    auto it = levels_.find(position->price());
    if (it != levels_.end())
    {
        it->second.erase_order(position);
         if (it->second.sim_quantity == 0)
             levels_.erase(it);
     }
 }
```

6.58.2.5 clean()

```
template<typename comp >
void ats::sim::price_levels< comp >::clean (
            ats::price_t price )
```

Referenced by ats::sim::sim_book::clean_level(), and ats::sim::price_levels< std::greater< ats::price_t > >::empty().

```
{
    auto it = levels_.find(price);
    if (it != levels_.end())
        it->second.clean();
}
```

6.58.2.6 empty()

```
template<typename comp = std::less<ats::price_t>>
bool ats::sim::price_levels< comp >::empty ( ) const  [inline]
```

```
{ return levels_.empty(); }
```

6.58.2.7 end()

```
template<typename comp = std::less<ats::price_t>>
iterator ats::sim::price_levels< comp >::end ( )  [inline]
```

```
{ return levels_.end(); }
```

6.58.2.8 erase_level()

```
template<typename comp >
void ats::sim::price_levels< comp >::erase_level (
            ats::price_t price )
```

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::add_order_status_listener().

```
{
    auto it = levels_.find(price);
    if (it != levels_.end())
        levels_.erase(it);
}
```

6.58.2.9 erase_order()

```
template<typename comp >
bool ats::sim::price_levels< comp >::erase_order (
            price_level::iterator position )
```

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::add_order_status_listener().

```
{
    auto it = levels_.find(position->price());
    if (it != levels_.end())
        return it->second.erase_order(position);
    else
        return false;
}
```

6.58.2.10 execute_all_orders()

```
template<typename comp >
void ats::sim::price_levels< comp >::execute_all_orders (
            ats::price_t price,
            const ats::timestamp_t & time,
            order_container & orders )
```

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::empty(), ats::sim::sim_book::execute_all_orders(), and ats::sim::price_levels< std::greater< ats::price_t > >::process_delete_msg().

```
{
    auto it = levels_.find(price);
    if (it != levels_.end())
    {
        it->second.execute_all_orders(time, order_status_listener_, orders);
        levels_.erase(it);
    }
}
```

6.58.2.11 execute_orders()

```
template<typename comp >
void ats::sim::price_levels< comp >::execute_orders (
            ats::price_t price,
            long quantity,
            const ats::timestamp_t & time,
            order_container & orders )
```

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::empty(), and ats::sim::sim_book::execute_orders().

```
        {
            auto it = levels_.find(price);
            if (it != levels_.end())
            {
                it->second.execute_orders(quantity, time, order_status_listener_, orders);
                if (it->second.sim_quantity == 0)
                    levels_.erase(it);
            }
        }
```

6.58.2.12 get_level()

```
template<typename comp = std::less<ats::price_t>>
const price_level* ats::sim::price_levels< comp >::get_level (
            ats::price_t price ) const  [inline]
```

Referenced by ats::sim::sim_book::get_level().

```
            {
                auto it = levels_.find(price);
                return it == levels_.cend() ? nullptr : &it->second;
            }
```

6.58.2.13 insert_order()

```
template<typename comp >
price_level::iterator ats::sim::price_levels< comp >::insert_order (
            const ats::limit_order & order )
```

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::add_order_status_listener(), and ats::sim::sim_book::insert_order().

```
        {
            auto it = levels_.find(order.price());
            if (it != levels_.end())
                return it->second.insert_order(order);
            else
            {
                price_level l(order.price());
                l.insert_order(order);
                auto inserted = levels_.insert(std::make_pair(order.price(), l)).first;
                return inserted->second.begin();
            }
        }
```

6.58.2.14 process_change_msg()

```
template<typename comp >
void ats::sim::price_levels< comp >::process_change_msg (
            const ats::level2_message & msg,
            order_container & orders )
```

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::empty(), and ats::sim::sim_book::process_change_msg().

```
        {
            auto it = levels_.find(msg.price);
            if (it != levels_.cend())
            {
                it->second.process_change_msg(msg, order_status_listener_, orders);

                if (it->second.sim_quantity == 0)
                    levels_.erase(it);
            }
        }
```

6.58.2.15 process_delete_msg()

```
template<typename comp >
void ats::sim::price_levels< comp >::process_delete_msg (
            const ats::level2_message & msg,
            order_container & orders )
```

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::empty(), and ats::sim::sim_book::process_delete_msg().

```
        {
            auto it = levels_.find(msg.price);
            if (it != levels_.cend())
            {
                if (it->second.traded_quantity == msg.quantity)
                    execute_all_orders(msg.price, msg.time, orders);
                else
                {
                    it->second.clean();
                    it->second.traded_quantity = 0;
                }
            }
        }
```

6.58.2.16 process_insert_msg()

```
template<typename comp >
void ats::sim::price_levels< comp >::process_insert_msg (
            const ats::level2_message & msg )
```

Referenced by ats::sim::price_levels< std::greater< ats::price_t > >::empty(), and ats::sim::sim_book::process_insert_msg().

```
{
    auto it = levels_.find(msg.price);
    if (it != levels_.cend())
    {
        ats::sim::price_level& level = it->second;
        ats::order_side side = msg.entry_type == ats::entry_type::Bid ?
            ats::order_side::Buy : ats::order_side::SellShort;
        if (!level.is_defined())
        {
            ats::limit_order order(0, msg.symbol, msg.quantity, side,
                ats::order_time_in_force::GTC, msg.price);
            level.insert_order(order);
        }
        else
        {
            long delta = level.quantity - level.sim_quantity;
            if (msg.quantity > delta)
            {
                ats::limit_order order(0, msg.symbol, msg.quantity - delta, side,
                    ats::order_time_in_force::GTC, msg.price);
                level.add_order(order);
            }
            else if (msg.quantity < delta)
                level.clean(delta - msg.quantity);
        }
    }
}
```

6.58.2.17 top_level() [1/2]

```
template<typename comp = std::less<ats::price_t>>
const price_level* ats::sim::price_levels< comp >::top_level ( ) const [inline]
```

Referenced by ats::sim::sim_book::best_ask(), and ats::sim::sim_book::best_bid().

```
{
    return levels_.empty() ? nullptr : &levels_.cbegin()->second;
}
```

6.58.2.18 top_level() [2/2]

```
template<typename comp = std::less<ats::price_t>>
price_level* ats::sim::price_levels< comp >::top_level ( ) [inline]
```

```
{
    return levels_.empty() ? nullptr : &levels_.begin()->second;
}
```

The documentation for this class was generated from the following file:

- order_book/simulation/sim_book_price_levels.hpp

6.59 ats::order_book_detail::price_levels< compare > Class Template Reference

```
#include <price_levels.hpp>
```

Inheritance diagram for ats::order_book_detail::price_levels< compare >:

Collaboration diagram for ats::order_book_detail::price_levels< compare >:

Public Types

- typedef ats::order_book_detail::price_level price_level_type
- typedef std::map< ats::price_t, price_level_type, compare > container_type
- typedef container_type::iterator iterator
- typedef container_type::const_iterator const_iterator
- typedef container_type::reverse_iterator reverse_iterator
- typedef container_type::const_reverse_iterator const_reverse_iterator

Public Member Functions

- price_levels (size_t max_levels)
- void update (const ats::level2_message &msg)
- void update2 (ats::level2_message &msg)
- iterator begin ()
- const_iterator cbegin () const
- iterator end ()
- const_iterator cend () const
- reverse_iterator rbegin ()
- const_reverse_iterator crbegin () const
- reverse_iterator rend ()
- const_reverse_iterator crend ()
- price_level_type & operator[ ] (ats::price_t price)
- const price_level_type & operator[ ] (ats::price_t price) const
- price_level_type & at (ats::price_t price)
- const price_level_type & at (ats::price_t price) const
- bool empty () const
- void clear ()
- iterator find (ats::price_t price)
- const_iterator find (ats::price_t price) const
- size_t size () const
- size_t displayed_depth () const
- iterator get_level (size_t index)
- const_iterator get_level (size_t index) const
- container_type & levels ()

6.59.1 Member Typedef Documentation

6.59.1.1 const_iterator

```
template<typename compare = std::less<ats::price_t>>
typedef container_type::const_iterator ats::order_book_detail::price_levels< compare >::const↩
_iterator
```

6.59.1.2 const_reverse_iterator

```
template<typename compare = std::less<ats::price_t>>
typedef container_type::const_reverse_iterator ats::order_book_detail::price_levels< compare
>::const_reverse_iterator
```

6.59.1.3 container_type

```
template<typename compare = std::less<ats::price_t>>
typedef std::map<ats::price_t, price_level_type, compare> ats::order_book_detail::price_↩
levels< compare >::container_type
```

6.59.1.4 iterator

```
template<typename compare = std::less<ats::price_t>>
typedef container_type::iterator ats::order_book_detail::price_levels< compare >::iterator
```

6.59.1.5 price_level_type

```
template<typename compare = std::less<ats::price_t>>
typedef ats::order_book_detail::price_level ats::order_book_detail::price_levels< compare >↩
::price_level_type
```

6.59.1.6 reverse_iterator

```
template<typename compare = std::less<ats::price_t>>
typedef container_type::reverse_iterator ats::order_book_detail::price_levels< compare >↩
::reverse_iterator
```

6.59.2 Constructor & Destructor Documentation

6.59.2.1 price_levels()

```
template<typename compare = std::less<ats::price_t>>
ats::order_book_detail::price_levels< compare >::price_levels (
            size_t max_levels )  [inline]
```

```
            : max_levels_(max_levels), levels_() { }
```

6.59.3 Member Function Documentation

6.59.3.1 at() [1/2]

```
template<typename compare = std::less<ats::price_t>>
price_level_type& ats::order_book_detail::price_levels< compare >::at (
            ats::price_t price )  [inline]
```

```
{ return levels_.at(price); }
```

6.59.3.2 at() [2/2]

```
template<typename compare = std::less<ats::price_t>>
const price_level_type& ats::order_book_detail::price_levels< compare >::at (
            ats::price_t price ) const  [inline]
```

```
{ return levels_.at(price); }
```

6.59.3.3 begin()

```
template<typename compare = std::less<ats::price_t>>
iterator ats::order_book_detail::price_levels< compare >::begin ( )  [inline]
```

Referenced by ats::exchange_order_book::begin_ask(), and ats::exchange_order_book::begin_bid().

```
{ return levels_.begin(); }
```

6.59.3.4 cbegin()

```
template<typename compare = std::less<ats::price_t>>
const_iterator ats::order_book_detail::price_levels< compare >::cbegin ( ) const  [inline]
```

Referenced by ats::exchange_order_book::best_ask(), ats::exchange_order_book::best_bid(), ats::exchange_order_book::bid_ask_spread(), ats::exchange_order_book::cbegin_ask(), ats::exchange_order_book::cbegin_bid(), ats::exchange_order_book::midpoint(), and ats::exchange_order_book::update2().

