

Reference Manual

Generated by Doxygen 1.6.1

Sun Dec 22 13:20:22 2013

Contents

1	muTrade API documentation	1
1.1	Introduction	1
2	Class Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	Class Documentation	7
4.1	Application Class Reference	7
4.1.1	Detailed Description	7
4.1.2	Member Function Documentation	8
4.1.2.1	onExecutionReport	8
4.1.2.2	onLoadInstrumentEnd	8
4.1.2.3	onLogin	8
4.1.2.4	onLogout	8
4.1.2.5	onTick	8
4.2	sampleApplication::ApplicationImpl Class Reference	9
4.2.1	Member Function Documentation	9
4.2.1.1	onExecutionReport	9
4.2.1.2	onLogin	9
4.2.1.3	onLogout	10
4.2.1.4	onTick	10
4.3	Context Class Reference	11
4.3.1	Detailed Description	12
4.3.2	Member Function Documentation	12
4.3.2.1	enableLogging	12
4.3.2.2	getInstance	12

4.3.2.3	getInstrument	12
4.3.2.4	loadInstrument	12
4.3.2.5	login	12
4.3.2.6	login	12
4.3.2.7	logout	13
4.3.2.8	placeOrder	13
4.3.2.9	setApplication	14
4.3.2.10	subscribe	14
4.3.2.11	unsubscribe	14
4.4	ExecutionReport Class Reference	15
4.4.1	Detailed Description	16
4.4.2	Member Function Documentation	16
4.4.2.1	getClientId	16
4.4.2.2	getClOrderId	17
4.4.2.3	getErrorCode	17
4.4.2.4	getLastFillPrice	17
4.4.2.5	getLastFillQuantity	17
4.4.2.6	getOrderMode	17
4.4.2.7	getOrderQuantity	17
4.4.2.8	getOriginalClOrderId	18
4.4.2.9	getSymbolId	18
4.4.2.10	getTradeId	18
4.5	ExecutionResponse Class Reference	19
4.5.1	Detailed Description	19
4.6	Instrument Class Reference	20
4.6.1	Detailed Description	20
4.7	Logger Class Reference	22
4.7.1	Detailed Description	22
4.7.2	Member Function Documentation	22
4.7.2.1	getInstance	22
4.7.2.2	setLogLevel	22
4.8	MarketData Class Reference	23
4.8.1	Detailed Description	24
4.8.2	Member Function Documentation	24
4.8.2.1	getAvgPrice	24
4.8.2.2	getAvgPriceForQty	24

4.8.2.3	getCount	24
4.8.2.4	getDayClose	24
4.8.2.5	getDayHigh	24
4.8.2.6	getDayLow	25
4.8.2.7	getDayOpen	25
4.8.2.8	getDepth	25
4.8.2.9	getInstrument	25
4.8.2.10	getLastPrice	25
4.8.2.11	getLastQty	25
4.8.2.12	getLastTime	25
4.8.2.13	getPrice	25
4.8.2.14	getQty	25
4.8.2.15	getQtyForAvgPrice	25
4.8.2.16	getQtyForWorstPrice	25
4.8.2.17	getRank	26
4.8.2.18	getTotalQty	26
4.8.2.19	getWorstPriceForQty	26
4.8.2.20	hasQty	26
4.9	NetPositions Class Reference	27
4.9.1	Detailed Description	27
4.9.2	Member Function Documentation	27
4.9.2.1	getPosition	27
4.9.2.2	update	27
4.10	Order Class Reference	28
4.10.1	Detailed Description	29
4.10.2	Member Function Documentation	30
4.10.2.1	getClOrdId	30
4.10.2.2	getExchangeOrderId	30
4.10.2.3	getFilledQuantity	30
4.10.2.4	getOrderMode	30
4.10.2.5	getOrderStatus	30
4.10.2.6	getOrderType	30
4.10.2.7	getOrderValidity	31
4.10.2.8	getPrice	31
4.10.2.9	getQuantity	31
4.10.2.10	getSecurityType	31

4.10.2.11	getStopPrice	31
4.10.2.12	getSymbol	31
4.10.2.13	getTransactionType	32
4.10.2.14	setClientId	32
4.10.2.15	setOrderMode	32
4.10.2.16	setOrderStatus	32
4.10.2.17	setOrderType	32
4.10.2.18	setPrice	32
4.10.2.19	setQuantity	33
4.10.2.20	setSymbol	33
4.10.2.21	setTransactionType	33
4.11	OrderBook Class Reference	34
4.11.1	Detailed Description	34
4.11.2	Member Function Documentation	34
4.11.2.1	getOrder	34
4.11.2.2	insert	34
4.11.2.3	update	34
4.12	Portfolio Class Reference	35
4.12.1	Detailed Description	35
4.12.2	Member Function Documentation	35
4.12.2.1	getInstance	35
4.12.2.2	getOrderByTokenId	35
4.12.2.3	insert	36
4.13	Position Class Reference	37
4.13.1	Detailed Description	37
4.13.2	Constructor & Destructor Documentation	38
4.13.2.1	Position	38
4.13.2.2	Position	38
4.14	sampleApplication::Program Class Reference	39
4.15	Quote Class Reference	40
4.16	Trade Class Reference	41
4.17	TradeBook Class Reference	42
4.17.1	Detailed Description	42
4.17.2	Member Function Documentation	42
4.17.2.1	getTradeQue	42
4.17.2.2	update	42

