

# My Project

Generated by Doxygen 1.8.1.2

Tue Jan 28 2014 16:33:13



# Contents

<b>1</b>	<b>muTrade API documentation</b>	<b>1</b>
1.1	Introduction . . . . .	1
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>Namespace Documentation</b>	<b>7</b>
4.1	mutrade::detail Namespace Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	mutrade::AbstractLogger Class Reference . . . . .	9
5.2	mutrade::Application Class Reference . . . . .	9
5.2.1	Detailed Description . . . . .	9
5.2.2	Member Function Documentation . . . . .	10
5.2.2.1	onExecutionReport . . . . .	10
5.2.2.2	onLoadInstrumentEnd . . . . .	10
5.2.2.3	onLogin . . . . .	10
5.2.2.4	onLogout . . . . .	10
5.2.2.5	onTick . . . . .	10
5.3	mutrade::Context Class Reference . . . . .	10
5.3.1	Detailed Description . . . . .	11
5.3.2	Member Function Documentation . . . . .	11
5.3.2.1	enableLogging . . . . .	11
5.3.2.2	getInstance . . . . .	12
5.3.2.3	getInstrument . . . . .	12
5.3.2.4	loadInstrument . . . . .	12
5.3.2.5	login . . . . .	12
5.3.2.6	logout . . . . .	12
5.3.2.7	placeOrder . . . . .	12

5.3.2.8	setApplication	13
5.3.2.9	subscribe	13
5.3.2.10	unsubscribe	13
5.4	mutrade::ExecutionReport Class Reference	13
5.4.1	Detailed Description	15
5.4.2	Member Function Documentation	15
5.4.2.1	getClientId	15
5.4.2.2	getClOrderId	15
5.4.2.3	getErrorCode	15
5.4.2.4	getLastFillPrice	15
5.4.2.5	getLastFillQuantity	16
5.4.2.6	getOrderMode	16
5.4.2.7	getOrderQuantity	16
5.4.2.8	getOriginalClOrderId	16
5.4.2.9	getSymbolId	16
5.4.2.10	getTradeId	16
5.5	mutrade::ExecutionResponse Class Reference	17
5.5.1	Detailed Description	17
5.6	mutrade::Instrument Class Reference	17
5.6.1	Member Function Documentation	18
5.6.1.1	getStrikePrice	18
5.7	mutrade::Logger Class Reference	18
5.7.1	Detailed Description	18
5.7.2	Member Function Documentation	18
5.7.2.1	getInstance	19
5.7.2.2	setLogLevel	19
5.8	mutrade::MarketData Class Reference	19
5.8.1	Detailed Description	20
5.8.2	Member Function Documentation	20
5.8.2.1	getAvgPrice	20
5.8.2.2	getAvgPriceForQty	20
5.8.2.3	getCount	20
5.8.2.4	getDayClose	20
5.8.2.5	getDayHigh	20
5.8.2.6	getDayLow	21
5.8.2.7	getDayOpen	21
5.8.2.8	getDepth	21
5.8.2.9	getLastPrice	21
5.8.2.10	getLastQty	21
5.8.2.11	getLastTime	21

5.8.2.12	<a href="#">getPrice</a>	21
5.8.2.13	<a href="#">getQty</a>	21
5.8.2.14	<a href="#">getQtyForAvgPrice</a>	21
5.8.2.15	<a href="#">getQtyForWorstPrice</a>	22
5.8.2.16	<a href="#">getRank</a>	22
5.8.2.17	<a href="#">getTotalQty</a>	22
5.8.2.18	<a href="#">getWorstPriceForQty</a>	22
5.8.2.19	<a href="#">hasQty</a>	22
5.9	<a href="#">mutrade::MarketDataSubscription Class Reference</a>	22
5.10	<a href="#">mutrade::NetPositions Class Reference</a>	22
5.10.1	<a href="#">Detailed Description</a>	23
5.10.2	<a href="#">Member Function Documentation</a>	23
5.10.2.1	<a href="#">getPosition</a>	23
5.10.2.2	<a href="#">update</a>	23
5.11	<a href="#">mutrade::Order Class Reference</a>	23
5.11.1	<a href="#">Detailed Description</a>	25
5.11.2	<a href="#">Member Function Documentation</a>	25
5.11.2.1	<a href="#">getClOrdId</a>	25
5.11.2.2	<a href="#">getExchangeOrderId</a>	25
5.11.2.3	<a href="#">getFilledQuantity</a>	25
5.11.2.4	<a href="#">getOrderMode</a>	25
5.11.2.5	<a href="#">getOrderStatus</a>	25
5.11.2.6	<a href="#">getOrderType</a>	26
5.11.2.7	<a href="#">getOrderValidity</a>	26
5.11.2.8	<a href="#">getPrice</a>	26
5.11.2.9	<a href="#">getQuantity</a>	26
5.11.2.10	<a href="#">getSecurityType</a>	26
5.11.2.11	<a href="#">getStopPrice</a>	26
5.11.2.12	<a href="#">getSymbol</a>	26
5.11.2.13	<a href="#">getTransactionType</a>	27
5.11.2.14	<a href="#">setClOrdId</a>	27
5.11.2.15	<a href="#">setOrderMode</a>	27
5.11.2.16	<a href="#">setOrderStatus</a>	27
5.11.2.17	<a href="#">setOrderType</a>	27
5.11.2.18	<a href="#">setPrice</a>	27
5.11.2.19	<a href="#">setQuantity</a>	27
5.11.2.20	<a href="#">setSymbol</a>	28
5.11.2.21	<a href="#">setTransactionType</a>	28
5.12	<a href="#">mutrade::OrderBook Class Reference</a>	28
5.12.1	<a href="#">Detailed Description</a>	28

5.12.2	Member Function Documentation	28
5.12.2.1	getOrder	29
5.12.2.2	insert	29
5.12.2.3	update	29
5.13	mutrade::Portfolio Class Reference	29
5.13.1	Detailed Description	30
5.13.2	Member Function Documentation	30
5.13.2.1	getInstance	30
5.13.2.2	getOrderByTokenId	30
5.13.2.3	handleConfirmations	30
5.13.2.4	handleResponse	30
5.13.2.5	insert	30
5.14	mutrade::Position Class Reference	31
5.14.1	Detailed Description	31
5.14.2	Constructor & Destructor Documentation	32
5.14.2.1	Position	32
5.14.2.2	Position	32
5.15	mutrade::Quote Class Reference	32
5.16	mutrade::Trade Class Reference	33
5.17	mutrade::TradeBook Class Reference	34
5.17.1	Detailed Description	34
5.17.2	Member Function Documentation	34
5.17.2.1	getTrades	34
5.17.2.2	update	34

# Chapter 1

## muTrade API documentation

### 1.1 Introduction

This is an early release of the muTrade API, which exposes the core trading functionalities and allows the developer to write an event driven trading algorithm.

#### Note

This version of API is still experimental and the functionality/interface may break in the future versions of API.

#### Code Flow

- 1) [Application](#) developer has to override the virtual methods of [Application](#) class.
- 2) Register your overridden [Application](#) class to API using [setApplication](#) function.
- 3) Call [login](#) function. Once user is logged in, application developer has to control its flow from the overridden [Application](#) class.
- 4) In [onLogin](#) user has to call [loadInstrument](#). [User must load the instrument before using it.]
- 5) In [onLoadInstrumentEnd](#) user should call [subscribe](#) function in order to get live ticks/quotes from the server.
- 6) For every subscribed symbol user will get an event [onTick](#).
- 7) Based on the ticks user can place their order using [placeOrder](#).
- 8) For every placed order user will get an event [onExecutionReport](#).