```
{ return levels_.cbegin(); }
```

6.59.3.5 cend()

```
template<typename compare = std::less<ats::price_t>>
const_iterator ats::order_book_detail::price_levels< compare >::cend ( ) const  [inline]
```

Referenced by ats::exchange_order_book::ask_at(), ats::exchange_order_book::bid_at(), ats::exchange_order_book::cend_ask(), and ats::exchange_order_book::cend_bid().

```
{ return levels_.cend(); }
```

6.59.3.6 clear()

```
template<typename compare = std::less<ats::price_t>>
void ats::order_book_detail::price_levels< compare >::clear ( )  [inline]
```

Referenced by ats::exchange_order_book::clear().

```
{ levels_.clear(); }
```

6.59.3.7 crbegin()

```
template<typename compare = std::less<ats::price_t>>
const_reverse_iterator ats::order_book_detail::price_levels< compare >::crbegin ( ) const
[inline]
```

Referenced by ats::sim::fifo_exchange_order_book::add_order().

```
{ return levels_.crbegin(); }
```

6.59.3.8 crend()

```
template<typename compare = std::less<ats::price_t>>
const_reverse_iterator ats::order_book_detail::price_levels< compare >::crend ( )  [inline]
```

```
145 { return levels_.crend(); }
```

6.59.3.9 displayed_depth()

```
template<typename compare = std::less<ats::price_t>>
size_t ats::order_book_detail::price_levels< compare >::displayed_depth ( ) const  [inline]
```

```
168 { return max_levels_; }
```

6.59.3.10 empty()

```
template<typename compare = std::less<ats::price_t>>
bool ats::order_book_detail::price_levels< compare >::empty ( ) const  [inline]
```

Referenced by ats::sim::fifo_exchange_order_book::add_order(), ats::exchange_order_book::best_ask(), ats↩::exchange_order_book::best_bid(), ats::exchange_order_book::midpoint(), ats::level2_execution_engine::on_↩order_book_changed(), ats::level2_execution_engine::send_order(), and ats::exchange_order_book::update2().

```
153 { return levels_.empty(); }
```

6.59.3.11 end()

```
template<typename compare = std::less<ats::price_t>>
iterator ats::order_book_detail::price_levels< compare >::end ( )  [inline]
```

Referenced by ats::exchange_order_book::end_ask(), and ats::exchange_order_book::end_bid().

```
140 { return levels_.end(); }
```

6.59.3.12 find() [1/2]

```
template<typename compare = std::less<ats::price_t>>
iterator ats::order_book_detail::price_levels< compare >::find (
            ats::price_t price )  [inline]
```

Referenced by ats::exchange_order_book::ask_at(), and ats::exchange_order_book::bid_at().

```
{
    return levels_.find(price);
}
```

6.59.3.13 find() [2/2]

```
template<typename compare = std::less<ats::price_t>>
const_iterator ats::order_book_detail::price_levels< compare >::find (
            ats::price_t price ) const  [inline]
```

```
{
    return levels_.find(price);
}
```

6.59.3.14 get_level() [1/2]

```
template<typename compare = std::less<ats::price_t>>
iterator ats::order_book_detail::price_levels< compare >::get_level (
            size_t index )  [inline]
```

```
{
    if (index > levels_.size())
        return levels_.end();
    else
    {
        iterator it = levels_.begin();
        for (size_t i = 1; i < index; ++i) ++it;
        return it;
    }
}
```

6.59.3.15 get_level() [2/2]

```
template<typename compare = std::less<ats::price_t>>
const_iterator ats::order_book_detail::price_levels< compare >::get_level (
            size_t index ) const  [inline]
```

```
{
    if (index > levels_.size())
        return levels_.cend();
    else
    {
        iterator it = levels_.cbegin();
        for (size_t i = 1; i < index; ++i) ++it;
        return it;
    }
}
```

6.59.3.16 levels()

```
template<typename compare = std::less<ats::price_t>>
container_type& ats::order_book_detail::price_levels< compare >::levels ( ) [inline]
```

```
{ return levels_; }
```

6.59.3.17 operator[]() [1/2]

```
template<typename compare = std::less<ats::price_t>>
price_level_type& ats::order_book_detail::price_levels< compare >::operator[] (
             ats::price_t price ) [inline]
```

```
{ return levels_[price]; }
```

6.59.3.18 operator[]() [2/2]

```
template<typename compare = std::less<ats::price_t>>
const price_level_type& ats::order_book_detail::price_levels< compare >::operator[] (
             ats::price_t price ) const [inline]
```

```
{ return levels_[price]; }
```

6.59.3.19 rbegin()

```
template<typename compare = std::less<ats::price_t>>
reverse_iterator ats::order_book_detail::price_levels< compare >::rbegin ( ) [inline]
```

```
{ return levels_.rbegin(); }
```

6.59.3.20 rend()

```
template<typename compare = std::less<ats::price_t>>
reverse_iterator ats::order_book_detail::price_levels< compare >::rend ( ) [inline]
```

```
{ return levels_.rend(); }
```

6.59.3.21 size()

```
template<typename compare = std::less<ats::price_t>>
size_t ats::order_book_detail::price_levels< compare >::size ( ) const  [inline]
```

```
{ return levels_.size(); }
```

6.59.3.22 update()

```
template<typename compare = std::less<ats::price_t>>
void ats::order_book_detail::price_levels< compare >::update (
            const ats::level2_message & msg )  [inline]
```

Referenced by ats::exchange_order_book::update().

```
    {
        // Trade messages are not used to update order book
        if (msg.entry_type == ats::entry_type::Trade) return;

        switch (msg.update_action)
        {
        case ats::update_action::New:
            insert_level(msg.price, price_level_type(msg.price, msg.quantity, msg.order_count));
            break;
        case ats::update_action::Change:
            change_level(msg.price, msg.quantity, msg.order_count);
            break;
        case ats::update_action::Delete:
            delete_level(msg.price);
            break;
        default:
            break;
        }
    }
```

6.59.3.23 update2()

```
template<typename compare = std::less<ats::price_t>>
void ats::order_book_detail::price_levels< compare >::update2 (
            ats::level2_message & msg )  [inline]
```

Referenced by ats::exchange_order_book::update2().

```
    {
        // Trade messages are not used to update order book
        if (msg.entry_type == ats::entry_type::Trade) return;

        if (msg.update_action == ats::update_action::New)
            insert_level(msg.price, price_level_type(msg.price, msg.quantity, msg.order_count));
        else if (msg.update_action == ats::update_action::Change)
        {
            auto it = levels_.find(msg.price);
            if (it != levels_.end())
            {
                long qty_delta = msg.quantity - it->second.quantity;
```

```
                    long oct_delta = msg.order_count - it->second.order_count;

                    it->second.quantity = msg.quantity;
                    it->second.order_count = msg.order_count;

                    msg.quantity = qty_delta;
                    msg.order_count = oct_delta;
                }
                else
                {
                    // this may be dangerous as we insert if we can't find:
                    insert_level(msg.price, price_level_type(msg.price, msg.quantity, msg.order_count));
                }
            }
            else if (msg.update_action == ats::update_action::Delete)
            {
                delete_level(msg.price);
                msg.quantity = -msg.quantity;
            }
        }
```

The documentation for this class was generated from the following file:

- order_book/detail/price_levels.hpp

6.60 ats::recursive_timer Class Reference

`#include <recursive_timer.hpp>`

Collaboration diagram for ats::recursive_timer:

Public Types

- typedef std::function< void(const ats::timestamp_t &)> time_listener

Public Member Functions

- recursive_timer (const boost::posix_time::time_duration &period=boost::posix_time::seconds(1))
- void init (const boost::posix_time::time_duration &period)
- void update (const ats::timestamp_t &time)
- void add_time_listener (const time_listener &listener)

6.60.1 Member Typedef Documentation

6.60.1.1 time_listener

`typedef std::function<void(const ats::timestamp_t&)> ats::recursive_timer::time_listener`

6.60.2 Constructor & Destructor Documentation

6.60.2.1 recursive_timer()

```
ats::recursive_timer::recursive_timer (
            const boost::posix_time::time_duration & period = boost::posix_time::seconds(1) )
[inline]
```

```
            : period_(period) { }
```

6.60.3 Member Function Documentation

6.60.3.1 add_time_listener()

```
void ats::recursive_timer::add_time_listener (
            const time_listener & listener )  [inline]
```

```
{ time_listeners_.push_back(listener); }
```

6.60.3.2 init()

```
void ats::recursive_timer::init (
            const boost::posix_time::time_duration & period )  [inline]
```

```
{ period_ = period; }
```

6.60.3.3 update()

```
void ats::recursive_timer::update (
            const ats::timestamp_t & time )  [inline]
```

References ats::date_time::date_time::date(), and ats::date_time::date_time::is_not_a_date_time().

```
    {
        if (end_time_.is_not_a_date_time() || time.date() != end_time_.date())
        {
            end_time_ = time.date();
            while (end_time_ + period_ < time)
                end_time_ += period_;

            invoke_listeners(end_time_);
        }
        else if (time - end_time_ >= period_)
        {
            end_time_ += period_;
            for (; end_time_ + period_ < time; end_time_ += period_)
                invoke_listeners(end_time_);

            invoke_listeners(end_time_);
        }
    }
```

The documentation for this class was generated from the following file:

- recursive_timer.hpp

6.61 ats::report_engine Class Reference

```
#include <report_engine.hpp>
```

Collaboration diagram for ats::report_engine:

Public Member Functions

- void add_pnl_item (const ats::pnl_item &item)
- iterator begin ()
- const_iterator cbegin () const
- iterator end ()
- const_iterator cend () const

6.61.1 Member Function Documentation

6.61.1.1 add_pnl_item()

```
void ats::report_engine::add_pnl_item (
             const ats::pnl_item & item )  [inline]
```

```
        {
            performance_.push_back(item);
        }
```

6.61.1.2 begin()

```
iterator ats::report_engine::begin ( )  [inline]
```

```
{ return performance_.begin(); }
```

6.61.1.3 cbegin()

```
const_iterator ats::report_engine::cbegin ( ) const  [inline]
```

```
{ return performance_.cbegin(); }
```

6.61.1.4 cend()

```
const_iterator ats::report_engine::cend ( ) const  [inline]
```

```
{ return performance_.cend(); }
```

6.61.1.5 end()