4.18 TradeQue Class Reference	43
4.18.1 Detailed Description	44

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

Class Index

2.1 Class Hierarchy

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

Application	7
sampleApplication::ApplicationImpl	9
Context	11
ExecutionReport	15
ExecutionResponse	19
Instrument	20
Logger	22
MarketData	23
NetPositions	27
Order	28
OrderBook	34
Portfolio	35
Position	37
sampleApplication::Program	39
Quote	40
Trade	41
TradeBook	42
TradeQue	43

Chapter 3

Class Index

3.1 Class List

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

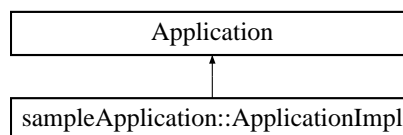
Application (Abstract Application class, to be overridden by the developer)	7
sampleApplication::ApplicationImpl	9
Context (Context class for the algorithm)	11
ExecutionReport (Execution Report Class)	15
ExecutionResponse (Execution Response)	19
Instrument (Instrument class)	20
Logger (Singleton Logging class)	22
MarketData (MarketData Class)	23
NetPositions (NetPositions class)	27
Order (Order Class)	28
OrderBook (OrderBook class)	34
Portfolio (Portfolio class)	35
Position (Position class)	37
sampleApplication::Program	39
Quote	40
Trade	41
TradeBook (TradeBook class)	42
TradeQue (TradeQue class)	43

Chapter 4

Class Documentation

4.1 Application Class Reference

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



Public Member Functions

- virtual void [onTick](#) ([MarketData](#) arg0)
Event called when a tick is received.
- virtual void [onLogin](#) (bool status)
Event called when Login message is returned.
- virtual void [onLogout](#) (bool status)
Event called when Logout message is returned.
- virtual void [onExecutionReport](#) ([ExecutionReport](#) report)
Event called when an execution is received from Server.
- virtual void [onLoadInstrumentEnd](#) (string instrumentName, bool success)
Event called when instrument is loaded from the back-end.

4.1.1 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.

4.1.2 Member Function Documentation

4.1.2.1 `virtual void Application::onExecutionReport (ExecutionReport report) [virtual]`

Event called when an execution is received from Server.

Parameters:

Reimplemented in [sampleApplication::ApplicationImpl](#).

4.1.2.2 `virtual void Application::onLoadInstrumentEnd (string instrumentName, bool success) [virtual]`

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

Parameters:

4.1.2.3 `virtual void Application::onLogin (bool status) [virtual]`

Event called when Login message is returned.

Parameters:

Reimplemented in [sampleApplication::ApplicationImpl](#).

4.1.2.4 `virtual void Application::onLogout (bool status) [virtual]`

Event called when Logout message is returned.

Parameters:

Reimplemented in [sampleApplication::ApplicationImpl](#).

4.1.2.5 `virtual void Application::onTick (MarketData arg0) [virtual]`

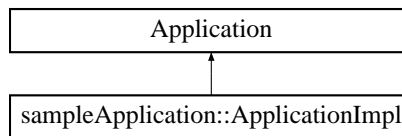
Event called when a tick is received.

Parameters:

Reimplemented in [sampleApplication::ApplicationImpl](#).

4.2 sampleApplication::ApplicationImpl Class Reference

Inheritance diagram for sampleApplication::ApplicationImpl::



Public Member Functions

- **ApplicationImpl** (Data d)
- void **firstLegBid** ()
- void **modifyOrder** ()
- void **cancelFirstLeg** ()
- void **placeSecondLegMarket** ()
- bool **getAveragePrice** ([MarketData](#) md, int qty, Side side, int leg)
- override void **onLogin** (bool status)
Event called when Login message is returned.
- override void **onLogout** (bool status)
Event called when Logout message is returned.
- override void **onTick** ([MarketData](#) md)
Event called when a tick is received.
- override void **onLoadInstrumentEnd** (String instrumentName, bool success)
- override void **onExecutionReport** ([ExecutionReport](#) report)
Event called when an execution is received from Server.

4.2.1 Member Function Documentation

4.2.1.1 override void sampleApplication::ApplicationImpl::onExecutionReport ([ExecutionReport](#) report) [virtual]

Event called when an execution is received from Server.

Parameters:

Reimplemented from [Application](#).

4.2.1.2 override void sampleApplication::ApplicationImpl::onLogin (bool status) [virtual]

Event called when Login message is returned.

Parameters:

Reimplemented from [Application](#).

4.2.1.3 override void sampleApplication::ApplicationImpl::onLogout (bool *status*) [virtual]

Event called when Logout message is returned.

Parameters:

Reimplemented from [Application](#).

4.2.1.4 override void sampleApplication::ApplicationImpl::onTick (MarketData *arg0*) [virtual]

Event called when a tick is received.