[OrderBook](#), [TradeBook](#) and [NetPositions](#) can be accessed from the [Portfolio](#) class.





## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">mutrade::detail</a>	
<a href="#">Instrument</a> class	7



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">mutrade::AbstractLogger</a>	9
<a href="#">mutrade::Application</a>	
Abstract <a href="#">Application</a> class, to be overridden by the developer	9
<a href="#">mutrade::Context</a>	
Context class for the algorithm	10
<a href="#">mutrade::ExecutionReport</a>	
Execution Report Class	13
<a href="#">mutrade::ExecutionResponse</a>	
Execution Response	17
<a href="#">mutrade::Instrument</a>	17
<a href="#">mutrade::Logger</a>	
Abstract <a href="#">Logger</a> class	18
<a href="#">mutrade::MarketData</a>	
MarketData Class	19
<a href="#">mutrade::MarketDataSubscription</a>	22
<a href="#">mutrade::NetPositions</a>	
NetPositions class	22
<a href="#">mutrade::Order</a>	
Order Class	23
<a href="#">mutrade::OrderBook</a>	
OrderBook class	28
<a href="#">mutrade::Portfolio</a>	
Portfolio class	29
<a href="#">mutrade::Position</a>	
Position class	31
<a href="#">mutrade::Quote</a>	32
<a href="#">mutrade::Trade</a>	33
<a href="#">mutrade::TradeBook</a>	
TradeBook class	34



## Chapter 4

# Namespace Documentation

### 4.1 mutrade::detail Namespace Reference

[Instrument](#) class.

#### 4.1.1 Detailed Description

[Instrument](#) class. [Instrument](#) can be generated from a string identifier which uniquely identifies a particular string. This string takes the following format:

**Equities** "<exchange> <symbol> <series, if any>"

"<exchange> <security-id>"

**Futures** "<exchange> <symbol> <maturity-date>"

"<exchange> <symbol> <year-month, if single contract>"

**Options** "<exchange> <symbol> <maturity-date> <strike> <call/put>"

For example:

- "NSE SBIN EQ" equity with symbol and series
- "BSE 500112" equity with BSE scrip-code (SBI)
- "BSE SBI A" equity with symbol and series
- "NSE SBIN 20130926" future listed on NSE with expiry date in format YYYY-MM-DD
- "BSE SBI 26SEP2013" future on BSE with expiry date in format DDMONYYYYY
- "NSE SBIN 26SEP2013 145000 C" SBIN Call option on NSE with strike price of 1450.00
- "NSE SBIN 20130926 145000 P" SBIN Put option on NSE with strike price of 1450.00



## Chapter 5

# Class Documentation

### 5.1 mutrade::AbstractLogger Class Reference

#### Public Member Functions

- virtual void **log** (LogLevel level, const std::string &message)=0
- virtual int **getLevel** (LogLevel level)=0
- void **setLogLevel** (LogLevel level)
- int **getLogLevel** ()

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

- logger.h

### 5.2 mutrade::Application Class Reference

Abstract [Application](#) class, to be overridden by the developer.

```
#include <application.h>
```

#### Public Member Functions

- virtual void **onTick** (const [MarketData](#) &)=0  
*Event called when a tick is received.*
- virtual void **onLogin** (bool status)=0  
*Event called when Login message is returned.*
- virtual void **onLogout** (bool status)=0  
*Event called when Logout message is returned.*
- virtual void **onExecutionReport** ([ExecutionReport](#) &report)  
*Event called when an execution is received from Server.*
- virtual void **onLoadInstrumentEnd** (const String instrumentName, bool success)=0  
*Event called when instrument is loaded from the back-end.*

#### 5.2.1 Detailed Description

Abstract [Application](#) class, to be overridden by the developer.

[Application](#) is the base abstract class. An application developer, using muTrade API needs to inherit from this class and override the virtual methods of this class.

## 5.2.2 Member Function Documentation

5.2.2.1 `virtual void mutrade::Application::onExecutionReport ( ExecutionReport & report )` [virtual]

Event called when an execution is received from Server.

Parameters

--	--

5.2.2.2 `virtual void mutrade::Application::onLoadInstrumentEnd ( const String instrumentName, bool success )` [pure virtual]

Event called when instrument is loaded from the back-end.

Parameters

--	--

5.2.2.3 `virtual void mutrade::Application::onLogin ( bool status )` [pure virtual]

Event called when Login message is returned.

Parameters

--	--

5.2.2.4 `virtual void mutrade::Application::onLogout ( bool status )` [pure virtual]

Event called when Logout message is returned.

Parameters

--	--

5.2.2.5 `virtual void mutrade::Application::onTick ( const MarketData & )` [pure virtual]

Event called when a tick is received.

Parameters

--	--

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

- application.h

## 5.3 mutrade::Context Class Reference

[Context](#) class for the algorithm.

```
#include <context.h>
```



## Public Member Functions

- void [login](#) (Int32 userId, const String &password, String host, Int16 port, bool restoreState=false)  
*Login to muTrade server with with userId and password.*
- void [logout](#) ()  
*Logout from the muTrade server.*
- bool [placeOrder](#) (const [mutrade::Order](#) &order)  
*Send an order to the muTrade server.*
- void [enableLogging](#) (LogLevel level=INFO)  
*Enable logging of various events.*
- void [subscribe](#) (const [Instrument](#) &t)  
*Subscribe market data for a particular instrument.*
- void [unsubscribe](#) (const [Instrument](#) &t)  
*Unsubscribe market data for a previously subscribed instrument.*
- void [loadInstrument](#) (const String &s)  
*Load static data for an instrument from the muTrade server.*
- [Instrument](#) \* [getInstrument](#) (const String &t) const  
*Get static data for a particular instrument using symbol [loadInstrument](#) must be called for the string before calling this method.*
- detail::ContextImpl \* [getContextImpl](#) ()  
*Get the instance of ContextImpl class.*
- [Application](#) \* [getApplication](#) ()  
*Get the instance of [Application](#) class.*
- void [setApplication](#) ([Application](#) \*)  
*Set the instance of [Application](#) class. User need to resigster it's derived application class to context. User must call this function before login.*

## Static Public Member Functions

- static [Context](#) \* [getInstance](#) ()  
*Get an Instance of the [Context](#) class.*

### 5.3.1 Detailed Description

[Context](#) class for the algorithm.

This class contains the event engine for the applicaton and does the actual communication with the muTrade server. [Application](#) object containing the trading logic is associated with this class. Also, this class is used to tweak various parameters, which are global to the application.

### 5.3.2 Member Function Documentation

#### 5.3.2.1 void mutrade::Context::enableLogging ( LogLevel level = INFO )

Enable logging of various events.

#### Parameters

lc	logging level for how much log we want to generate
----	--

These logs also go to syslog on Linux/UNIX and to Event Log on Windows

### 5.3.2.2 static Context\* mutrade::Context::getInstance ( ) [inline],[static]

Get an Instance of the [Context](#) class.

[Context](#) class is a Singleton class, which will have only one instance. This instance can be accessed using the `getInstance` method.

### 5.3.2.3 Instrument\* mutrade::Context::getInstrument ( const String & t ) const

Get static data for a particular instrument using symbol [loadInstrument](#) must be called for the string before calling this method.