```
iterator ats::report_engine::end ( )  [inline]
```

```
{ return performance_.end(); }
```

The documentation for this class was generated from the following file:

- report/report_engine.hpp

6.62 ats::security_base Class Reference

```
#include <security_base.hpp>
```

Collaboration diagram for ats::security_base:

Public Types

- typedef boost::circular_buffer< ats::bar > bar_container
- typedef std::function< void(const ats::timestamp_t &)> time_listener

Public Member Functions

- security_base (const ats::symbol_key &symbol, portfolio_base *portfolio)
- security_base (const ats::symbol_key &symbol, portfolio_base *portfolio, const boost::posix_time::time_duration bar_periodicity, size_t bars_to_store)
- virtual ∼security_base ()
- std::ofstream & LOG ()
- const ats::timestamp_t & current_time () const
- const ats::position & get_position () const
- const ats::order_book & get_order_book (const ats::symbol_key &symbol) const
- const ats::symbol_key & symbol () const
- const ats::order_book & order_book () const
- const ats::exchange_order_book * exchange_order_book (const std::string &exchange) const
- const ats::timestamp_t & last_update_time () const
- const ats::price_t & last_price () const
- long get_inventory () const
- void process_time_update (const ats::timestamp_t &time)
- void process_message (const ats::level2_message_packet &msg)

- void create_order_book (const std::string &exchange, size_t book_depth)
- virtual void on_init ()
- virtual void on_exit ()
- virtual void on_order_status_changed (const ats::order_status_message &msg)
- virtual void on_order_book_changed (const ats::level2_message_packet &msg)
- virtual void on_trade (const ats::trade_message &msg)
- virtual void on_bar_open (const ats::bar &bar)
- virtual void on_bar_close (const ats::bar &bar)
- const bar_container & bars () const
- const ats::bar ∗ current_bar () const
- void set_bar_parameters (const boost::posix_time::time_duration &bar_periodicity, size_t bars_to_store)
- void update_bars (const ats::timestamp_t &time, ats::price_t price, long quantity)
- void add_time_listener (const time_listener &listener)

Public Attributes

- ats::symbol_key symbol_
- ats::order_book order_book_
- bar_container bars_
- boost::posix_time::time_duration bar_periodicity_
- ats::timestamp_t last_update_time_
- ats::price_t last_price_ = 0
- std::list< time_listener > time_listeners_

Protected Attributes

- ats::portfolio_base ∗ portfolio_

6.62.1 Member Typedef Documentation

6.62.1.1 bar_container

```
typedef boost::circular_buffer<ats::bar> ats::security_base::bar_container
```

6.62.1.2 time_listener

```
typedef std::function<void(const ats::timestamp_t&)> ats::security_base::time_listener
```

6.62.2 Constructor & Destructor Documentation

6.62.2.1 security_base() [1/2]

```
ats::security_base::security_base (
              const ats::symbol_key & symbol,
              portfolio_base * portfolio )  [inline]
```

References on_init().

```
        : symbol_(symbol), order_book_(symbol), portfolio_(portfolio)
    {
        on_init();
    }
```

6.62.2.2 security_base() [2/2]

```
ats::security_base::security_base (
              const ats::symbol_key & symbol,
              portfolio_base * portfolio,
              const boost::posix_time::time_duration bar_periodicity,
              size_t bars_to_store )  [inline]
```

References on_init().

```
        : symbol_(symbol), order_book_(symbol), portfolio_(portfolio),
          bars_(bars_to_store), bar_periodicity_(bar_periodicity)
    {
        on_init();
    }
```

6.62.2.3 ∼security_base()

```
virtual ats::security_base::~security_base ( )  [inline], [virtual]
```

References current_time(), get_order_book(), get_position(), LOG(), on_exit(), and symbol().

```
    {
        on_exit();
    }
```

6.62.3 Member Function Documentation

6.62.3.1 add_time_listener()

```
void ats::security_base::add_time_listener (
             const time_listener & listener )  [inline]
```

References time_listeners_.

```
{ time_listeners_.push_back(listener); }
```

6.62.3.2 bars()

```
const bar_container& ats::security_base::bars ( ) const  [inline]
```

References bars_.

```
{ return bars_; }
```

6.62.3.3 create_order_book()

```
void ats::security_base::create_order_book (
             const std::string & exchange,
             size_t book_depth )  [inline]
```

References ats::order_book::add_order_book(), and order_book_.

```
{
    order_book_.add_order_book(exchange, book_depth);
}
```

6.62.3.4 current_bar()

```
const ats::bar* ats::security_base::current_bar ( ) const  [inline]
```

References bars_.

```
{
    return bars_.size() != 0 ? &bars_[0] : nullptr;
}
```

6.62.3.5 current_time()

```
const ats::timestamp_t & ats::security_base::current_time ( ) const
```

References ats::portfolio_base::current_time(), and portfolio_.

Referenced by ~security_base().

```
{
    return portfolio_->current_time();
}
```

6.62.3.6 exchange_order_book()

```
const ats::exchange_order_book* ats::security_base::exchange_order_book (
             const std::string & exchange ) const  [inline]
```

References ats::order_book::get(), and order_book_.

```
{ return order_book_.get(exchange); }
```

6.62.3.7 get_inventory()

```
long ats::security_base::get_inventory ( ) const  [inline]
```

References get_position(), and ats::position::inventory().

```
{ return get_position().inventory(); }
```

6.62.3.8 get_order_book()

```
const ats::order_book & ats::security_base::get_order_book (
             const ats::symbol_key & symbol ) const
```

References ats::portfolio_base::get_order_book(), and portfolio_.

Referenced by ~security_base().

```
{
    return portfolio_->get_order_book(symbol);
}
```

6.62.3.9 get_position()

```
const ats::position & ats::security_base::get_position ( ) const
```

References ats::portfolio_base::get_position(), portfolio_, and symbol().

Referenced by get_inventory(), and ∼security_base().

```
{
    return portfolio_->get_position(symbol());
}
```

6.62.3.10 last_price()

```
const ats::price_t& ats::security_base::last_price ( ) const  [inline]
```

References last_price_.

```
{ return last_price_; }
```

6.62.3.11 last_update_time()

```
const ats::timestamp_t& ats::security_base::last_update_time ( ) const  [inline]
```

References last_update_time_.

```
{ return last_update_time_; }
```

6.62.3.12 LOG()

```
std::ofstream & ats::security_base::LOG ( )
```

References ats::portfolio_base::LOG(), and portfolio_.

Referenced by ∼security_base().

```
{
    return portfolio_->LOG();
}
```

6.62.3.13 on_bar_close()

```
virtual void ats::security_base::on_bar_close (
             const ats::bar & bar )  [inline], [virtual]
```

Referenced by update_bars().

```
114 { }
```

6.62.3.14 on_bar_open()

```
virtual void ats::security_base::on_bar_open (
             const ats::bar & bar )  [inline], [virtual]
```

Referenced by update_bars().

```
113 { }
```

6.62.3.15 on_exit()

```
virtual void ats::security_base::on_exit ( )  [inline], [virtual]
```

Referenced by ∼security_base().

```
106 { }
```

6.62.3.16 on_init()

```
virtual void ats::security_base::on_init ( )  [inline], [virtual]
```

Referenced by security_base().

```
105 { }
```

6.62.3.17 on_order_book_changed()

```
virtual void ats::security_base::on_order_book_changed (
             const ats::level2_message_packet & msg )  [inline], [virtual]
```

Referenced by process_message().

```
{ }
```

6.62.3.18 on_order_status_changed()

```
virtual void ats::security_base::on_order_status_changed (
             const ats::order_status_message & msg )  [inline], [virtual]
```

```
{ }
```

6.62.3.19 on_trade()

```
virtual void ats::security_base::on_trade (
             const ats::trade_message & msg )  [inline], [virtual]
```

```
{ }
```

6.62.3.20 order_book()

```
const ats::order_book& ats::security_base::order_book ( ) const  [inline]
```

References order_book_.

```
{ return order_book_; }
```

6.62.3.21 process_message()

```
void ats::security_base::process_message (
             const ats::level2_message_packet & msg )  [inline]
```

References last_price_, last_update_time_, ats::instrument_message_packet< MessageT >::messages, on_order_book_changed(), order_book_, process_time_update(), ats::message::time, ats::Trade, ats::order_book::update2(), and update_bars().

```
{
    // Update the time of the last received message
    last_update_time_ = msg.time;

    // Transform the message into a "message with deltas"
    ats::level2_message_packet msg_delta = msg;
    for (auto& m : msg_delta.messages)
    {
        order_book_.update2(m);
        if (m.entry_type == ats::entry_type::Trade)
        {
            update_bars(m.time, m.price, m.quantity);
            last_price_ = m.price;
        }
    }

    process_time_update(msg.time);

    // Respond to the new message
    on_order_book_changed(msg_delta);
}
```

6.62.3.22 process_time_update()

```
void ats::security_base::process_time_update (
             const ats::timestamp_t & time )  [inline]
```

References time_listeners_.

Referenced by process_message().

```
{
    // Notify all interested listeners about the reception of a new message
    for (const auto& l : time_listeners_)
        l(time);
}
```

6.62.3.23 set_bar_parameters()

```
void ats::security_base::set_bar_parameters (
             const boost::posix_time::time_duration & bar_periodicity,
             size_t bars_to_store )  [inline]
```

References bar_periodicity_, bars_, and update_bars().

```
{
    bar_periodicity_ = bar_periodicity;
    bars_.set_capacity(bars_to_store);
}
```

6.62.3.24 symbol()

```
const ats::symbol_key& ats::security_base::symbol ( ) const  [inline]
```

References symbol_.

Referenced by ats::portfolio_base::add_security(), get_position(), and ∼security_base().

```
{ return symbol_; }
```

6.62.3.25 update_bars()

```
void ats::security_base::update_bars (
            const ats::timestamp_t & time,
            ats::price_t price,
            long quantity )  [inline]
```

References bar_periodicity_, bars_, ats::bar::close, ats::date_time::date_time::date(), ats::bar::high, ats::bar::low, on_bar_close(), on_bar_open(), ats::bar::open, ats::bar::quantity, and ats::bar::time_open.

Referenced by process_message(), and set_bar_parameters().