Parameters:

Reimplemented from [Application](#).

4.3 Context Class Reference

[Context](#) class for the algorithm.

Public Member Functions

- virtual void **Dispose** ()
- void [login](#) (int userId, string password, string host, short port, bool restoreState)
Login to muTrade server with with userId and password.
- void [login](#) (int userId, string password, string host, short port)
Login to muTrade server with with userId and password.
- void [logout](#) ()
Logout from the muTrade server.
- bool [placeOrder](#) ([Order](#) order)
Send an order to the muTrade server.
- void [enableLogging](#) (LogLevel level)
Enable logging of various events.
- void [subscribe](#) ([Instrument](#) t)
Susbscribe market data for a particular instrument.
- void [unsubscribe](#) ([Instrument](#) t)
Unsusbscribe market data for a previously subscribed instrument.
- void [loadInstrument](#) (string s)
Load static data for an instrument from the muTrade server.
- [Instrument](#) [getInstrument](#) (string t)
Get static data for a particular instrument using symbol [loadInstrument](#) must be called for the string before calling this method.
- [Application](#) [getApplication](#) ()
Get the instance of [Application](#) class.
- void [setApplication](#) ([Application](#) arg0)
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.

4.3.1 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.

4.3.2 Member Function Documentation

4.3.2.1 void Context::enableLogging (LogLevel *level*)

Enable logging of various events.

Parameters:

`\c` 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

4.3.2.2 static Context Context::getInstance () [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.

4.3.2.3 Instrument Context::getInstrument (string *t*)

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

4.3.2.4 void Context::loadInstrument (string *s*)

Load static data for an instrument from the muTrade server.

4.3.2.5 void Context::login (int *userId*, string *password*, string *host*, short *port*)

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

Parameters:

userId to login with

password for user

host ip

port of host

4.3.2.6 void Context::login (int *userId*, string *password*, string *host*, short *port*, bool *restoreState*)

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

Parameters:

userId to login with
password for user
host ip
port of host
restore previous trade

4.3.2.7 void Context::logout ()

Logout from the muTrade server.

4.3.2.8 bool Context::placeOrder (Order order)

Send an order to the muTrade server.

Parameters:

order [Order](#)

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 ClOrdId of previous order.

For **Cancel [Order](#)**

[TransactionType](#) to `TransactionType_CANCEL`.

[Order](#) Status to OrderStatus_PENDING.

Symbol Name with [Instrument](#) name.

setClOrderId to ClOrdId of previous order.

4.3.2.9 void Context::setApplication (Application *arg0*)

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

4.3.2.10 void Context::subscribe (Instrument *t*)

Subscribe market data for a particular instrument. [loadInstrument](#) must be called for the string before calling this method.

4.3.2.11 void Context::unsubscribe (Instrument *t*)

Unsubscribe market data for a previously subscribed instrument. [loadInstrument](#) must be called for the string before calling this method.

4.4 ExecutionReport Class Reference

Execution Report Class.

Public Member Functions

- long [getClOrderId](#) ()
Get Client Order Id.
- long **getExchangeOrderId** ()
- long [getSymbolId](#) ()
Get Symbol Id.
- int [getLastFillQuantity](#) ()
Get Last Fill Quantity.
- int [getLastFillPrice](#) ()
Get Last Fill Price.
- int **getExchangeEntryTime** ()
- int **getExchangeModifyTime** ()
- int **getStrategyId** ()
- int [getClientId](#) ()
Get Client Id.
- int **getLimitPrice** ()
- OrderStatus **getOrderStatus** ()
- OrderMode [getOrderMode](#) ()
Get Order Mode.
- int [getOrderQuantity](#) ()
Get Orde Quantity.
- int **getOrderPrice** ()
- int **getIOCCanceledQuantity** ()
- long [getOriginalClOrderId](#) ()
Get Original Original Id.
- long **getConfirmationTimeSeconds** ()
- long **getConfirmationTimeMicroSeconds** ()
- byte **getIsOffline** ()
- long **getSequenceNumber** ()
- long [getTradeId](#) ()
Get Trade Id.
- int [getErrorCode](#) ()
Get Error Code.
- int **getReasonText** ()

- byte **getUnknownOrder** ()
- string **getInstrumentName** ()
- void **setClOrderId** (long clOrderId)
- void **setExchangeOrderId** (long exchangeOrderId)
- void **setSymbolId** (long symbolId)
- void **setLastFillQuantity** (int qty)
- void **setLastFillPrice** (int price)
- void **setExchangeEntryTime** (int exchangeEntryTime)
- void **setExchangeModifyTime** (int exchangeModifyTime)
- void **setStrategyId** (int strategyId)
- void **setClientId** (int clientId)
- void **setLimitPrice** (int limitPrice)
- void **setOrderStatus** (OrderStatus orderStatus)
- void **setOrderMode** (OrderMode orderMode)
- void **setOrderQuantity** (int quantity)
- void **setOrderPrice** (int price)
- void **setIOCCanceledQuantity** (int quantity)
- void **setOriginalClOrderId** (long originalClOrderId)
- void **setConfirmationTimeSeconds** (long seconds)
- void **setConfirmationTimeMicroSeconds** (long microSeconds)
- void **setIsOffline** (byte isOffline)
- void **setSequenceNumber** (long sequenceNumber)
- void **setTradeId** (long tradeId)
- void **setErrorCode** (int errorCode)
- void **setReasonText** (int reasonText)
- void **setUnknownOrder** (byte unknownOrder)
- void **setInstrumentName** (string instrumentName)
- void **dump** ()
- void **dumpCSV** (SWIGTYPE_p_std__ofstream csvFile)
- int **serialize** (string buf)

4.4.1 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.