### 5.3.2.4 void mutrade::Context::loadInstrument ( const String & s )

Load static data for an instrument from the muTrade server.

### 5.3.2.5 void mutrade::Context::login ( Int32 *userId*, const String & *password*, String *host*, Int16 *port*, bool *restoreState* = false )

Login to muTrade server with with *userId* and *password*.

#### Parameters

<i>userId</i>	to login with
<i>password</i>	for user
<i>host</i>	ip
<i>port</i>	of host
<i>restore</i>	previous trade

### 5.3.2.6 void mutrade::Context::logout ( )

Logout from the muTrade server.

### 5.3.2.7 bool mutrade::Context::placeOrder ( const mutrade::Order & order )

Send an order to the muTrade server.

#### Parameters

<i>order</i>	<a href="#">Order</a>
--------------	-----------------------

Before placing the order user need to set the order with these informations.

For **New Order**

[TransactionType](#) to `TransactionType_NEW`.

Symbol Name with [Instrument](#) name.

[Order](#) Mode with [OrderMode](#).

[Order](#) Quantity. It must be multiple of lot size.

[Order](#) Price. Try to set it in multiple of tick size.

Order Validity to [TimeInForce](#).

Disclosed Quantity as order qty.

[Order](#) Type with [OrderType](#).

[Order](#) Status to `OrderStatus_PENDING`.

Security Type to [InstrumentType](#).

For **Modify Order**

[TransactionType](#) to `TransactionType_MODIFY`.

[Order](#) Quantity.

[Order](#) Price. Try to set it in multiple of tick size.

Order Validity to [TimeInForce](#).

[Order](#) Type with [OrderType](#).

[Order](#) Status to `OrderStatus_PENDING`.

setCLOrderId to CLOrId of previous order.

For **Cancel Order**

[TransactionType](#) to `TransactionType_CANCEL`.

[Order](#) Status to `OrderStatus_PENDING`.

Symbol Name with [Instrument](#) name.

setCLOrderId to CLOrId of previous order.

#### 5.3.2.8 void mutrade::Context::setApplication ( Application \* )

Set the instance of [Application](#) class. User need to resigster it's derived application class to context. User must call this function before login.

#### 5.3.2.9 void mutrade::Context::subscribe ( const Instrument & t )

Subsbscribe market data for a particular instrument.

[loadInstrument](#) must be called for the string before calling this method.

#### 5.3.2.10 void mutrade::Context::unsubscribe ( const Instrument & t )

Unsubsbscribe market data for a previously subscribed instrument.

[loadInstrument](#) must be called for the string before calling this method.

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

- context.h

## 5.4 mutrade::ExecutionReport Class Reference

Execution Report Class.

```
#include <types.h>
```

### Public Member Functions

- **ExecutionReport** (const char \*buf)
- **ExecutionReport** (RSP::OrderConfirmation &confirmation)

- void **initialize** ()
- Int64 **getClOrderId** () const  
*Get Client Order Id.*
- String **getExchangeOrderId** () const
- Int64 **getSymbolId** () const  
*Get Symbol Id.*
- Int32 **getLastFillQuantity** () const  
*Get Last Fill Quantity.*
- Int32 **getLastFillPrice** () const  
*Get Last Fill Price.*
- Int32 **getExchangeEntryTime** () const
- Int32 **getExchangeModifyTime** () const
- Int32 **getStrategyId** () const
- Int32 **getClientId** () const  
*Get Client Id.*
- Int32 **getLimitPrice** () const
- UChar **getOrderStatus** () const
- OrderMode **getOrderMode** () const  
*Get Order Mode.*
- Int32 **getOrderQuantity** () const  
*Get Orde Quantity.*
- Int32 **getOrderPrice** () const
- Int32 **getIOCCanceledQuantity** () const
- Int64 **getOriginalClOrderId** () const  
*Get Original Original Id.*
- Int64 **getConfirmationTimeSeconds** () const
- Int64 **getConfirmationTimeMicroSeconds** () const
- UChar **getIsOffline** () const
- Int64 **getSequenceNumber** () const
- String **getTradeId** () const  
*Get Trade Id.*
- Int32 **getErrorCode** () const  
*Get Error Code.*
- Int32 **getReasonText** () const
- UChar **getUnknownOrder** () const
- String **getInstrumentName** () const
- void **setClOrderId** (Int64 clOrderId)
- void **setExchangeOrderId** (String exchangeOrderId)
- void **setSymbolId** (Int64 symbolId)
- void **setLastFillQuantity** (Int32 qty)
- void **setLastFillPrice** (Int32 price)
- void **setExchangeEntryTime** (Int32 exchangeEntryTime)
- void **setExchangeModifyTime** (Int32 exchangeModifyTime)
- void **setStrategyId** (Int32 strategyId)
- void **setClientId** (Int32 clientId)
- void **setLimitPrice** (Int32 limitPrice)
- void **setOrderStatus** (UChar orderStatus)
- void **setOrderMode** (OrderMode orderMode)
- void **setOrderQuantity** (Int32 quantity)
- void **setOrderPrice** (Int32 price)
- void **setIOCCanceledQuantity** (Int32 quantity)
- void **setOriginalClOrderId** (Int64 originalClOrderId)
- void **setConfirmationTimeSeconds** (Int64 seconds)

- void **setConfirmationTimeMicroSeconds** (Int64 microSeconds)
- void **setIsOffline** (UChar isOffline)
- void **setSequenceNumber** (Int64 sequenceNumber)
- void **setTradeld** (String tradeld)
- void **setErrorCode** (Int32 errorCode)
- void **setReasonText** (Int32 reasonText)
- void **setUnknownOrder** (UChar unknownOrder)
- void **setInstrumentName** (String instrumentName)
- void **dump** ()
- void **dumpCSV** (std::ofstream &csvFile)
- int **serialize** (char \*buf)

### 5.4.1 Detailed Description

Execution Report Class.

User will get Execution Report as order confirmation from the exchange. For user it is read only class. Api will update the members of this class.

### 5.4.2 Member Function Documentation

#### 5.4.2.1 Int32 mutrade::ExecutionReport::getClientId ( ) const [inline]

Get Client Id.

Returns

User Id.

#### 5.4.2.2 Int64 mutrade::ExecutionReport::getClOrderId ( ) const [inline]

Get Client [Order](#) Id.

Returns

Client [Order](#) Id.

#### 5.4.2.3 Int32 mutrade::ExecutionReport::getErrorCode ( ) const [inline]

Get Error Code.

Returns

Error Code. This field is useful when dealing with BSE.

#### 5.4.2.4 Int32 mutrade::ExecutionReport::getLastFillPrice ( ) const [inline]

Get Last Fill Price.

Returns

Last Fill Price.

5.4.2.5 `Int32 mutrade::ExecutionReport::getLastFillQuantity ( ) const [inline]`

Get Last Fill Quantity.

Returns

Filled Quantity.

5.4.2.6 `OrderMode mutrade::ExecutionReport::getOrderMode ( ) const [inline]`

Get [Order](#) Mode.

Returns

[OrderMode](#) Buy or sell order.

5.4.2.7 `Int32 mutrade::ExecutionReport::getOrderQuantity ( ) const [inline]`

Get Orde Quantity.

Returns

Ordered qty.

5.4.2.8 `Int64 mutrade::ExecutionReport::getOriginalCIOrderId ( ) const [inline]`

Get Original Original Id.