```
    {
        if (bars_.capacity() == 0) return;

        // NOTE: New bars will always be pushed forward so that bars_[0] was the last bar

        if (bars_.empty())
        {
            ats::date_time::date_time time_close = time.date();
            while (time_close <= time)
                time_close += bar_periodicity_;

            ats::bar bar;
            bar.open = bar.high = bar.low = bar.close = price;
            bar.quantity = quantity;
            bar.time_open = time_close - bar_periodicity_;
            bars_.push_front(bar);

            on_bar_open(bar);
        }
        else
        {
            ats::bar& last_bar = *bars_.begin();
            if (time - last_bar.time_open < bar_periodicity_)
            {
                last_bar.quantity += quantity;
                last_bar.close = price;
                if (price < last_bar.low)
                    last_bar.low = price;
                else if (price > last_bar.high)
                    last_bar.high = price;
            }
            else
            {
                on_bar_close(last_bar);

                ats::bar new_bar;
                new_bar.open = new_bar.high = new_bar.low = new_bar.close = price;
                new_bar.quantity = quantity;
                new_bar.time_open = last_bar.time_open + bar_periodicity_;

                // Insert empty bars if there were no trades for a long time
                ats::date_time::date_time time_close(new_bar.time_open + bar_periodicity_);
                while (time_close < time)
                {
                    ats::bar empty_bar;
                    empty_bar.open = empty_bar.high = empty_bar.low = empty_bar.close = last_bar.close;
                    empty_bar.quantity = 0;
                    empty_bar.time_open = new_bar.time_open;
                    bars_.push_front(empty_bar);

                    on_bar_close(empty_bar);

                    new_bar.time_open = time_close;
                    time_close += bar_periodicity_;
                }

                bars_.push_front(new_bar);

                on_bar_open(new_bar);
            }
        }
    }
```

6.62.4 Member Data Documentation

6.62.4.1 bar_periodicity_

`boost::posix_time::time_duration ats::security_base::bar_periodicity_`

Referenced by set_bar_parameters(), and update_bars().

6.62.4.2 bars_

`bar_container ats::security_base::bars_`

Referenced by bars(), current_bar(), set_bar_parameters(), and update_bars().

6.62.4.3 last_price_

`ats::price_t ats::security_base::last_price_ = 0`

Referenced by last_price(), and process_message().

6.62.4.4 last_update_time_

`ats::timestamp_t ats::security_base::last_update_time_`

Referenced by last_update_time(), and process_message().

6.62.4.5 order_book_

```
ats::order_book ats::security_base::order_book_
```

Referenced by create_order_book(), exchange_order_book(), order_book(), and process_message().

6.62.4.6 portfolio_

```
ats::portfolio_base* ats::security_base::portfolio_ [protected]
```

Referenced by current_time(), get_order_book(), get_position(), and LOG().

6.62.4.7 symbol_

```
ats::symbol_key ats::security_base::symbol_
```

Referenced by symbol().

6.62.4.8 time_listeners_

```
std::list<time_listener> ats::security_base::time_listeners_
```

Referenced by add_time_listener(), and process_time_update().

The documentation for this class was generated from the following files:

- security/security_base.hpp
- security/security_base.cpp

6.63 ats::security_container Class Reference

```
#include <security_container.hpp>
```

Collaboration diagram for ats::security_container:

Public Types

- typedef std::shared_ptr< ats::security_base > security_ptr
- typedef std::unordered_multimap< std::string, security_ptr > container_type
- typedef container_type::iterator iterator
- typedef container_type::const_iterator const_iterator
- typedef std::pair< std::string, std::string > key_type

Public Member Functions

- security_container & operator+= (const security_ptr &sec)
- security_container & operator+= (security_ptr &&sec)
- iterator find (const std::string &symbol, const std::string &exchange)
- std::pair< iterator, iterator > equal_range (const std::string &symbol)
- iterator begin ()
- const_iterator cbegin () const
- iterator end ()
- const_iterator cend () const

6.63.1 Member Typedef Documentation

6.63.1.1 const_iterator

```
typedef container_type::const_iterator ats::security_container::const_iterator
```

6.63.1.2 container_type

```
typedef std::unordered_multimap<std::string, security_ptr> ats::security_container::container↵
_type
```

6.63.1.3 iterator

```
typedef container_type::iterator ats::security_container::iterator
```

6.63.1.4 key_type

```
typedef std::pair<std::string, std::string> ats::security_container::key_type
```

6.63.1.5 security_ptr

```
typedef std::shared_ptr<ats::security_base> ats::security_container::security_ptr
```

6.63.2 Member Function Documentation

6.63.2.1 begin()

```
iterator ats::security_container::begin ( )  [inline]
```

```
{ return securities_.begin(); }
```

6.63.2.2 cbegin()

```
const_iterator ats::security_container::cbegin ( ) const  [inline]
```

```
{ return securities_.cbegin(); }
```

6.63.2.3 cend()

```
const_iterator ats::security_container::cend ( ) const  [inline]
```

```
{ return securities_.cend(); }
```

6.63.2.4 end()

```
iterator ats::security_container::end ( )  [inline]
```

```
{ return securities_.end(); }
```

6.63.2.5 equal_range()

```
std::pair<iterator, iterator> ats::security_container::equal_range (
             const std::string & symbol )  [inline]
```

```
        {
            return securities_.equal_range(symbol);
        }
```

6.63.2.6 find()

```
iterator ats::security_container::find (
            const std::string & symbol,
            const std::string & exchange )  [inline]
```

```
        {
            auto find = iterators_.find(std::make_pair(symbol, exchange));
            return find != iterators_.end() ? find->second : securities_.end();
        }
```

6.63.2.7 operator+=() [1/2]

```
security_container& ats::security_container::operator+= (
            const security_ptr & sec )  [inline]
```

```
        {
            auto it = securities_.insert(container_type::value_type(sec->symbol().to_string(), sec));
//          auto key = std::make_pair(sec->symbol().symbol, sec->exchange());
//          iterators_.insert(iterator_container::value_type(key, it));
            return *this;
        }
```

6.63.2.8 operator+=() [2/2]

```
security_container& ats::security_container::operator+= (
            security_ptr && sec )  [inline]
```

```
        {
//          auto key = std::make_pair(sec->symbol().symbol, sec->exchange());
//          auto it = securities_.insert(container_type::value_type(key.first, std::move(sec)));
//          iterators_.insert(iterator_container::value_type(key, it));
            return *this;
        }
```

The documentation for this class was generated from the following file:

- container/security_container.hpp

6.64 ats::sim::sim_book Class Reference

```
#include <sim_book.hpp>
```

Collaboration diagram for ats::sim::sim_book:

Public Types

- typedef price_levels< std::greater< ats::price_t > > bid_container
- typedef price_levels< std::less< ats::price_t > > ask_container
- typedef std::unordered_map< ats::orderid_t, price_level::iterator > order_container

Public Member Functions

- void add_order (const ats::limit_order &order)
- void insert_order (const ats::limit_order &order)
- void cancel_order (const ats::orderid_t &id, const ats::timestamp_t &time)
- void process_trade (ats::price_t price, long quantity, const ats::timestamp_t &time)
- void add_order_status_listener (const ats::order_status_handler &listener)
- const price_level ∗ best_bid () const
- price_level ∗ best_bid ()
- const price_level ∗ best_ask () const
- price_level ∗ best_ask ()
- const price_level ∗ get_level (ats::price_t price, bool is_bid) const
- const order_container & get_sim_orders () const
- void clean_level (ats::price_t, bool is_bid)
- void execute_all_orders (ats::price_t price, const ats::timestamp_t &time, bool is_bid)
- void execute_orders (ats::price_t price, long quantity, const ats::timestamp_t &time, bool is_bid)
- void execute_crosses (const ats::price_t ∗bid, const ats::price_t ∗ask, const ats::timestamp_t &time)
- void process_change_msg (const ats::level2_message &msg)
- void process_delete_msg (const ats::level2_message &msg)
- void process_trade_msg (const ats::level2_message &msg)
- void process_insert_msg (const ats::level2_message &msg)
- void process_level2_msg (const ats::level2_message &msg)

6.64.1 Member Typedef Documentation

6.64.1.1 ask_container

```
typedef price_levels<std::less<ats::price_t> > ats::sim::sim_book::ask_container
```

6.64.1.2 bid_container

```
typedef price_levels<std::greater<ats::price_t> > ats::sim::sim_book::bid_container
```

6.64.1.3 order_container

```
typedef std::unordered_map<ats::orderid_t, price_level::iterator> ats::sim::sim_book::order_←
container
```

6.64.2 Member Function Documentation

6.64.2.1 add_order()

```
void ats::sim::sim_book::add_order (
            const ats::limit_order & order )  [inline]
```

References ats::sim::price_levels< comp >::add_order(), best_ask(), best_bid(), ats::Buy, ats::BuyCover, execute_all_orders(), execute_orders(), ats::order::id(), ats::limit_order::price(), ats::sim::price_level::price(), ats::order::quantity(), ats::sim::price_level::quantity, ats::order::side(), ats::order::symbol(), ats::order::time_in_force(), and ats::order::transact_time.

Referenced by ats::sim::fifo_exchange_order_book::add_order().

```
{
    // First, check for crosses, then add if there still is a quantity left
    price_level::iterator order_pos;
    if (order.side() == ats::order_side::Buy || order.side() == ats::order_side::BuyCover)
    {
        if (best_ask() != nullptr && order.price() >= best_ask()->price())
        {
            long unexecuted_qty = order.quantity();
            while (best_ask() != nullptr && order.price() >= best_ask()->price() && unexecuted_qty > 0)
            {
                if (unexecuted_qty <= best_ask()->quantity)
                {
                    ats::order_status_filled_message msg(order.id(), order.transact_time, best_ask()->price(), unexecuted_qty);
                    execute_orders(best_ask()->price(), unexecuted_qty, order.transact_time, false);

                    if (order_status_listener_ != nullptr)
                        order_status_listener_(msg);
                    return;
                }
                else
                {
                    ats::order_status_partially_filled_message msg(order.id(), order.transact_time, best_ask()->price(), best_ask()->quantity);
                    unexecuted_qty -= best_ask()->quantity;
                    execute_all_orders(best_ask()->price(), order.transact_time, false);

                    if (order_status_listener_ != nullptr)
                        order_status_listener_(msg);
                }
            }

            if (unexecuted_qty > 0)
            {
                ats::limit_order reduced_order(order.id(), order.symbol(), unexecuted_qty, order.side(),
                        order.time_in_force(), order.price());
                order_pos = bids_.add_order(reduced_order);
            }
        }
        else
            order_pos = bids_.add_order(order);
    }
    else
    {
        if (best_bid() != nullptr && order.price() <= best_bid()->price())
        {
            long unexecuted_qty = order.quantity();
            while (best_bid() != nullptr && order.price() <= best_bid()->price() && unexecuted_qty > 0)
            {
                if (unexecuted_qty <= best_bid()->quantity)
```

```
                    {
                        ats::order_status_filled_message msg(order.id(), order.transact_time, best_bid()->price(), unexecuted_qty);
                        execute_orders(best_bid()->price(), unexecuted_qty, order.transact_time, true);

                        if (order_status_listener_ != nullptr)
                            order_status_listener_(msg);
                        return;
                    }
                    else
                    {
                        ats::order_status_partially_filled_message msg(order.id(), order.transact_time, best_bid()->price(), best_bid()->quantity);
                        unexecuted_qty -= best_bid()->quantity;
                        execute_all_orders(best_bid()->price(), order.transact_time, true);

                        if (order_status_listener_ != nullptr)
                            order_status_listener_(msg);
                    }
                }

                if (unexecuted_qty > 0)
                {
                    ats::limit_order reduced_order(order.id(), order.symbol(), unexecuted_qty, order.side(),
                        order.time_in_force(), order.price());
                    order_pos = asks_.add_order(reduced_order);
                }
            }
            else
                order_pos = asks_.add_order(order);
        }

        // For bookkeeping
        if (order.id() != 0)
            sim_orders_.insert(std::make_pair(order.id(), order_pos));
    }
```

6.64.2.2 add_order_status_listener()

void ats::sim::sim_book::add_order_status_listener (
    const ats::order_status_handler & *listener*)   [inline]

References ats::sim::price_levels< comp >::add_order_status_listener().

Referenced by ats::sim::fifo_exchange_order_book::add_order_status_listener().