4.4.2 Member Function Documentation

4.4.2.1 int ExecutionReport::getClientId ()

Get Client Id.

Returns:

User Id.

4.4.2.2 long ExecutionReport::getClOrderId ()

Get Client [Order](#) Id.

Returns:

Client [Order](#) Id.

4.4.2.3 int ExecutionReport::getErrorCode ()

Get Error Code.

Returns:

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

4.4.2.4 int ExecutionReport::getLastFillPrice ()

Get Last Fill Price.

Returns:

Last Fill Price.

4.4.2.5 int ExecutionReport::getLastFillQuantity ()

Get Last Fill Quantity.

Returns:

Filled Quantity.

4.4.2.6 OrderMode ExecutionReport::getOrderMode ()

Get [Order](#) Mode.

Returns:

[OrderMode](#) Buy or sell order.

4.4.2.7 int ExecutionReport::getOrderQuantity ()

Get Order Quantity.

Returns:

Ordered qty.

4.4.2.8 long ExecutionReport::getOriginalClOrderId ()

Get Original Original Id.

Returns:

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

4.4.2.9 long ExecutionReport::getSymbolId ()

Get Symbol Id.

Returns:

Symbol Id.

4.4.2.10 long ExecutionReport::getTradeId ()

Get [Trade](#) Id.

Returns:

[Trade](#) Id.

4.5 ExecutionResponse Class Reference

Execution Response.

Public Member Functions

- **ExecutionResponse** (string buf)
- void **dump** ()

4.5.1 Detailed Description

Execution Response. Internally used by API.

4.6 Instrument Class Reference

[Instrument](#) class.

Public Member Functions

- [Instrument](#) ()
Create an instrument from string identifier.
- [Instrument](#) (string identifier)
Create an instrument from string identifier.
- InstrumentType [getInstrumentType](#) ()
Get Type of instrument (STOCK/FUTURE/OPTION).
- long [getStrikePrice](#) ()
Get Strike Price of the option (for OPTIONS).
- string [getSeries](#) ()
Get Series of instrument.
- int [getLotSize](#) ()
Get Lot Size of the instrument (for FUTURE/OPTION).
- int [getTickSize](#) ()
Get Tick Size for instrument.
- OptionType [getOptionType](#) ()
Get Option Type [OptionType](#).
- string [getInstrumentName](#) ()
Get [Instrument](#) name as string.

4.6.1 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 DDMONYYYY
- "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

4.7 Logger Class Reference

Singleton Logging class.

Public Member Functions

- void [setLogLevel](#) (LogLevel level)
Set Log Level.
- void **log** (LogLevel level, string message)

Static Public Member Functions

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

4.7.1 Detailed Description

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.

4.7.2 Member Function Documentation

4.7.2.1 static `Logger Logger::getInstance ()` [**static**]

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

4.7.2.2 `void Logger::setLogLevel (LogLevel level)`

Set Log Level.

Parameters:

level

4.8 MarketData Class Reference

[MarketData](#) Class.

Public Member Functions

- **MarketData** ([Quote](#) arg0)
- [Instrument](#) [getInstrument](#) ()
Last Traded Price of the [Instrument](#).
- int [getLastPrice](#) ()
Last Traded Quantity of the [Instrument](#).
- int [getLastQty](#) ()
Time of last trade.
- int [getLastTime](#) ()
Total Quantity traded in the day.
- int [getTotalQty](#) ()
Total Quantity traded in the day.
- int [getDepth](#) (Side side)
Depth available on Bid/Ask side.
- int [getPrice](#) (Side side, int rank)
Get Price available at Rank on Bid/Ask side.
- int [getQty](#) (Side side, int rank)
Get Quantity available at Rank on Bid/Ask side.
- int [getRank](#) (Side side, int price)
Get Rank in Market depth for a particular price.
- bool [getCount](#) (Side side, int rank)
get [Order](#) count at Bid/Ask side
- bool [hasQty](#) (Side side, int qty)
Check if a particular qty is available at Bid/Ask side.
- int [getAvgPrice](#) (Side side, int qty)
Get Best average price for a particular quantity.
- int [getQtyForAvgPrice](#) (Side side, int avgPrice)
Get maximum quantity available at Average Price.
- int [getAvgPriceForQty](#) (Side side, int qty)
Get average price for a particular quantity.

- `int getQtyForWorstPrice` (Side side, int worstPrice)

Get maximum quantity at worstPrice or better.

- `int getWorstPriceForQty` (Side side, int qty)

Get Worst price for a particular quantity.