Returns

Original Ordered Id. User must update this field while modifying the order.

5.4.2.9 `Int64 mutrade::ExecutionReport::getSymbolId ( ) const [inline]`

Get Symbol Id.

Returns

Symbol Id.

5.4.2.10 `String mutrade::ExecutionReport::getTradeId ( ) const [inline]`

Get [Trade](#) Id.

Returns

[Trade](#) Id.

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

- `types.h`

## 5.5 mutrade::ExecutionResponse Class Reference

Execution Response.

```
#include <types.h>
```

### Public Member Functions

- **ExecutionResponse** (const char \*buf)
- void **dump** ()
- UInt64 **getClOrderId** () const
- UChar **getTransactionType** () const
- UChar **getResponseTypes** () const
- UInt32 **getTokenId** () const
- void **setClOrderId** (UInt64 clOrderId)
- void **setTransactionType** (UChar transactionType)
- void **setResponseType** (UChar responseType)
- void **setTokenId** (UInt32 val)

### 5.5.1 Detailed Description

Execution Response.

Internally used by API.

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

- types.h

## 5.6 mutrade::Instrument Class Reference

### Public Member Functions

- **Instrument** ()  
*Create an instrument from string identifier.*
- **Instrument** (const std::string &identifier)  
*Create an instrument from string identifier.*
- InstrumentType **getInstrumentType** () const  
*Get Type of instrument (STOCK/FUTURE/OPTION)*
- Int64 **getStrikePrice** () const  
*Get Expiry Date of the instrument (for FUTURE/OPTION)*
- OptionType **getOptionType** () const  
*Get Type of the option - CALL / PUT (for OPTIONS)*
- String **getSeries** () const  
*Get Series of instrument.*
- Int32 **getLotSize** () const  
*Get Lot Size of the instrument (for FUTURE/OPTION)*
- Int32 **getTickSize** () const  
*Get Tick Size for instrument.*
- bool **operator<** (const **Instrument** &rhs) const
- bool **operator==** (const **Instrument** &rhs) const
- String **getInstrumentName** ()  
*Get Instrument name as string.*

### 5.6.1 Member Function Documentation

#### 5.6.1.1 Int64 mutrade::Instrument::getStrikePrice ( ) const

Get Expiry Date of the instrument (for FUTURE/OPTION)

Get Strike Price of the option (for OPTIONS)

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

- instrument.h

## 5.7 mutrade::Logger Class Reference

Abstract [Logger](#) class.

```
#include <logger.h>
```

### Public Member Functions

- void [setLogLevel](#) (LogLevel level)  
*Set Log Level.*
- void **log** (LogLevel level, const std::string &message)
- template<typename T >  
void **log** (LogLevel level, const std::string &message, const T &param)
- template<typename T1 , typename T2 >  
void **log** (LogLevel level, const std::string &message, const T1 &param1, const T2 &param2)
- template<typename T1 , typename T2 , typename T3 >  
void **log** (LogLevel level, const std::string &message, const T1 &param1, const T2 &param2, const T3 &param3)
- template<typename T1 , typename T2 , typename T3 , typename T4 >  
void **log** (LogLevel level, const std::string &message, const T1 &param1, const T2 &param2, const T3 &param3, const T4 &param4)
- template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >  
void **log** (LogLevel level, const std::string &message, const T1 &param1, const T2 &param2, const T3 &param3, const T4 &param4, const T5 &param5)

### Static Public Member Functions

- static [Logger](#) \* [getInstance](#) ()  
*Get an Instance of the [Context](#) class.*

#### 5.7.1 Detailed Description

Abstract [Logger](#) class.

Singleton Logging class.

This is the class which should be used in code to use the logging functionality. The parameters to be used in the log function, must have stream operators available.

#### 5.7.2 Member Function Documentation



5.7.2.1 static **Logger\*** mutrade::Logger::getInstance ( ) [inline],[static]

Get an Instance of the [Context](#) class.

[Logger](#) class is a Singleton class, which will have only one instance. This instance can be accessed using the `getInstance` method.

5.7.2.2 void mutrade::Logger::setLogLevel ( LogLevel *level* ) [inline]

Set Log Level.

#### Parameters

<i>level</i>	
--------------	--

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

- `logger.h`

## 5.8 mutrade::MarketData Class Reference

[MarketData](#) Class.

```
#include <marketdata.h>
```

### Public Member Functions

- **MarketData** (const [Quote](#) &)
- [Instrument](#) **getInstrument** () const
- Int32 **getLastPrice** () const  
*Last Traded Price of the [Instrument](#).*
- Int32 **getLastQty** () const  
*Last Traded Quantity of the [Instrument](#).*
- Int32 **getLastTime** () const  
*Time of last trade.*
- Int32 **getTotalQty** () const  
*Total Quantity traded in the day.*
- Int32 **getDepth** (Side side) const  
*Depth available on Bid/Ask side.*
- Int32 **getPrice** (Side side, Int32 rank)  
*Get Price available at Rank on Bid/Ask side.*
- Int32 **getQty** (Side side, Int32 rank)  
*Get Quantity available at Rank on Bid/Ask side.*
- Int32 **getRank** (Side side, Int32 price) const  
*Get Rank in Market depth for a particular price.*
- Boolean **getCount** (Side side, Int32 rank) const  
*get [Order](#) count at Bid/Ask side*
- Boolean **hasQty** (Side side, Int32 qty) const  
*Check if a particular qty is available at Bid/Ask side.*
- Int32 **getAvgPrice** (Side side, Int32 qty) const  
*Get Best average price for a particular quantity.*
- Int32 **getQtyForAvgPrice** (Side side, Int32 avgPrice) const

- Get maximum quantity available at Average Price.*

  - Int32 [getAvgPriceForQty](#) (Side side, Int32 qty) const

*Get average price for a particular quantity.*

  - Int32 [getQtyForWorstPrice](#) (Side side, Int32 worstPrice) const

*Get maximum quantity at worstPrice or better.*

  - Int32 [getWorstPriceForQty](#) (Side side, Int32 qty) const

*Get Worst price for a particular quantity.*

  - Int32 [getDayOpen](#) () const

*Get Day's Open Price.*

  - Int32 [getDayHigh](#) () const

*Get Day's High Price.*

  - Int32 [getDayLow](#) () const

*Get Day's Low Price.*

  - Int32 [getDayClose](#) () const

*Get Previous Day's Close Price.*

### 5.8.1 Detailed Description

[MarketData](#) Class.

### 5.8.2 Member Function Documentation

#### 5.8.2.1 Int32 mutrade::MarketData::getAvgPrice ( Side side, Int32 qty ) const

Get Best average price for a particular quantity.

Get Best average price available for a particular quantity

#### 5.8.2.2 Int32 mutrade::MarketData::getAvgPriceForQty ( Side side, Int32 qty ) const

Get average price for a particular quantity.

Get Average Price for a particular quantity which is available on Bid/Ask side

#### 5.8.2.3 Boolean mutrade::MarketData::getCount ( Side side, Int32 rank ) const

get [Order](#) count at Bid/Ask side

Get [Order](#) count at Bid/Ask side. This data may not be available for all exchanges.

#### 5.8.2.4 Int32 mutrade::MarketData::getDayClose ( ) const

Get Previous Day's Close Price.