```
    {
        order_status_listener_ = listener;
        bids_.add_order_status_listener(listener);
        asks_.add_order_status_listener(listener);
    }
```

6.64.2.3 best_ask() [1/2]

const price_level* ats::sim::sim_book::best_ask ( ) const   [inline]

References ats::sim::price_levels< comp >::top_level().

Referenced by add_order(), execute_crosses(), process_change_msg(), process_trade(), and process_trade_msg().

```
{ return asks_.top_level(); }
```

6.64.2.4 best_ask() [2/2]

```
price_level* ats::sim::sim_book::best_ask ( )  [inline]
```

References ats::sim::price_levels< comp >::top_level().

```
{ return asks_.top_level(); }
```

6.64.2.5 best_bid() [1/2]

```
const price_level* ats::sim::sim_book::best_bid ( ) const  [inline]
```

References ats::sim::price_levels< comp >::top_level().

Referenced by add_order(), execute_crosses(), process_change_msg(), process_trade(), and process_trade_msg().

```
{ return bids_.top_level(); }
```

6.64.2.6 best_bid() [2/2]

```
price_level* ats::sim::sim_book::best_bid ( )  [inline]
```

References ats::sim::price_levels< comp >::top_level().

```
{ return bids_.top_level(); }
```

6.64.2.7 cancel_order()

```
void ats::sim::sim_book::cancel_order (
            const ats::orderid_t & id,
            const ats::timestamp_t & time )  [inline]
```

References ats::Buy, ats::BuyCover, and ats::sim::price_levels< comp >::cancel_order().

Referenced by ats::sim::fifo_exchange_order_book::cancel_order().

```
 {
     auto find = sim_orders_.find(id);
     if (find == sim_orders_.end()) return;

     auto pos = find->second;
     if (pos->side() == ats::order_side::Buy || pos->side() ==
ats::order_side::BuyCover)
         bids_.cancel_order(pos, time);
     else
         asks_.cancel_order(pos, time);

     sim_orders_.erase(find);

     if (order_status_listener_ != nullptr)
         order_status_listener_(ats::order_status_cancelled_message(
id, time, "Canceled by trader"));
 }
```

6.64.2.8 clean_level()

```
void ats::sim::sim_book::clean_level (
               ats::price_t price,
               bool is_bid )  [inline]
```

References ats::sim::price_levels< comp >::clean().

Referenced by get_sim_orders().

```
{
    if (is_bid)
        bids_.clean(price);
    else
        asks_.clean(price);
}
```

6.64.2.9 execute_all_orders()

```
void ats::sim::sim_book::execute_all_orders (
               ats::price_t price,
               const ats::timestamp_t & time,
               bool is_bid )  [inline]
```

References ats::sim::price_levels< comp >::execute_all_orders().

Referenced by add_order(), execute_crosses(), get_sim_orders(), process_change_msg(), and process_trade().

```
{
    if (is_bid)
        bids_.execute_all_orders(price, time, sim_orders_);
    else
        asks_.execute_all_orders(price, time, sim_orders_);
}
```

6.64.2.10 execute_crosses()

```
void ats::sim::sim_book::execute_crosses (
               const ats::price_t * bid,
               const ats::price_t * ask,
               const ats::timestamp_t & time )  [inline]
```

References best_ask(), best_bid(), and execute_all_orders().

Referenced by get_sim_orders(), and ats::sim::fifo_exchange_order_book::update().

```
{
    if (bid != nullptr)
    {
        while (best_ask() != nullptr && best_ask()->price() <= *bid)
            execute_all_orders(best_ask()->price(), time, false);
    }

    if (ask != nullptr)
    {
        while (best_bid() != nullptr && best_bid()->price() >= *ask)
            execute_all_orders(best_bid()->price(), time, true);
    }
}
```

6.64.2.11 execute_orders()

```
void ats::sim::sim_book::execute_orders (
            ats::price_t price,
            long quantity,
            const ats::timestamp_t & time,
            bool is_bid )  [inline]
```

References ats::sim::price_levels< comp >::execute_orders().

Referenced by add_order(), and get_sim_orders().

```
    {
        if (is_bid)
            bids_.execute_orders(price, quantity, time, sim_orders_);
        else
            asks_.execute_orders(price, quantity, time, sim_orders_);
    }
```

6.64.2.12 get_level()

```
const price_level* ats::sim::sim_book::get_level (
            ats::price_t price,
            bool is_bid ) const  [inline]
```

References ats::sim::price_levels< comp >::get_level().

Referenced by ats::sim::fifo_exchange_order_book::add_order(), and ats::sim::fifo_exchange_order_book::get_level().

```
        { return is_bid ? bids_.get_level(price) : asks_.get_level(price); }
```

6.64.2.13 get_sim_orders()

```
const order_container& ats::sim::sim_book::get_sim_orders ( ) const  [inline]
```

References clean_level(), execute_all_orders(), execute_crosses(), execute_orders(), process_change_msg(), process_delete_msg(), process_insert_msg(), process_level2_msg(), and process_trade_msg().

Referenced by ats::sim::fifo_exchange_order_book::get_sim_orders().

```
{ return sim_orders_; }
```

6.64.2.14 insert_order()

```
void ats::sim::sim_book::insert_order (
            const ats::limit_order & order )  [inline]
```

References ats::Buy, ats::BuyCover, ats::order::id(), ats::sim::price_levels< comp >::insert_order(), and ats::order::side().

Referenced by ats::sim::fifo_exchange_order_book::add_order().

```
    {
        price_level::iterator order_pos;
        if (order.side() == ats::order_side::Buy || order.
  side() == ats::order_side::BuyCover)
            order_pos = bids_.insert_order(order);
        else
            order_pos = asks_.insert_order(order);

        if (order.id() != 0)
            sim_orders_.insert(std::make_pair(order.id(), order_pos));
    }
```

6.64.2.15 process_change_msg()

```
void ats::sim::sim_book::process_change_msg (
            const ats::level2_message & msg )  [inline]
```

References best_ask(), best_bid(), ats::Bid, ats::level2_message::entry_type, execute_all_orders(), ats::level2_message::price, ats::sim::price_levels< comp >::process_change_msg(), and ats::message::time.

Referenced by get_sim_orders(), ats::sim::fifo_exchange_order_book::process_change_msg(), and process_level2_msg().

```
    {
        if (msg.entry_type == ats::entry_type::Bid)
        {
            while (best_ask() != nullptr && msg.price >= best_ask()->price())
                execute_all_orders(best_ask()->price(), msg.
  time, false);

            bids_.process_change_msg(msg, sim_orders_);
        }
        else
        {
            while (best_bid() != nullptr && msg.price < best_bid()->price())
                execute_all_orders(best_bid()->price(), msg.
  time, true);

            asks_.process_change_msg(msg, sim_orders_);
        }
    }
```

6.64.2.16 process_delete_msg()

```
void ats::sim::sim_book::process_delete_msg (
            const ats::level2_message & msg )  [inline]
```

References ats::Bid, ats::level2_message::entry_type, and ats::sim::price_levels< comp >::process_delete_msg().

Referenced by get_sim_orders(), ats::sim::fifo_exchange_order_book::process_delete_msg(), and process_level2_msg().

```
{
    if (msg.entry_type == ats::entry_type::Bid)
        bids_.process_delete_msg(msg, sim_orders_);
    else
        asks_.process_delete_msg(msg, sim_orders_);
}
```

6.64.2.17 process_insert_msg()

```
void ats::sim::sim_book::process_insert_msg (
            const ats::level2_message & msg )  [inline]
```

References ats::Bid, ats::level2_message::entry_type, and ats::sim::price_levels< comp >::process_insert_msg().

Referenced by get_sim_orders(), ats::sim::fifo_exchange_order_book::process_insert_msg(), and process_level2_msg().

```
{
    if (msg.entry_type == ats::entry_type::Bid)
        bids_.process_insert_msg(msg);
    else
        asks_.process_insert_msg(msg);
}
```

6.64.2.18 process_level2_msg()

```
void ats::sim::sim_book::process_level2_msg (
            const ats::level2_message & msg )  [inline]
```

References ats::Change, ats::Delete, ats::level2_message::entry_type, ats::New, process_change_msg(), process_delete_msg(), process_insert_msg(), process_trade_msg(), ats::Trade, and ats::level2_message::update_action.

Referenced by get_sim_orders(), and ats::sim::fifo_exchange_order_book::update().

```
{
    if (msg.entry_type == ats::entry_type::Trade)
        process_trade_msg(msg);
    else
    {
        switch (msg.update_action)
        {
        case ats::update_action::Change:
            process_change_msg(msg);
            break;
        case ats::update_action::Delete:
            process_delete_msg(msg);
            break;
        case ats::update_action::New:
            process_insert_msg(msg);
            break;
        default:
            break;
        }
    }
}
```

6.64.2.19 process_trade()

```
void ats::sim::sim_book::process_trade (
            ats::price_t price,
            long quantity,
            const ats::timestamp_t & time ) [inline]
```

References best_ask(), best_bid(), and execute_all_orders().

Referenced by process_trade_msg().