- `int getDayOpen` ()

Get Day's Open Price.

- `int getDayHigh` ()

Get Day's High Price.

- `int getDayLow` ()

Get Day's Low Price.

- `int getDayClose` ()

Get Previous Day's Close Price.

4.8.1 Detailed Description

[MarketData](#) Class.

4.8.2 Member Function Documentation

4.8.2.1 `int MarketData::getAvgPrice` (Side side, int qty)

Get Best average price for a particular quantity. Get Best average price available for a particular quantity

4.8.2.2 `int MarketData::getAvgPriceForQty` (Side side, int qty)

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

4.8.2.3 `bool MarketData::getCount` (Side side, int rank)

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

4.8.2.4 `int MarketData::getDayClose` ()

Get Previous Day's Close Price. Get Previous Day's Close Price

4.8.2.5 `int MarketData::getDayHigh` ()

Get Day's High Price. Get Day's High Price

4.8.2.6 int MarketData::getDayLow ()

Get Day's Low Price. Get Day's Low Price

4.8.2.7 int MarketData::getDayOpen ()

Get Day's Open Price. Get Day's Open Price

4.8.2.8 int MarketData::getDepth (Side *side*)

Depth available on Bid/Ask side. Depth available on Bid/Ask side

4.8.2.9 Instrument MarketData::getInstrument ()

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

4.8.2.10 int MarketData::getLastPrice ()

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

4.8.2.11 int MarketData::getLastQty ()

Time of last trade. Time of last trade

4.8.2.12 int MarketData::getLastTime ()

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

4.8.2.13 int MarketData::getPrice (Side *side*, int *rank*)

Get Price available at Rank on Bid/Ask side. Get Price available at Rank on Bid/Ask side

4.8.2.14 int MarketData::getQty (Side *side*, int *rank*)

Get Quantity available at Rank on Bid/Ask side. Get Quantity available at Rank on Bid/Ask side

4.8.2.15 int MarketData::getQtyForAvgPrice (Side *side*, int *avgPrice*)

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

4.8.2.16 int MarketData::getQtyForWorstPrice (Side *side*, int *worstPrice*)

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

4.8.2.17 int MarketData::getRank (Side *side*, int *price*)

Get Rank in Market depth for a particular price. Get Rank in Market depth for a particular price

4.8.2.18 int MarketData::getTotalQty ()

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

4.8.2.19 int MarketData::getWorstPriceForQty (Side *side*, int *qty*)

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

4.8.2.20 bool MarketData::hasQty (Side *side*, int *qty*)

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

4.9 NetPositions Class Reference

[NetPositions](#) class.

Public Member Functions

- [Position](#) [getPosition](#) ([Instrument](#) instrument, [Side](#) orderMode)
Get Net Positions for an [Instrument](#) and [Side](#).
- int [update](#) ([ExecutionReport](#) report)
Updates the position in the [NetPositions](#).

4.9.1 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.

4.9.2 Member Function Documentation

4.9.2.1 Position [NetPositions::getPosition](#) ([Instrument](#) *instrument*, [Side](#) *orderMode*)

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

Parameters:

instrument

side (BUY/SELL)

4.9.2.2 int [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.

4.10 Order Class Reference

[Order](#) Class.

Public Member Functions

- int **getClientId** ()
- TransactionType **getTransactionType** ()
Trasnaction Type.
- long **getClOrdId** ()
Client order Id.
- long **getOrigClOrdId** ()
- string **getExchangeOrderId** ()
Exchange [Order](#) Id.
- string **getSymbol** ()
[Instrument](#) name.
- Side **getOrderMode** ()
[Order](#) Mode.
- int **getQuantity** ()
[Order](#) Quantity.
- int **getDisclosedQuantity** ()
- int **getFilledQuantity** ()
Filled quantity.
- int **getOldQuantity** ()
- int **getPrice** ()
[Order](#) Price.
- int **getStopPrice** ()
Stop Price.
- InstrumentType **getSecurityType** ()
[Instrument](#) Type.
- TimeInForce **getOrderValidity** ()
Time in force.
- OrderType **getOrderType** ()
[Order](#) Type.
- OrderStatus **getOrderStatus** ()
[Order](#) Status.

- int **getExchangeEntryTime** ()
- int **getExchangeModifyTime** ()
- void **setClientId** (int val)
*Set Client **Order** Id.*
- void **setTransactionType** (TransactionType val)
Set Transaction Type.
- void **setClOrdId** (long val)
- void **setOrigClOrdId** (long val)
- void **setExchangeOrderId** (string val)
- void **setSymbol** (string val)
Set Symbol.
- void **setOrderMode** (Side val)
*Set **Order** Mode.*
- void **setQuantity** (int val)
*Set **Order** Quantity.*
- void **setDisclosedQuantity** (int val)
- void **setFilledQuantity** (int val)
- void **setOldQuantity** (int val)
- void **setPrice** (int val)
*Set **Order** Price.*
- void **setStopPrice** (int val)
- void **setSecurityType** (InstrumentType val)
- void **setOrderValidity** (TimeInForce val)
- void **setOrderType** (OrderType val)
*Set **Order** Type.*
- void **setOrderStatus** (OrderStatus val)
*Set **Order** Status.*
- void **setExchangeEntryTime** (int val)
- void **setExchangeModifyTime** (int val)
- void **initialize** ()
- void **dumpOrder** ()

4.10.1 Detailed Description

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

4.10.2 Member Function Documentation

4.10.2.1 long Order::getClOrdId ()

Client order Id.