Get Previous Day's Close Price

#### 5.8.2.5 Int32 mutrade::MarketData::getDayHigh ( ) const

Get Day's High Price.

Get Day's High Price

**5.8.2.6 Int32 mutrade::MarketData::getDayLow ( ) const**

Get Day's Low Price.

Get Day's Low Price

**5.8.2.7 Int32 mutrade::MarketData::getDayOpen ( ) const**

Get Day's Open Price.

Get Day's Open Price

**5.8.2.8 Int32 mutrade::MarketData::getDepth ( Side *side* ) const**

Depth available on Bid/Ask side.

Depth available on Bid/Ask side

**5.8.2.9 Int32 mutrade::MarketData::getLastPrice ( ) const**

Last Traded Price of the [Instrument](#).

Last Traded Price of the [Instrument](#)

**5.8.2.10 Int32 mutrade::MarketData::getLastQty ( ) const**

Last Traded Quantity of the [Instrument](#).

Last Traded Quantity of the [Instrument](#)

**5.8.2.11 Int32 mutrade::MarketData::getLastTime ( ) const**

Time of last trade.

Time of last trade

**5.8.2.12 Int32 mutrade::MarketData::getPrice ( Side *side*, Int32 *rank* )**

Get Price available at Rank on Bid/Ask side.

Get Price available at Rank on Bid/Ask side

**5.8.2.13 Int32 mutrade::MarketData::getQty ( Side *side*, Int32 *rank* )**

Get Quantity available at Rank on Bid/Ask side.

Get Quantity available at Rank on Bid/Ask side

**5.8.2.14 Int32 mutrade::MarketData::getQtyForAvgPrice ( Side *side*, Int32 *avgPrice* ) const**

Get maximum quantity available at Average Price.

Get Maximum Quantity which is available on Bid/Ask side at specified Average Price or better.

#### 5.8.2.15 `Int32 mutrade::MarketData::getQtyForWorstPrice ( Side side, Int32 worstPrice ) const`

Get maximum quantity at worstPrice or better.

Get Maximum Quantity which is available on Bid/Ask side for Worst Price or better.

#### 5.8.2.16 `Int32 mutrade::MarketData::getRank ( Side side, Int32 price ) const`

Get Rank in Market depth for a particular price.

Get Rank in Market depth for a particular price

#### 5.8.2.17 `Int32 mutrade::MarketData::getTotalQty ( ) const`

Total Quantity traded in the day.

Total Quantity traded in the day. This data may not be provided by all the exchanges.

#### 5.8.2.18 `Int32 mutrade::MarketData::getWorstPriceForQty ( Side side, Int32 qty ) const`

Get Worst price for a particular quantity.

Get Worst Price which is available on Bid/Ask side for a particular quantity

#### 5.8.2.19 `Boolean mutrade::MarketData::hasQty ( Side side, Int32 qty ) const`

Check if a particular qty is available at Bid/Ask side.

Check if a particular qty is available at Bid/Ask side

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

- `marketdata.h`

## 5.9 `mutrade::MarketDataSubscription` Class Reference

### Public Member Functions

- void **subscribe** (int userId, long symbolId)
- void **unsubscribe** (int userId, long symbolId)
- bool **isSubscribed** (int userId, long symbolId)

### Static Public Member Functions

- static `MarketDataSubscription` \* **getInstance** ()

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

- `mdSubscription.h`

## 5.10 `mutrade::NetPositions` Class Reference

`NetPositions` class.

```
#include <netpositions.h>
```

## Public Member Functions

- [Position](#) \* [getPosition](#) ([Instrument](#) &instrument, Side orderMode) throw (std::domain\_error)  
*Get Net Positions for an [Instrument](#) and Side.*
- int [update](#) ([ExecutionReport](#) &report)  
*Updates the position in the [NetPositions](#).*

### 5.10.1 Detailed Description

[NetPositions](#) class.

This class stores the list of all the positions which the client has accumulated through the trading day.

#### Note

The trades which happened before the connection was made can be replayed back from the server and this class will then be able to provide the net positions for the day.

### 5.10.2 Member Function Documentation

#### 5.10.2.1 [Position](#)\* mutrade::NetPositions::getPosition ( [Instrument](#) & instrument, Side orderMode ) throw (std::domain\_error)

Get Net Positions for an [Instrument](#) and Side.

#### Parameters

<i>instrument</i>	
<i>side</i>	( BUY/SELL )

#### 5.10.2.2 int mutrade::NetPositions::update ( [ExecutionReport](#) & report )

Updates the position in the [NetPositions](#).

This method updates the positions which are received as executions from the exchange.

#### Note

The user of the API does not need to call this method. It is called by the API automatically when an execution is received.

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

- netpositions.h

## 5.11 mutrade::Order Class Reference

[Order](#) Class.

```
#include <order.h>
```

## Public Member Functions

- Int64 [getClOrdId](#) ()

- Client order Id.*
- TransactionType [getTransactionType](#) ()
  - Trasnaction Type.*
- Int64 **getOrigCLOrdId** ()
- String [getExchangeOrderId](#) ()
  - Exchange Order Id.*
- String [getSymbol](#) ()
  - Instrument name.*
- OrderMode [getOrderMode](#) ()
  - Order Mode.*
- Int32 [getQuantity](#) ()
  - Order Quantity.*
- Int32 **getDisclosedQuantity** ()
- Int32 [getFilledQuantity](#) ()
  - Filled quantity.*
- Int32 **getOldQuantity** ()
- Int32 [getPrice](#) ()
  - Order Price.*
- Int32 [getStopPrice](#) ()
  - Stop Price.*
- UChar [getSecurityType](#) ()
  - Instrument Type.*
- TimeInForce [getOrderValidity](#) ()
  - Time in force.*
- OrderType [getOrderType](#) ()
  - Order Type.*
- OrderStatus [getOrderStatus](#) ()
  - Order Status.*
- Int64 **getExchangeEntryTime** ()
- Int64 **getExchangeModifyTime** ()
- void [setCLOrdId](#) (Int64 val)
  - Set Client Order Id.*
- void [setTransactionType](#) (TransactionType val)
  - Set Transaction Type.*
- void **setOrigCLOrdId** (Int64 val)
- void **setExchangeOrderId** (String val)
- void [setSymbol](#) (String val)
  - Set Symbol.*
- void [setOrderMode](#) (OrderMode val)
  - Set Order Mode.*
- void [setQuantity](#) (Int32 val)
  - Set Order Quantity.*
- void **setDisclosedQuantity** (Int32 val)
- void **setFilledQuantity** (Int32 val)
- void **setOldQuantity** (Int32 val)
- void [setPrice](#) (Int32 val)
  - Set Order Price.*
- void **setStopPrice** (Int32 val)
- void **setSecurityType** (UChar val)
- void **setOrderValidity** (TimeInForce val)
- void [setOrderType](#) (OrderType val)
  - Set Order Type.*

- void [setOrderStatus](#) (OrderStatus val)  
*Set [Order](#) Status.*
- void **setExchangeEntryTime** (Int32 val)
- void **setExchangeModifyTime** (Int32 val)

### 5.11.1 Detailed Description

[Order](#) Class.

User has to set the fields of this class while placing order.(New/Modify/Cancel)

### 5.11.2 Member Function Documentation

#### 5.11.2.1 Int64 mutrade::Order::getClOrdId ( ) [inline]

Client order Id.

Returns

Client order Id.

#### 5.11.2.2 String mutrade::Order::getExchangeOrderId ( ) [inline]

Exchange [Order](#) Id.