```
    {
        // Check for crosses
        while (best_bid() != nullptr && price < best_bid()->price())
            execute_all_orders(best_bid()->price(), time, true);

        while (best_ask() != nullptr && price > best_ask()->price())
            execute_all_orders(best_ask()->price(), time, false);

        if (best_bid() != nullptr && price == best_bid()->price())
        {

        }
        else if (best_ask() != nullptr && price == best_ask()->price())
        {

        }
    }
```

6.64.2.20 process_trade_msg()

```
void ats::sim::sim_book::process_trade_msg (
            const ats::level2_message & msg ) [inline]
```

References best_ask(), best_bid(), ats::level2_message::price, process_trade(), ats::level2_message::quantity, ats::message::time, and ats::sim::price_level::traded_quantity.

Referenced by get_sim_orders(), process_level2_msg(), and ats::sim::fifo_exchange_order_book::process_trade_msg().

```
    {
        process_trade(msg.price, msg.quantity, msg.
    time);

        if (best_bid() != nullptr && msg.price == best_bid()->price())
            best_bid()->traded_quantity = msg.quantity;
        else if (best_ask() != nullptr && msg.price == best_ask()->price())
            best_ask()->traded_quantity = msg.quantity;
    }
```

The documentation for this class was generated from the following file:

- order_book/simulation/sim_book.hpp

6.65 ats::single_message_reader< MessageT > Class Template Reference

```
#include <message_reader.hpp>
```

Inheritance diagram for ats::single_message_reader< MessageT >:

Collaboration diagram for ats::single_message_reader< MessageT >:

Public Member Functions

- virtual ~single_message_reader ()
- virtual void send_message (ats::portfolio_base *universe) const override
- virtual const ats::message & get_last_message () const override

Protected Attributes

- MessageT message_

6.65.1 Constructor & Destructor Documentation

6.65.1.1 ~single_message_reader()

```
template<typename MessageT >
virtual ats::single_message_reader< MessageT >::~single_message_reader ( )  [inline], [virtual]
```

```
{ }
```

6.65.2 Member Function Documentation

6.65.2.1 get_last_message()

```
template<typename MessageT >
virtual const ats::message& ats::single_message_reader< MessageT >::get_last_message ( ) const
[inline], [override], [virtual]
```

Implements ats::message_reader.

```
{
    return static_cast<const ats::message&>(message_);
}
```

6.65.2.2 send_message()

```
template<typename MessageT >
virtual void ats::single_message_reader< MessageT >::send_message (
             ats::portfolio_base * universe ) const  [inline], [override], [virtual]
```

Implements ats::message_reader.

References ats::multievent_handler::invoke().

```
{
    universe->invoke(message_);
}
```

6.65.3 Member Data Documentation

6.65.3.1 message_

```
template<typename MessageT >
MessageT ats::single_message_reader< MessageT >::message_  [protected]
```

The documentation for this class was generated from the following file:

- data_feed/historical/message_reader.hpp

6.66 ats::stop_order Class Reference

```
#include <stop_order.hpp>
```

Inheritance diagram for ats::stop_order:

Collaboration diagram for ats::stop_order:

Public Member Functions

- stop_order (const ats::orderid_t &id, const std::string &symbol, long quantity, ats::order_side side, ats::order_time_in_force time_in_force, ats::price_t stop_price)
- ats::price_t price () const

Additional Inherited Members

6.66.1 Constructor & Destructor Documentation

6.66.1.1 stop_order()

```
ats::stop_order::stop_order (
             const ats::orderid_t & id,
             const std::string & symbol,
             long quantity,
             ats::order_side side,
             ats::order_time_in_force time_in_force,
             ats::price_t stop_price )  [inline]
```

```
15            : ats::order(id, symbol, quantity, side,
     time_in_force), price_(stop_price) { }
```

6.66.2 Member Function Documentation

6.66.2.1 price()

```
ats::price_t ats::stop_order::price ( ) const  [inline]
```

Referenced by ats::level2_execution_engine::send_order().

```
{ return price_; }
```

The documentation for this class was generated from the following file:

- order/stop_order.hpp

6.67 ats::date_time::stop_watch Class Reference

```
#include <stop_watch.hpp>
```

Collaboration diagram for ats::date_time::stop_watch:

Public Member Functions

- stop_watch ()
- void start ()
- void stop ()
- std::chrono::duration< std::chrono::system_clock::rep, std::chrono::system_clock::period > elapsed ()
- long long milliseconds ()
- long long microseconds ()
- long long seconds ()

6.67.1 Constructor & Destructor Documentation

6.67.1.1 stop_watch()

```
ats::date_time::stop_watch::stop_watch ( )  [inline]
```

References start(), and stop().

```
        {
            this->start();
            this->stop();
        }
```

6.67.2 Member Function Documentation

6.67.2.1 elapsed()

```
std::chrono::duration<std::chrono::system_clock::rep, std::chrono::system_clock::period> ats::date_time::stop_watch::elapsed ( ) [inline]
```

```
        {
            return end_t - start_t;
        }
```

6.67.2.2 microseconds()

```
long long ats::date_time::stop_watch::microseconds ( ) [inline]
```

```
        {
            auto dif = std::chrono::duration_cast<std::chrono::microseconds>(end_t - start_t);
            return dif.count();
        }
```

6.67.2.3 milliseconds()

```
long long ats::date_time::stop_watch::milliseconds ( ) [inline]
```

```
        {
            auto dif = std::chrono::duration_cast<std::chrono::milliseconds>(end_t - start_t);
            return dif.count();
        }
```

6.67.2.4 seconds()

```
long long ats::date_time::stop_watch::seconds ( ) [inline]
```

```
        {
            auto dif = std::chrono::duration_cast<std::chrono::seconds>(end_t - start_t);
            return dif.count();
        }
```

6.67.2.5 start()

```
void ats::date_time::stop_watch::start ( )  [inline]
```

Referenced by stop_watch().

```
{ start_t = std::chrono::system_clock::now(); }
```

6.67.2.6 stop()

```
void ats::date_time::stop_watch::stop ( )  [inline]
```

Referenced by stop_watch().

```
{ end_t = std::chrono::system_clock::now(); }
```

The documentation for this class was generated from the following file:

- date_time/stop_watch.hpp

6.68 ats::stream_reader< MsgT > Class Template Reference

```
#include <stream_reader.hpp>
```

Collaboration diagram for ats::stream_reader< MsgT >:

Public Member Functions

- virtual ~stream_reader ()
- virtual bool read (std::queue< MsgT > &messages)=0

6.68.1 Constructor & Destructor Documentation

6.68.1.1 ~stream_reader()

```
template<typename MsgT >
virtual ats::stream_reader< MsgT >::~stream_reader ( )  [inline], [virtual]
```

References ats::stream_reader< MsgT >::read().

```
{ }
```

6.68.2 Member Function Documentation

6.68.2.1 read()

```
template<typename MsgT >
virtual bool ats::stream_reader< MsgT >::read (
             std::queue< MsgT > & messages )  [pure virtual]
```

Referenced by ats::stream_reader< MsgT >::∼stream_reader().

The documentation for this class was generated from the following file:

- io/stream_reader.hpp

6.69 ats::symbol_key Struct Reference

```
#include <types.hpp>
```

Collaboration diagram for ats::symbol_key:

Public Member Functions

- symbol_key (const std::string &symbol, size_t index)
- const std::string & to_string () const

Public Attributes

- std::string name
- size_t index

Friends

- std::ostream & operator<< (std::ostream &os, const ats::symbol_key &key)

6.69.1 Constructor & Destructor Documentation

6.69.1.1 symbol_key()

```
ats::symbol_key::symbol_key (
             const std::string & symbol,
             size_t index )  [inline]
```

```
             : name(symbol), index(index) { }
```

6.69.2 Member Function Documentation

6.69.2.1 to_string()

```
const std::string& ats::symbol_key::to_string ( ) const  [inline]
```

References name.

Referenced by ats::level2_execution_engine::subscribe().

```
{ return name; }
```

6.69.3 Friends And Related Function Documentation

6.69.3.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const ats::symbol_key & key )  [friend]
```

```
        {
            os << key.name;
            return os;
        }
```

6.69.4 Member Data Documentation

6.69.4.1 index

```
size_t ats::symbol_key::index
```

Referenced by ats::order_book_manager::create_order_book(), ats::portfolio_base::create_order_book(), ats::portfolio_base::get_casted_security(), ats::order_processor::get_order_book(), ats::order_book_manager::get_order_book(), ats::portfolio_base::get_order_book(), ats::order_processor::get_position(), ats::portfolio_base::get_position(), ats::portfolio_base::get_security(), ats::portfolio_base::process_execution(), ats::portfolio_base::process_message(), ats::portfolio_base::process_order_status_message(), ats::portfolio_base::send_order(), and ats::order_book_manager::update_order_book().

6.69.4.2 name

```
std::string ats::symbol_key::name
```

Referenced by ats::portfolio_base::add_security(), ats::portfolio_base::send_order(), and to_string().

The documentation for this struct was generated from the following file:

- types.hpp

6.70 ats::trade_message Struct Reference

```
#include <trade_message.hpp>
```

Inheritance diagram for ats::trade_message:

Collaboration diagram for ats::trade_message:

Public Attributes

- price_t price
- long quantity
- size_t seq_number = 0
- ats::market_state state
- ats::aggressor_side aggressor_side = ats::aggressor_side::Undefined

Additional Inherited Members

6.70.1 Member Data Documentation

6.70.1.1 aggressor_side

```
ats::aggressor_side ats::trade_message::aggressor_side = ats::aggressor_side::Undefined
```

6.70.1.2 price

```
price_t ats::trade_message::price
```

6.70.1.3 quantity

```
long ats::trade_message::quantity
```

6.70.1.4 seq_number

```
size_t ats::trade_message::seq_number = 0
```

6.70.1.5 state

```
ats::market_state ats::trade_message::state
```

The documentation for this struct was generated from the following file:

- message/trade_message.hpp

Chapter 7

File Documentation

7.1 container/double_key_lookup.hpp File Reference

```
#include <unordered_map>
```

Include dependency graph for double_key_lookup.hpp:

7.2 container/limit_order_container.hpp File Reference

```
#include <unordered_map>
#include <unordered_set>
#include <memory>
#include <ats/order/limit_order.hpp>
#include <ats/types.hpp>
```

Include dependency graph for limit_order_container.hpp:

Classes

- class ats::limit_order_container
- class ats::limit_order_container1

Namespaces

- ats

7.3 container/order_container.hpp File Reference

```
#include <memory>
#include <string>
#include <unordered_map>
#include <ats/order/order.hpp>
#include <ats/types.hpp>
```

Include dependency graph for order_container.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::order_container

Namespaces

- ats

7.4 container/order_container1.hpp File Reference

```
#include <memory>
#include <unordered_map>
#include <typeindex>
#include <ats/order/order.hpp>
#include <ats/types.hpp>
```

Include dependency graph for order_container1.hpp:

Classes

- class ats::order_container1

Namespaces

- ats

7.5 container/security_container.hpp File Reference

```
#include <ats/security/security_base.hpp>
#include <unordered_map>
#include <string>
#include <memory>
#include <utility>
```