Returns:

Client order Id.

4.10.2.2 string Order::getExchangeOrderId ()

Exchange [Order](#) Id.

Returns:

Exchange order Id.

4.10.2.3 int Order::getFilledQuantity ()

Filled quantity.

Returns:

Filled qty.

4.10.2.4 Side Order::getOrderMode ()

[Order](#) Mode.

Returns:

[OrderMode](#)

4.10.2.5 OrderStatus Order::getOrderStatus ()

[Order](#) Status.

Returns:

[OrderStatus](#)

4.10.2.6 OrderType Order::getOrderType ()

[Order](#) Type.

Returns:

OrderType_LIMIT/OrderType_MARKET/OrderType_STOP_LIMIT.

4.10.2.7 TimeInForce Order::getOrderValidity ()

Time in force.

Returns:

TimeInForce_DAY/TimeInForce_IOC.

4.10.2.8 int Order::getPrice ()

Order Price.

Returns:

Order Price.

4.10.2.9 int Order::getQuantity ()

Order Quantity.

Returns:

Order quantity.

4.10.2.10 InstrumentType Order::getSecurityType ()

Instrument Type.

Returns:

InstrumentType_STOCK/InstrumentType_FUTURE/ InstrumentType_OPTION.

4.10.2.11 int Order::getStopPrice ()

Stop Price.

Returns:

Stop Price.

4.10.2.12 string Order::getSymbol ()

Instrument name.

Returns:

Instrument name.

4.10.2.13 TransactionType Order::getTransactionType ()

Trasnaction Type.

Returns:

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

4.10.2.14 void Order::setClientId (int *val*)

Set Client [Order](#) Id.

Parameters:

val

4.10.2.15 void Order::setOrderMode (Side *val*)

Set [Order](#) Mode.

Parameters:

val [OrderMode](#) User must set this field while placing New [Order](#).

4.10.2.16 void Order::setOrderStatus (OrderStatus *val*)

Set [Order](#) Status.

Parameters:

val Interanally updated by API.

4.10.2.17 void Order::setOrderType (OrderType *val*)

Set [Order](#) Type.

Parameters:

val User must set this field for transaction type New/Modify.

4.10.2.18 void Order::setPrice (int *val*)

Set [Order](#) Price.

Parameters:

val User must set this field for transaction type New/Modify.

4.10.2.19 void Order::setQuantity (int *val*)

Set [Order](#) Quantity.

Parameters:

val User must set this field for transaction type New/Modify.

4.10.2.20 void Order::setSymbol (string *val*)

Set Symbol.

Parameters:

val [Instrument](#) Name. User must set this field in case of New [Order](#) type.

4.10.2.21 void Order::setTransactionType (TransactionType *val*)

Set Transaction Type.

Parameters:

val User must update this field accordingly. [New/Modify/Cancel]

4.11 OrderBook Class Reference

[OrderBook](#) class.

Public Member Functions

- [Order](#) [getOrder](#) (long *clOrderId*)
Get the order details.
- int [update](#) ([ExecutionReport](#) report, bool reconcileOldOrders)
Updates the trade in the [OrderBook](#).
- int [update](#) ([ExecutionReport](#) report)
- void [insert](#) ([Order](#) order)
Insterts the order in the [OrderBook](#).

4.11.1 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.

4.11.2 Member Function Documentation

4.11.2.1 Order [OrderBook::getOrder](#) (long *clOrderId*)

Get the order details.

Parameters:

clOrderId (client order ID generated by the server)

4.11.2.2 void [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.

4.11.2.3 int [OrderBook::update](#) ([ExecutionReport](#) *report*, bool *reconcileOldOrders*)

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.

4.12 Portfolio Class Reference

[Portfolio](#) class.

Public Member Functions

- void [insert](#) ([Order](#) order)
Insterts the order in the [Portfolio](#).
- [Order](#) [getOrderById](#) (int 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.

4.12.1 Detailed Description

[Portfolio](#) class.

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

4.12.2 Member Function Documentation

4.12.2.1 static [Portfolio](#) [Portfolio::getInstance](#) () [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.

4.12.2.2 [Order](#) [Portfolio::getOrderById](#) (int *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.

4.12.2.3 void Portfolio::insert (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.

4.13 Position Class Reference

[Position](#) class.

Public Member Functions

- [Position](#) ([Instrument](#) instrument)
Position.
- [Position](#) ([Position](#) arg0)
Position class copy constructor.
- void [initialize](#) ()
Initialize class members with default values.
- int [getQuantity](#) ()
Get total quantity for current postion.
- int [getAveragePrice](#) ()
Get Average Price for current postion.
- [Instrument](#) [getInstrument](#) ()
Get Instrument from current postion.
- Side [getOrderMode](#) ()
Get Side of current postion.
- void [setQuantity](#) (int val)
- void [setAveragePrice](#) (int val)
Set Average Price for current postion.
- void [setInstrument](#) ([Instrument](#) val)
Set Instrument from current postion.
- void [setOrderType](#) (Side val)
Set Side of current postion.