Returns

Exchange order Id.

#### 5.11.2.3 Int32 mutrade::Order::getFilledQuantity ( ) [inline]

Filled quantity.

Returns

Filled qty.

#### 5.11.2.4 OrderMode mutrade::Order::getOrderMode ( ) [inline]

[Order](#) Mode.

Returns

[OrderMode](#)

#### 5.11.2.5 OrderStatus mutrade::Order::getOrderStatus ( ) [inline]

[Order](#) Status.

Returns

[OrderStatus](#)

5.11.2.6 `OrderType mutrade::Order::getOrderType ( ) [inline]`

[Order](#) Type.

Returns

`OrderType_LIMIT/OrderType_MARKET/OrderType_STOP_LIMIT`.

5.11.2.7 `TimelnForce mutrade::Order::getOrderValidity ( ) [inline]`

Time in force.

Returns

`TimelnForce_DAY/TimelnForce_IOC`.

5.11.2.8 `Int32 mutrade::Order::getPrice ( ) [inline]`

[Order](#) Price.

Returns

[Order](#) Price.

5.11.2.9 `Int32 mutrade::Order::getQuantity ( ) [inline]`

[Order](#) Quantity.

Returns

[Order](#) quantity.

5.11.2.10 `UChar mutrade::Order::getSecurityType ( ) [inline]`

[Instrument](#) Type.

Returns

`InstrumentType_STOCK/InstrumentType_FUTURE/ InstrumentType_OPTION`.

5.11.2.11 `Int32 mutrade::Order::getStopPrice ( ) [inline]`

Stop Price.

Returns

Stop Price.

5.11.2.12 `String mutrade::Order::getSymbol ( ) [inline]`

[Instrument](#) name.

Returns

[Instrument](#) name.



5.11.2.13 `TransactionType mutrade::Order::getTransactionType ( ) [inline]`

Trasnsaction Type.

Returns

[TransactionType](#) [New/Modify/Cancel]

5.11.2.14 `void mutrade::Order::setClOrdId ( Int64 val ) [inline]`

Set Client [Order](#) Id.

Parameters

<i>val</i>	
------------	--

5.11.2.15 `void mutrade::Order::setOrderMode ( OrderMode val ) [inline]`

Set [Order](#) Mode.

Parameters

<i>val</i>	<a href="#">OrderMode</a> User must set this field while placing New <a href="#">Order</a> .
------------	--

5.11.2.16 `void mutrade::Order::setOrderStatus ( OrderStatus val ) [inline]`

Set [Order](#) Status.

Parameters

<i>val</i>	Interanally updated by API.
------------	-----------------------------

5.11.2.17 `void mutrade::Order::setOrderType ( OrderType val ) [inline]`

Set [Order](#) Type.

Parameters

<i>val</i>	User must set this field for transaction type New/Modify.
------------	---

5.11.2.18 `void mutrade::Order::setPrice ( Int32 val ) [inline]`

Set [Order](#) Price.

Parameters

<i>val</i>	User must set this field for transaction type New/Modify.
------------	---

5.11.2.19 `void mutrade::Order::setQuantity ( Int32 val ) [inline]`

Set [Order](#) Quantity.

## Parameters

<i>val</i>	User must set this field for transaction type New/Modify.
------------	---

5.11.2.20 `void mutrade::Order::setSymbol ( String val )` `[inline]`

Set Symbol.

## Parameters

<i>val</i>	<a href="#">Instrument</a> Name. User must set this field in case of New <a href="#">Order</a> type.
------------	--

5.11.2.21 `void mutrade::Order::setTransactionType ( TransactionType val )` `[inline]`

Set Transaction Type.

## Parameters

<i>val</i>	User must update this field accordingly. [New/Modify/Cancel]
------------	--

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

- `order.h`

## 5.12 mutrade::OrderBook Class Reference

[OrderBook](#) class.

```
#include <orderbook.h>
```

### Public Member Functions

- `Order * getOrder (Int64 clOrderId) throw (std::domain_error)`  
*Get the order details.*
- `int update (ExecutionReport &report, bool reconcileOldOrders=false)`  
*Updates the trade in the [OrderBook](#).*
- `void insert (Order *order)`  
*Insterts the order in the [OrderBook](#).*

### 5.12.1 Detailed Description

[OrderBook](#) class.

This class stores the list of all the orders which have been placed during the day.

#### Note

Only the orders placed during the current session will be available from this class. Orders placed before the connection was made will not be available via this class.

### 5.12.2 Member Function Documentation

#### 5.12.2.1 `Order*` mutrade::OrderBook::getOrder ( `Int64 cOrderId` ) throw (std::domain\_error)

Get the order details.

##### Parameters

<code>cOrderId</code>	(client order ID generated by the server)
-----------------------	---

#### 5.12.2.2 `void` mutrade::OrderBook::insert ( `Order * order` )

Insterts the order in the [OrderBook](#).

##### Note

The user of the API does not need to call this method. It is called by the API automatically when an execution is received.

#### 5.12.2.3 `int` mutrade::OrderBook::update ( `ExecutionReport & report`, `bool reconcileOldOrders = false` )

Updates the trade in the [OrderBook](#).

This method updates the order which are sent by the API. The user of the API does not need to call this method. It is called by the API automatically when an order is placed.

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

- `orderbook.h`

## 5.13 mutrade::Portfolio Class Reference

[Portfolio](#) class.

```
#include <portfolio.h>
```

### Public Member Functions

- `void` [insert](#) ([mutrade::Order](#) \*order)  
*Insterts the order in the [Portfolio](#).*
- `void` [handleResponse](#) ([mutrade::ExecutionResponse](#) \*rsp)  
*Handle Response from the server.*
- `void` [handleConfirmations](#) ([mutrade::ExecutionReport](#) \*conf, UNSIGNED\_SHORT responseCategory)  
*Handle Confirmations from the server.*
- [mutrade::Order](#) \* [getOrderByTokenId](#) (Int32 tokenId)  
*Gets [Order](#) From TokenId.*
- [NetPositions](#) & [getNetPositions](#) ()  
*Get cumulative Net Positions.*
- [OrderBook](#) & [getOrderBook](#) ()  
*Get [Order](#) Book (list of all the orders placed)*
- [TradeBook](#) & [getTradeBook](#) ()  
*Get [Trade](#) Book (list of all the trades placed)*

## Static Public Member Functions

- static [Portfolio](#) \* [getInstance](#) ()

*Get an Instance of the [Portfolio](#) class.*

### 5.13.1 Detailed Description

[Portfolio](#) class.

This class contains the portfolio for the trader/algorithm. [Portfolio](#) class provides [OrderBook](#), [TradeBook](#) and Net Positions for the trader.

### 5.13.2 Member Function Documentation

#### 5.13.2.1 static [Portfolio](#)\* [mutrade::Portfolio::getInstance](#) ( ) `[inline],[static]`

Get an Instance of the [Portfolio](#) class.

[Portfolio](#) class is singleton class, which will have only one instance. This instance can be accessed using the `getInstance` method.

#### 5.13.2.2 [mutrade::Order](#)\* [mutrade::Portfolio::getOrderByTokenId](#) ( [Int32](#) *tokenId* )

Gets [Order](#) From [TokenId](#).

#### Note

The user of the API does not need to call this method. It is called by the API automatically when an execution is received.