Include dependency graph for security_container.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::security_container

Namespaces

- ats

7.6 custom_data_feeds/level2_historical_data_feed.hpp File Reference

```
#include <ats/data_feed/historical/historical_data_feed.hpp>
#include <ats/portfolio/portfolio_base.hpp>
```

Include dependency graph for level2_historical_data_feed.hpp:

Classes

- class ats::level2_historical_data_feed

Namespaces

- ats

7.7 custom_message_readers/level2_exchange_message_reader.hpp File Reference

```
#include <string>
#include <array>
#include <fstream>
#include <ats/data_feed/historical/exchange_message_reader_base.hpp>
#include <ats/message/level2_message.hpp>
#include <ats/io/csv_reader.hpp>
```

Include dependency graph for level2_exchange_message_reader.hpp:

Classes

- class ats::level2_exchange_message_reader

Namespaces

- ats

7.8 custom_message_readers/level2_message_reader.hpp File Reference

```
#include <ats/data_feed/historical/exchange_message_reader_base.hpp>
#include <ats/message/level2_message.hpp>
#include <ats/io/csv_reader.hpp>
#include <ats/io/tokenize.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include "boost/date_time/gregorian/gregorian.hpp"
```

Include dependency graph for level2_message_reader.hpp:

Classes

- class ats::l2_message_reader

Namespaces

- ats

7.9 data_feed/data_feed.hpp File Reference

```
#include <ats/event_handler/multievent_handler.hpp>
#include <ats/portfolio/portfolio_base.hpp>
```

Include dependency graph for data_feed.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::data_feed

Namespaces

- ats

7.10 data_feed/historical/csv_message_reader.hpp File Reference

```
#include <ats/io/csv_reader.hpp>
#include "message_reader.hpp"
```

Include dependency graph for csv_message_reader.hpp:

Classes

- class ats::csv_single_message_reader< MessageT, n_columns >

Namespaces

- ats

7.11 data_feed/historical/exchange_message_reader_base.hpp File Reference

```
#include <ats/portfolio/portfolio_base.hpp>
#include "message_reader.hpp"
```

Include dependency graph for exchange_message_reader_base.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::exchange_message_reader_base< MessageT >

Namespaces

- ats

7.12 data_feed/historical/historical_data_feed.hpp File Reference

```
#include <memory>
#include <vector>
#include <map>
#include <utility>
#include "message_reader.hpp"
#include <ats/data_feed/data_feed.hpp>
#include <ats/message/message.hpp>
#include <ats/portfolio/portfolio_base.hpp>
```

Include dependency graph for historical_data_feed.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::historical_data_feed

Namespaces

- ats

7.13 data_feed/historical/message_reader.hpp File Reference

```
#include <ats/event_handler/multievent_handler.hpp>
#include <ats/message/message.hpp>
```

Include dependency graph for message_reader.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::message_reader
- class ats::single_message_reader< MessageT >

Namespaces

- ats

7.14 date_time/date_time.hpp File Reference

```
#include <string>
#include <cstdlib>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
```

Include dependency graph for date_time.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::date_time::date_time

Namespaces

- ats
- ats::date_time

7.15 date_time/stop_watch.hpp File Reference

```
#include <chrono>
```

Include dependency graph for stop_watch.hpp:

Classes

- class ats::date_time::stop_watch

Namespaces

- ats
- ats::date_time

7.16 event_handler/event_handler.hpp File Reference

```
#include <functional>
#include <list>
#include <utility>
```

Include dependency graph for event_handler.hpp:

Classes

- class ats::event_handler< T >
- class ats::event_handler< ReturnT(Args...)>

Namespaces

- ats

7.17 event_handler/multievent_handler.hpp File Reference

```
#include <unordered_map>
#include <list>
#include <typeindex>
#include <memory>
#include <functional>
```

Include dependency graph for multievent_handler.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ats::detail::function_base
- struct ats::detail::basic_function< T >
- class ats::multievent_handler

Namespaces

- ats
- ats::detail

7.18 event_handler/multievent_memfunc_handler.hpp File Reference

```
#include <typeindex>
#include <unordered_map>
#include <memory>
#include <ats/message/message.hpp>
```

Include dependency graph for multievent_memfunc_handler.hpp:

Classes

- class ats::detail::member_function_handler_base
- class ats::detail::member_function_handler< Object, EventArgsT >
- class ats::multievent_memfunc_handler

Namespaces

- ats
- ats::detail

Typedefs

- using basic_message = ats::instrument_message

7.18.1 Typedef Documentation

7.18.1.1 basic_message

```
using basic_message = ats::instrument_message
```

7.19 execution_engine/execution_engine.hpp File Reference

```
#include <string>
#include <ats/event_handler/multievent_handler.hpp>
#include <ats/order/market_order.hpp>
#include <ats/order/limit_order.hpp>
#include <ats/order/stop_order.hpp>
#include <ats/handler_types.hpp>
#include <ats/types.hpp>
```

Include dependency graph for execution_engine.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::execution_engine

Namespaces

- ats

7.20 execution_engine/level2/level2_execution_engine.cpp File Reference

```
#include "level2_execution_engine.hpp"
```

Include dependency graph for level2_execution_engine.cpp:

Namespaces

- ats

7.21 execution_engine/level2/level2_execution_engine.hpp File Reference

```
#include <string>
#include <map>
#include <unordered_map>
#include <memory>
#include <utility>
#include <typeindex>
#include <ats/execution_engine/execution_engine.hpp>
#include <ats/order_book/exchange_order_book.hpp>
#include <ats/order_book/simulation/fifo_exchange_order_book.hpp>
#include <ats/order_book/simulation/sim_book.hpp>
#include <ats/handler_types.hpp>
#include <ats/types.hpp>
```

Include dependency graph for level2_execution_engine.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::level2_execution_engine

Namespaces

- ats

7.22 handler_types.hpp File Reference

```
#include <functional>
#include <ats/message/level2_message.hpp>
#include <ats/message/trade_message.hpp>
#include <ats/message/order_status_message.hpp>
#include <ats/position/position.hpp>
```

Include dependency graph for handler_types.hpp: This graph shows which files directly or indirectly include this file:

Namespaces

- ats

Typedefs

- typedef std::function< void(const ats::order_status_message &)> ats::order_status_handler
- typedef std::function< void(const ats::position &)> ats::position_change_handler
- typedef std::function< void(const ats::level2_message_packet &)> ats::order_book_changed_handler

7.23 order_processor/handler_types.hpp File Reference

```
#include <functional>
#include <ats/message/order_status_message.hpp>
```

Include dependency graph for handler_types.hpp:

Namespaces

- ats

7.24 portfolio/handler_types.hpp File Reference

```
#include <functional>
#include <ats/message/level2_message.hpp>
#include <ats/message/trade_message.hpp>
#include <ats/message/order_status_message.hpp>
```

Include dependency graph for handler_types.hpp:

Namespaces

- ats

Typedefs

- typedef std::function< void(const ats::order_status_message &)> ats::on_order_status_changed_handler
- typedef std::function< void(const ats::level2_message_packet &)> ats::on_order_book_changed_handler
- typedef std::function< void(const ats::trade_message &)> ats::on_trade_handler

7.25 indicator/indicator.hpp File Reference

```
#include <ats/message/level2_message.hpp>
```

Include dependency graph for indicator.hpp:

Classes

- class ats::indicator< ValueT >

Namespaces

- ats

7.26 io/csv_reader.hpp File Reference

```
#include <string>
#include <array>
#include <fstream>
```

Include dependency graph for csv_reader.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::csv_reader< num_columns >

Namespaces

- ats

7.27 io/parser/fix_parser.hpp File Reference

```
#include <unordered_set>
#include <string>
#include <ats/message/level2_message.hpp>
#include <ats/date_time/date_time.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <quickfix/DataDictionary.h>
#include <quickfix/fix50sp2/MarketDataIncrementalRefresh.h>
```

Include dependency graph for fix_parser.hpp: This graph shows which files directly or indirectly include this file:

Namespaces

- ats

Functions

- static bool ats::parse_fix_msg (const char ∗fix_msg, ats::level2_message_packet &result, const FIX::Data↩ Dictionary &dictionary, const std::unordered_set< std::string > &symbols)

7.28 io/stream_reader.hpp File Reference

```
#include <istream>
#include <vector>
#include <queue>
```

Include dependency graph for stream_reader.hpp:

Classes

- class ats::stream_reader< MsgT >

Namespaces

- ats

7.29 io/tokenize.hpp File Reference

```
#include <string>
#include <array>
```

Include dependency graph for tokenize.hpp: This graph shows which files directly or indirectly include this file:

Namespaces

- ats

Functions

- template<size_t num_cols>
  static size_t ats::tokenize (const std::string &line, std::array< std::string, num_cols > &result, char sep=',')

7.30 io/transform/to_spread.hpp File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <ats/date_time/date_time.hpp>
```

Include dependency graph for to_spread.hpp:

Namespaces

- ats

Functions

- static void ats::to_spread (const std::string &x_file, const std::string &y_file, const std::string &out_file, const std::string &header="Symbol; Timestamp; BidPrc; AskPrc")

7.31 io/writer/fix_to_csv.hpp File Reference

```
#include <fstream>
#include <string>
#include <unordered_set>
#include <ats/io/parser/fix_parser.hpp>
#include <ats/message/level2_message.hpp>
```

Include dependency graph for fix_to_csv.hpp:

Namespaces

- ats

Functions

- static void ats::fix_to_csv (const std::string &fix_file, const std::string &csv_file, const std::string &fix_specs↩ _xml, const std::unordered_set< std::string > &symbols, bool print_seq_num=false, const std::string &eop↩ _str="EOP")

7.32 message/level2_message.hpp File Reference

```
#include <string>
#include <vector>
#include "message.hpp"
#include "message_defs.hpp"
```

Include dependency graph for level2_message.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ats::level2_message

Namespaces

- ats

Typedefs

- typedef ats::instrument_message_packet< level2_message > ats::level2_message_packet

7.33 message/message.hpp File Reference

```
#include <string>
#include <vector>
#include <utility>
#include <ats/types.hpp>
```

Include dependency graph for message.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ats::message
- struct ats::instrument_message
- struct ats::instrument_message_packet< MessageT >

Namespaces

- ats

7.34 message/message_defs.hpp File Reference