4.13.1 Detailed Description

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

Note:

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

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Position::Position (Instrument *instrument*)

[Position](#).

Parameters:

\c [Instrument](#) for which we will keep track of postion.

4.13.2.2 Position::Position (Position *arg0*)

[Position](#) class copy constructor.

Parameters:

\c [Position](#) object to copy.

4.14 sampleApplication::Program Class Reference

4.15 Quote Class Reference

Public Member Functions

- virtual void **Dispose** ()
- **Quote** ([Quote](#) arg0)
- void **setSymbolId** (long val)
- void **setNummberOfTrades** (long val)
- void **setVolume** (long val)
- void **setValue** (long val)
- void **setLastTradePrice** (long val)
- void **setLastTradeQty** (long val)
- void **setOpenPrice** (long val)
- void **setClosePrice** (long val)
- void **setHighPrice** (long val)
- void **setLowPrice** (long val)
- void **setTotalBidQty** (long val)
- void **setTotalAskQty** (long val)
- void **setLowerCktLimit** (long val)
- void **setUpperCktLimit** (long val)
- void **setDepth** (byte val)
- void **setBidPrice** (SWIGTYPE_p_long_long val)
- void **setBidQty** (SWIGTYPE_p_long_long val)
- void **setAskPrice** (SWIGTYPE_p_long_long val)
- void **setAskQty** (SWIGTYPE_p_long_long val)
- long **getSymbolId** ()
- long **getNumberOfTrades** ()
- long **getVolume** ()
- long **getValue** ()
- long **getLastTradePrice** ()
- long **getLastTradeQty** ()
- long **getOpenPrice** ()
- long **getClosePrice** ()
- long **getHighPrice** ()
- long **getLowPrice** ()
- long **getTotalBidQty** ()
- long **getTotalAskQty** ()
- long **getLowerCktLimit** ()
- long **getUpperCktLimit** ()
- byte **getDepth** ()
- SWIGTYPE_p_long_long **getBidPrice** ()
- SWIGTYPE_p_long_long **getBidQty** ()
- SWIGTYPE_p_long_long **getAskPrice** ()
- SWIGTYPE_p_long_long **getAskQty** ()

4.16 Trade Class Reference

Public Member Functions

- virtual void **Dispose** ()
- **Trade** ([Instrument](#) arg0)
- **Trade** ([Trade](#) arg0)
- void **initialize** ()
- [Instrument](#) **getInstrument** ()
- long **getTradeId** ()
- long **getClOrdId** ()
- long **getOrigClOrdId** ()
- long **getExchangeOrderId** ()
- Side **getOrderMode** ()
- int **getFilledQuantity** ()
- int **getFilledPrice** ()
- OrderType **getOrderType** ()
- int **getTradeTime** ()
- void **setInstrument** ([Instrument](#) val)
- void **setTradeId** (long val)
- void **setClOrdId** (long val)
- void **setOrigClOrdId** (long val)
- void **setExchangeOrderId** (long val)
- void **setOrderMode** (Side val)
- void **setFilledQuantity** (int val)
- void **setFilledPrice** (int val)
- void **setOrderType** (OrderType val)
- void **setTradeTime** (int val)

4.17 TradeBook Class Reference

[TradeBook](#) class.

Public Member Functions

- [TradeQue](#) [getTradeQue](#) (long *clOrderId*)
Get List of trades.
- int [update](#) ([ExecutionReport](#) report)
Updates the trade in the [TradeBook](#).

4.17.1 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.

4.17.2 Member Function Documentation

4.17.2.1 TradeQue [TradeBook::getTradeQue](#) (long *clOrderId*)

Get List of trades.

Parameters:

clOrderId (client order ID generated by the server) Returns trades in a queue.

4.17.2.2 int [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.

4.18 TradeQue Class Reference

[TradeQue](#) class.

Public Member Functions

- [TradeQue](#) ()
Creates a [TradeQue](#) of specified size and initialize them with given value.
- [TradeQue](#) (uint size, [Trade](#) value)
Creates a [TradeQue](#) of specified size and initialize them with given value.
- [TradeQue](#) (uint size)
Creates a [TradeQue](#) of specified size.
- **TradeQue** ([TradeQue](#) arg0)
- void **assign** (uint n, [Trade](#) value)
- void **swap** ([TradeQue](#) x)
- uint [size](#) ()
Get size of [TradeQue](#).
- uint **max_size** ()
- void [resize](#) (uint n, [Trade](#) c)
Resize [TradeQue](#).
- void **resize** (uint n)
- bool [empty](#) ()
Check whether [TradeQue](#) is empty.
- [Trade](#) [front](#) ()
Front element of [TradeQue](#).
- [Trade](#) [back](#) ()
Last element of [TradeQue](#).
- void [push_front](#) ([Trade](#) x)
Push element in the front of [TradeQue](#).
- void [push_back](#) ([Trade](#) x)
Push element in the back of [TradeQue](#).
- void [pop_front](#) ()
Pop element from front of [TradeQue](#).
- void [pop_back](#) ()
Pop element in the back of [TradeQue](#).
- void [clear](#) ()
Clear elements of [TradeQue](#).