#### 5.13.2.3 void [mutrade::Portfolio::handleConfirmations](#) ( [mutrade::ExecutionReport](#) \* *conf*, [UNSIGNED\\_SHORT](#) *responseCategory* )

Handle Confirmations from the server.

#### Note

The user of the API does not need to call this method. It is called by the API automatically when an execution is received.

#### 5.13.2.4 void [mutrade::Portfolio::handleResponse](#) ( [mutrade::ExecutionResponse](#) \* *rsp* )

Handle Response from the server.

#### Note

The user of the API does not need to call this method. It is called by the API automatically when an execution is received.

#### 5.13.2.5 void [mutrade::Portfolio::insert](#) ( [mutrade::Order](#) \* *order* )

Insterts the order in the [Portfolio](#).

**Note**

The user of the API does not need to call this method. It is called by the API automatically when an execution is received.

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

- portfolio.h

## 5.14 mutrade::Position Class Reference

[Position](#) class.

```
#include <position.h>
```

### Public Member Functions

- [Position](#) ([Instrument](#) &instrument)  
*Position.*
- [Position](#) (const [Position](#) &)  
*Position class copy constructor.*
- void [initialize](#) ()  
*Initialize class members with default values.*
- bool [operator<](#) (const [Position](#) &rhs) const  
*Overloaded comparison operator, in order to insert positions in NetPosition map.*
- Int32 [getQuantity](#) ()  
*Get total quantity for current position.*
- Int32 [getAveragePrice](#) ()  
*Get Average Price for current position.*
- [Instrument](#) [getInstrument](#) ()  
*Get Instrument from current position.*
- Side [getOrderMode](#) ()  
*Get Side of current position.*
- void [setQuantity](#) (Int32 val)  
*Set total quantity for current position.*
- void [setAveragePrice](#) (Int32 val)  
*Set Average Price for current position.*
- void [setInstrument](#) ([Instrument](#) val)  
*Set Instrument from current position.*
- void [setOrderType](#) (Side val)  
*Set Side of current position.*

### 5.14.1 Detailed Description

[Position](#) class.

This class is required for [NetPositions](#) class.

**Note**

We need to create an object of type [Position](#) before calling NetPosition.

### 5.14.2 Constructor & Destructor Documentation

#### 5.14.2.1 `mutrade::Position::Position ( Instrument & instrument )`

[Position](#).

##### Parameters

	c	<a href="#">Instrument</a> for which we will keep track of postion.
--	---	---

#### 5.14.2.2 `mutrade::Position::Position ( const Position & )`

[Position](#) class copy constructor.

##### Parameters

	c	<a href="#">Position</a> object to copy.
--	---	--

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

- `position.h`

## 5.15 `mutrade::Quote` Class Reference

### Public Member Functions

- **Quote** (const [Quote](#) &)
- [Quote](#) & **operator=** (const [Quote](#) &q)
- void **setSymbolId** (Int64 val)
- void **setNummberOfTrades** (Int64 val)
- void **setVolume** (Int64 val)
- void **setValue** (Int64 val)
- void **setLastTradePrice** (Int64 val)
- void **setLastTradeQty** (Int64 val)
- void **setOpenPrice** (Int64 val)
- void **setClosePrice** (Int64 val)
- void **setHighPrice** (Int64 val)
- void **setLowPrice** (Int64 val)
- void **setTotalBidQty** (Int64 val)
- void **setTotalAskQty** (Int64 val)
- void **setLowerCktLimit** (Int64 val)
- void **setUpperCktLimit** (Int64 val)
- void **setDepth** (UChar val)
- void **setBidPrice** (Int64 val[])
- void **setBidQty** (Int64 val[])
- void **setAskPrice** (Int64 val[])
- void **setAskQty** (Int64 val[])
- Int64 **getSymbolId** () const
- Int64 **getNumberOfTrades** () const
- Int64 **getVolume** () const
- Int64 **getValue** () const
- Int64 **getLastTradePrice** () const
- Int64 **getLastTradeQty** () const

- Int64 **getOpenPrice** () const
- Int64 **getClosePrice** () const
- Int64 **getHighPrice** () const
- Int64 **getLowPrice** () const
- Int64 **getTotalBidQty** () const
- Int64 **getTotalAskQty** () const
- Int64 **getLowerCktLimit** () const
- Int64 **getUpperCktLimit** () const
- UChar **getDepth** () const
- Int64 \* **getBidPrice** ()
- Int64 \* **getBidQty** ()
- Int64 \* **getAskPrice** ()
- Int64 \* **getAskQty** ()

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

- quotes.h

## 5.16 mutrade::Trade Class Reference

### Public Member Functions

- **Trade** ([Instrument](#) &)
- **Trade** (const [Trade](#) &)
- void **initialize** ()
- [Instrument](#) **getInstrument** ()
- String **getTradeld** ()
- Int64 **getCIOrdId** ()
- Int64 **getOrigCIOrdId** ()
- String **getExchangeOrderId** ()
- Side **getOrderMode** ()
- Int32 **getFilledQuantity** ()
- Int32 **getFilledPrice** ()
- OrderType **getOrderType** ()
- Int32 **getTradeTime** ()
- void **setInstrument** ([Instrument](#) val)
- void **setTradeld** (String val)
- void **setCIOrdId** (Int64 val)
- void **setOrigCIOrdId** (Int64 val)
- void **setExchangeOrderId** (String val)
- void **setOrderMode** (Side val)
- void **setFilledQuantity** (Int32 val)
- void **setFilledPrice** (Int32 val)
- void **setOrderType** (OrderType val)
- void **setTradeTime** (Int32 val)
- void **handleConfirmations** (RSP::OrderConfirmation \*confirmation)

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

- trade.h

## 5.17 mutrade::TradeBook Class Reference

[TradeBook](#) class.

```
#include <tradebook.h>
```

### Public Member Functions

- TradeList \* [getTrades](#) (Int64 clOrderId) throw (std::domain\_error)  
*Get List of trades.*
- TradeQue \* **getTradeQue** (Int64 clOrderId) throw (std::domain\_error)
- int [update](#) ([ExecutionReport](#) &report)  
*Updates the trade in the [TradeBook](#).*

### 5.17.1 Detailed Description

[TradeBook](#) class.

This class stores the list of all the trades which have happened during the day.

#### Note

The trades which happened before the connection was made can be replayed back from the server and this class will then be able to serve the list of all trades happened during the day.

### 5.17.2 Member Function Documentation

#### 5.17.2.1 TradeList\* mutrade::TradeBook::getTrades ( Int64 clOrderId ) throw (std::domain\_error)

Get List of trades.

#### Parameters

<i>clOrderId</i>	(client order ID generated by the server)
------------------	---

#### 5.17.2.2 int mutrade::TradeBook::update ( ExecutionReport & report )

Updates the trade in the [TradeBook](#).

This method updates the trades which are received as executions from the exchange.

#### Note

The user of the API does not need to call this method. It is called by the API automatically when an execution is received.