```
#include <string>
#include <vector>
#include "message.hpp"
```

Include dependency graph for message_defs.hpp: This graph shows which files directly or indirectly include this file:

Namespaces

- ats

Enumerations

- enum ats::trading_session_id { ats::trading_session_id::PreOpening = 0, ats::trading_session_id::Opening, ats::trading_session_id::Continuous }
- enum ats::update_action { ats::update_action::New = 0, ats::update_action::Change, ats::update_action::↩ Delete, ats::update_action::Overlay }
- enum ats::entry_type { ats::entry_type::Bid = 0, ats::entry_type::Ask, ats::entry_type::Trade }
- enum ats::aggressor_side { ats::aggressor_side::Undefined = 0, ats::aggressor_side::Buy, ats::aggressor_↩ side::Sell }
- enum ats::market_state { ats::market_state::PreOpening, ats::market_state::Opening, ats::market_state::↩ ContinuousTrading }

7.35 message/order_status_message.hpp File Reference

```
#include <string>
#include <memory>
#include <ats/types.hpp>
#include <ats/order/order_defs.hpp>
#include <ats/message/message.hpp>
#include <ats/message/order_status_message.hpp>
```

Include dependency graph for order_status_message.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ats::order_status_message
- struct ats::order_status_filled_message
- struct ats::order_status_partially_filled_message
- struct ats::order_status_cancelled_message
- struct ats::order_status_rejected_message
- struct ats::order_status_pending_new_message
- struct ats::order_status_new_message

Namespaces

- ats

7.36 message/trade_message.hpp File Reference

```
#include <string>
#include <vector>
#include "message_defs.hpp"
```

Include dependency graph for trade_message.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ats::trade_message

Namespaces

- ats

Typedefs

- typedef ats::instrument_message_packet< ats::trade_message > ats::trade_message_packet

7.37 order/bracket_order.hpp File Reference

```
#include "limit_order.hpp"
#include "stop_order.hpp"
```

Include dependency graph for bracket_order.hpp:

Classes

- class ats::bracket_order

Namespaces

- ats

7.38 order/limit_order.hpp File Reference

```
#include "order.hpp"
#include <ats/types.hpp>
```

Include dependency graph for limit_order.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::limit_order

Namespaces

- ats

7.39 order/market_order.hpp File Reference

```
#include <memory>
#include "order.hpp"
#include <ats/types.hpp>
```

Include dependency graph for market_order.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::market_order

Namespaces

- ats

7.40 order/order.hpp File Reference

```
#include <string>
#include <memory>
#include <typeindex>
#include "order_defs.hpp"
#include "order_type.hpp"
#include <ats/types.hpp>
```

Include dependency graph for order.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::order

Namespaces

- ats

7.41 order/order_defs.hpp File Reference

This graph shows which files directly or indirectly include this file:

Namespaces

- ats

Enumerations

- enum ats::order_side { ats::order_side::Buy, ats::order_side::Sell, ats::order_side::BuyCover, ats::order_side::SellShort }
- enum ats::order_time_in_force {
  ats::order_time_in_force::Day, ats::order_time_in_force::GTC, ats::order_time_in_force::IOC, ats::order_time_in_force::AtTheOpening,
  ats::order_time_in_force::AtTheClose, ats::order_time_in_force::FUNARI, ats::order_time_in_force::FOK, ats::order_time_in_force::GTX,
  ats::order_time_in_force::Date }
- enum ats::order_status {
  ats::order_status::New = 0, ats::order_status::PartiallyFilled, ats::order_status::Filled, ats::order_status::DoneForDay,
  ats::order_status::Canceled, ats::order_status::Replaced, ats::order_status::PendingCancel, ats::order_status::Rejected,
  ats::order_status::Suspended, ats::order_status::PendingNew, ats::order_status::Calculated, ats::order_status::Expired,
  ats::order_status::AcceptedForBidding, ats::order_status::PendingReplace }

7.42 order/order_type.hpp File Reference

This graph shows which files directly or indirectly include this file:

Namespaces

- ats

Enumerations

- enum ats::order_type { ats::order_type::Market = 1, ats::order_type::Limit, ats::order_type::Stop, ats::order_type::StopLimit }

7.43 order/stop_order.hpp File Reference

```
#include "order.hpp"
#include <ats/types.hpp>
```

Include dependency graph for stop_order.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::stop_order

Namespaces

- ats

7.44 order_book/detail/price_level.hpp File Reference

```
#include <ats/types.hpp>
```

Include dependency graph for price_level.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ats::order_book_detail::price_level

  price level in an order book

Namespaces

- ats
- ats::order_book_detail

7.45 order_book/detail/price_levels.hpp File Reference

```
#include <map>
#include <stdexcept>
#include <ats/message/level2_message.hpp>
#include <ats/order_book/detail/price_level.hpp>
#include <ats/types.hpp>
```

Include dependency graph for price_levels.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::order_book_detail::price_levels< compare >

Namespaces

- ats
- ats::order_book_detail

7.46 order_book/exchange_order_book.hpp File Reference

```
#include <iosfwd>
#include <fstream>
#include <ats/order_book/detail/price_levels.hpp>
#include <ats/message/level2_message.hpp>
#include <ats/types.hpp>
```

Include dependency graph for exchange_order_book.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::exchange_order_book

Namespaces

- ats

7.47 order_book/order_book.hpp File Reference

```
#include <unordered_map>
#include "exchange_order_book.hpp"
#include <ats/message/level2_message.hpp>
#include <ats/types.hpp>
```

Include dependency graph for order_book.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::order_book

Namespaces

- ats

7.48 order_book/order_book_manager.hpp File Reference

```
#include <vector>
#include "order_book.hpp"
#include <ats/message/level2_message.hpp>
#include <ats/types.hpp>
```

Include dependency graph for order_book_manager.hpp:

Classes

- class ats::order_book_manager

Namespaces

- ats

7.49 order_book/simulation/fifo_exchange_order_book.hpp File Reference

```
#include <string>
#include "sim_book.hpp"
#include <ats/order_book/exchange_order_book.hpp>
```

Include dependency graph for fifo_exchange_order_book.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::sim::fifo_exchange_order_book

Namespaces

- ats
- ats::sim

7.50 order_book/simulation/sim_book.hpp File Reference

```
#include <unordered_map>
#include "sim_book_price_levels.hpp"
#include <ats/message/order_status_message.hpp>
```

Include dependency graph for sim_book.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::sim::sim_book

Namespaces

- ats
- ats::sim

7.51 order_book/simulation/sim_book_price_level.hpp File Reference

```
#include <list>
#include <unordered_map>
#include <string>
#include <sstream>
#include <ats/order/limit_order.hpp>
#include <ats/message/level2_message.hpp>
#include <ats/handler_types.hpp>
```

Include dependency graph for sim_book_price_level.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::sim::price_level

Namespaces

- ats
- ats::sim

7.52 order_book/simulation/sim_book_price_levels.hpp File Reference

```
#include <map>
#include "sim_book_price_level.hpp"
```

Include dependency graph for sim_book_price_levels.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::sim::price_levels< comp >

Namespaces

- ats
- ats::sim

7.53 order_processor/execution.hpp File Reference

```
#include <string>
#include <ats/types.hpp>
```

Include dependency graph for execution.hpp:

Classes

- struct ats::execution

Namespaces

- ats

7.54 portfolio/order_processor.hpp File Reference

```
#include <vector>
#include <ats/order_book/order_book.hpp>
#include <ats/position/position.hpp>
#include <ats/types.hpp>
```

Include dependency graph for order_processor.hpp:

Classes

- class ats::order_processor

Namespaces

- ats

7.55 portfolio/portfolio_base.cpp File Reference

```
#include "portfolio_base.hpp"
```

Include dependency graph for portfolio_base.cpp:

Namespaces

- ats

7.56 portfolio/portfolio_base.hpp File Reference

```
#include <unordered_map>
#include <string>
#include <vector>
#include <utility>
#include <memory>
#include <ats/order/order.hpp>
#include <ats/container/security_container.hpp>
#include <ats/container/order_container.hpp>
#include <ats/security/security_base.hpp>
#include <ats/position/position.hpp>
#include <ats/execution_engine/level2/level2_execution_engine.hpp>
#include <ats/report/report_engine.hpp>
```

Include dependency graph for portfolio_base.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::portfolio_base

Namespaces

- ats

Enumerations

- enum ats::execution_mode { ats::execution_mode::BackTesting, ats::execution_mode::WarmUp, ats↩ ::execution_mode::RealTime }

7.57 position/position.hpp File Reference

```
#include <cstdint>
#include <stack>
#include <memory>
#include <ats/types.hpp>
#include <ats/message/order_status_message.hpp>
```

Include dependency graph for position.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ats::position_offset_message
- class ats::position

Namespaces

- ats

7.58 recursive_timer.hpp File Reference

```
#include <functional>
#include <list>
```

Include dependency graph for recursive_timer.hpp:

Classes

- class ats::recursive_timer

Namespaces

- ats

7.59 report/report_engine.hpp File Reference

```
#include <vector>
#include <ats/types.hpp>
```

Include dependency graph for report_engine.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ats::pnl_item
- struct ats::performance_item_basic
- struct ats::performance_item
- class ats::report_engine

Namespaces

- ats

7.60 security/bar.hpp File Reference

This graph shows which files directly or indirectly include this file:

Classes

- struct ats::bar

Namespaces

- ats

7.61 security/metainfo.hpp File Reference

```
#include <ats/types.hpp>
```

Include dependency graph for metainfo.hpp:

Classes

- struct ats::metainfo

Namespaces

- ats

7.62 security/security_base.cpp File Reference

```
#include "security_base.hpp"
#include <ats/portfolio/portfolio_base.hpp>
```

Include dependency graph for security_base.cpp:

Namespaces

- ats

7.63 security/security_base.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <map>
#include <functional>
#include <memory>
#include <list>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/posix_time/posix_time_io.hpp>
#include <boost/circular_buffer.hpp>
#include <ats/position/position.hpp>
#include <ats/order_book/order_book.hpp>
#include <ats/message/level2_message.hpp>
#include <ats/message/trade_message.hpp>
#include <ats/message/order_status_message.hpp>
#include "bar.hpp"
```

Include dependency graph for security_base.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class ats::security_base

Namespaces

- ats

7.64 types.hpp File Reference

```
#include <chrono>
#include <cstdint>
#include <string>
#include <ostream>
#include <ats/date_time/date_time.hpp>
```

Include dependency graph for types.hpp: This graph shows which files directly or indirectly include this file:

Classes

- struct ats::symbol_key

Namespaces

- ats

Typedefs

- using ats::price_t = int
- using ats::timestamp_t = ats::date_time::date_time
- using ats::orderid_t = uint64_t

Enumerations

- enum ats::subscription {
  ats::subscription::Level2, ats::subscription::Level1, ats::subscription::TimeAndSales, ats::subscription::Bar,
  ats::subscription::Daily, ats::subscription::Custom }