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

- tradebook.h



# Index

- enableLogging
  - mutrade::Context, [11](#)
- getAvgPrice
  - mutrade::MarketData, [20](#)
- getAvgPriceForQty
  - mutrade::MarketData, [20](#)
- getClOrdId
  - mutrade::Order, [25](#)
- getClOrderId
  - mutrade::ExecutionReport, [15](#)
- getClientId
  - mutrade::ExecutionReport, [15](#)
- getCount
  - mutrade::MarketData, [20](#)
- getDayClose
  - mutrade::MarketData, [20](#)
- getDayHigh
  - mutrade::MarketData, [20](#)
- getDayLow
  - mutrade::MarketData, [20](#)
- getDayOpen
  - mutrade::MarketData, [21](#)
- getDepth
  - mutrade::MarketData, [21](#)
- getErrorCode
  - mutrade::ExecutionReport, [15](#)
- getExchangeOrderId
  - mutrade::Order, [25](#)
- getFilledQuantity
  - mutrade::Order, [25](#)
- getInstance
  - mutrade::Context, [11](#)
  - mutrade::Logger, [18](#)
  - mutrade::Portfolio, [30](#)
- getInstrument
  - mutrade::Context, [12](#)
- getLastFillPrice
  - mutrade::ExecutionReport, [15](#)
- getLastFillQuantity
  - mutrade::ExecutionReport, [15](#)
- getLastPrice
  - mutrade::MarketData, [21](#)
- getLastQty
  - mutrade::MarketData, [21](#)
- getLastTime
  - mutrade::MarketData, [21](#)
- getOrder
  - mutrade::OrderBook, [28](#)
- getOrderByTokenId
  - mutrade::Portfolio, [30](#)
- getOrderMode
  - mutrade::ExecutionReport, [16](#)
  - mutrade::Order, [25](#)
- getOrderQuantity
  - mutrade::ExecutionReport, [16](#)
- getOrderStatus
  - mutrade::Order, [25](#)
- getOrderType
  - mutrade::Order, [25](#)
- getOrderValidity
  - mutrade::Order, [26](#)
- getOriginalClOrderId
  - mutrade::ExecutionReport, [16](#)
- getPosition
  - mutrade::NetPositions, [23](#)
- getPrice
  - mutrade::MarketData, [21](#)
  - mutrade::Order, [26](#)
- getQty
  - mutrade::MarketData, [21](#)
- getQtyForAvgPrice
  - mutrade::MarketData, [21](#)
- getQtyForWorstPrice
  - mutrade::MarketData, [21](#)
- getQuantity
  - mutrade::Order, [26](#)
- getRank
  - mutrade::MarketData, [22](#)
- getSecurityType
  - mutrade::Order, [26](#)
- getStopPrice
  - mutrade::Order, [26](#)
- getStrikePrice
  - mutrade::Instrument, [18](#)
- getSymbol
  - mutrade::Order, [26](#)
- getSymbolId
  - mutrade::ExecutionReport, [16](#)
- getTotalQty
  - mutrade::MarketData, [22](#)
- getTradeId
  - mutrade::ExecutionReport, [16](#)
- getTrades
  - mutrade::TradeBook, [34](#)
- getTransactionType
  - mutrade::Order, [26](#)
- getWorstPriceForQty
  - mutrade::MarketData, [22](#)

- handleConfirmations
  - mutrade::Portfolio, 30
- handleResponse
  - mutrade::Portfolio, 30
- hasQty
  - mutrade::MarketData, 22
- insert
  - mutrade::OrderBook, 29
  - mutrade::Portfolio, 30
- loadInstrument
  - mutrade::Context, 12
- login
  - mutrade::Context, 12
- logout
  - mutrade::Context, 12
- mutrade::AbstractLogger, 9
- mutrade::Application, 9
  - onExecutionReport, 10
  - onLoadInstrumentEnd, 10
  - onLogin, 10
  - onLogout, 10
  - onTick, 10
- mutrade::Context, 10
  - enableLogging, 11
  - getInstance, 11
  - getInstrument, 12
  - loadInstrument, 12
  - login, 12
  - logout, 12
  - placeOrder, 12
  - setApplication, 13
  - subscribe, 13
  - unsubscribe, 13
- mutrade::ExecutionReport, 13
  - getCIOrderId, 15
  - getClientId, 15
  - getErrorCode, 15
  - getLastFillPrice, 15
  - getLastFillQuantity, 15
  - getOrderMode, 16
  - getOrderQuantity, 16
  - getOriginalCIOrderId, 16
  - getSymbolId, 16
  - getTradId, 16
- mutrade::ExecutionResponse, 17
- mutrade::Instrument, 17
  - getStrikePrice, 18
- mutrade::Logger, 18
  - getInstance, 18
  - setLogLevel, 19
- mutrade::MarketData, 19
  - getAvgPrice, 20
  - getAvgPriceForQty, 20
  - getCount, 20
  - getDayClose, 20
  - getDayHigh, 20
  - getDayLow, 20
  - getDayOpen, 21
  - getDepth, 21
  - getLastPrice, 21
  - getLastQty, 21
  - getLastTime, 21
  - getPrice, 21
  - getQty, 21
  - getQtyForAvgPrice, 21
  - getQtyForWorstPrice, 21
  - getRank, 22
  - getTotalQty, 22
  - getWorstPriceForQty, 22
  - hasQty, 22
- mutrade::MarketDataSubscription, 22
- mutrade::NetPositions, 22
  - getPosition, 23
  - update, 23
- mutrade::Order, 23
  - getCIOrdId, 25
  - getExchangeOrderId, 25
  - getFilledQuantity, 25
  - getOrderMode, 25
  - getOrderStatus, 25
  - getOrderType, 25
  - getOrderValidity, 26
  - getPrice, 26
  - getQuantity, 26
  - getSecurityType, 26
  - getStopPrice, 26
  - getSymbol, 26
  - getTransactionType, 26
  - setCIOrdId, 27
  - setOrderMode, 27
  - setOrderStatus, 27
  - setOrderType, 27
  - setPrice, 27
  - setQuantity, 27
  - setSymbol, 28
  - setTransactionType, 28
- mutrade::OrderBook, 28
  - getOrder, 28
  - insert, 29
  - update, 29
- mutrade::Portfolio, 29
  - getInstance, 30
  - getOrderByTokenId, 30
  - handleConfirmations, 30
  - handleResponse, 30
  - insert, 30
- mutrade::Position, 31
  - Position, 32
- mutrade::Quote, 32
- mutrade::Trade, 33
- mutrade::TradeBook, 34
  - getTrades, 34
  - update, 34
- mutrade::detail, 7

- onExecutionReport
  - mutrade::Application, [10](#)
- onLoadInstrumentEnd
  - mutrade::Application, [10](#)
- onLogin
  - mutrade::Application, [10](#)
- onLogout
  - mutrade::Application, [10](#)
- onTick
  - mutrade::Application, [10](#)
- placeOrder
  - mutrade::Context, [12](#)
- Position
  - mutrade::Position, [32](#)
- setApplication
  - mutrade::Context, [13](#)
- setClOrdId
  - mutrade::Order, [27](#)
- setLogLevel
  - mutrade::Logger, [19](#)
- setOrderMode
  - mutrade::Order, [27](#)
- setOrderStatus
  - mutrade::Order, [27](#)
- setOrderType
  - mutrade::Order, [27](#)
- setPrice
  - mutrade::Order, [27](#)
- setQuantity
  - mutrade::Order, [27](#)
- setSymbol
  - mutrade::Order, [28](#)
- setTransactionType
  - mutrade::Order, [28](#)
- subscribe
  - mutrade::Context, [13](#)
- unsubscribe
  - mutrade::Context, [13](#)
- update
  - mutrade::NetPositions, [23](#)
  - mutrade::OrderBook, [29](#)
  - mutrade::TradeBook, [34](#)