- [Trade](#) **getitem** (int i)
Get trade at index i.
- void **setitem** (int i, [Trade](#) x)
- void **delitem** (int i)
Delete item at index i of [TradeQue](#).
- [TradeQue](#) **getslice** (int i, int j)
- void **setslice** (int i, int j, [TradeQue](#) v)
- void **delslice** (int i, int j)

4.18.1 Detailed Description

[TradeQue](#) class. This class stores the list of all trades which is in the tradebook. While fetching trades from tradebook using trade Id user will get this queue.

Index

- Application, 7
 - onExecutionReport, 8
 - onLoadInstrumentEnd, 8
 - onLogin, 8
 - onLogout, 8
 - onTick, 8
- Context, 11
 - enableLogging, 12
 - getInstance, 12
 - getInstrument, 12
 - loadInstrument, 12
 - login, 12
 - logout, 13
 - placeOrder, 13
 - setApplication, 14
 - subscribe, 14
 - unsubscribe, 14
- enableLogging
 - Context, 12
- ExecutionReport, 15
 - getClientId, 16
 - getClOrderId, 16
 - getErrorCode, 17
 - getLastFillPrice, 17
 - getLastFillQuantity, 17
 - getOrderMode, 17
 - getOrderQuantity, 17
 - getOriginalClOrderId, 17
 - getSymbolId, 18
 - getTradeId, 18
- ExecutionResponse, 19
- getAvgPrice
 - MarketData, 24
- getAvgPriceForQty
 - MarketData, 24
- getClientId
 - ExecutionReport, 16
- getClOrderId
 - ExecutionReport, 16
- getClOrdId
 - Order, 30
- getCount
 - MarketData, 24
- getDayClose
 - MarketData, 24
- getDayHigh
 - MarketData, 24
- getDayLow
 - MarketData, 24
- getDayOpen
 - MarketData, 25
- getDepth
 - MarketData, 25
- getErrorCode
 - ExecutionReport, 17
- getExchangeOrderId
 - Order, 30
- getFilledQuantity
 - Order, 30
- getInstance
 - Context, 12
 - Logger, 22
 - Portfolio, 35
- getInstrument
 - Context, 12
 - MarketData, 25
- getLastFillPrice
 - ExecutionReport, 17
- getLastFillQuantity
 - ExecutionReport, 17
- getLastPrice
 - MarketData, 25
- getLastQty
 - MarketData, 25
- getLastTime
 - MarketData, 25
- getOrder
 - OrderBook, 34
- getOrderByTokenId
 - Portfolio, 35
- getOrderMode
 - ExecutionReport, 17
 - Order, 30
- getOrderQuantity
 - ExecutionReport, 17
- getOrderStatus
 - Order, 30

- getOrderType
 - Order, 30
- getOrderValidity
 - Order, 30
- getOriginalClOrderId
 - ExecutionReport, 17
- getPosition
 - NetPositions, 27
- getPrice
 - MarketData, 25
 - Order, 31
- getQty
 - MarketData, 25
- getQtyForAvgPrice
 - MarketData, 25
- getQtyForWorstPrice
 - MarketData, 25
- getQuantity
 - Order, 31
- getRank
 - MarketData, 25
- getSecurityType
 - Order, 31
- getStopPrice
 - Order, 31
- getSymbol
 - Order, 31
- getSymbolId
 - ExecutionReport, 18
- getTotalQty
 - MarketData, 26
- getTradeId
 - ExecutionReport, 18
- getTradeQue
 - TradeBook, 42
- getTransactionType
 - Order, 31
- getWorstPriceForQty
 - MarketData, 26
- hasQty
 - MarketData, 26
- insert
 - OrderBook, 34
 - Portfolio, 35
- Instrument, 20
- loadInstrument
 - Context, 12
- Logger, 22
 - getInstance, 22
 - setLogLevel, 22
- login
 - Context, 12
- logout
 - Context, 13
- MarketData, 23
 - getAvgPrice, 24
 - getAvgPriceForQty, 24
 - getCount, 24
 - getDayClose, 24
 - getDayHigh, 24
 - getDayLow, 24
 - getDayOpen, 25
 - getDepth, 25
 - getInstrument, 25
 - getLastPrice, 25
 - getLastQty, 25
 - getLastTime, 25
 - getPrice, 25
 - getQty, 25
 - getQtyForAvgPrice, 25
 - getQtyForWorstPrice, 25
 - getRank, 25
 - getTotalQty, 26
 - getWorstPriceForQty, 26
 - hasQty, 26
- NetPositions, 27
 - getPosition, 27
 - update, 27
- onExecutionReport
 - Application, 8
 - sampleApplication::ApplicationImpl, 9
- onLoadInstrumentEnd
 - Application, 8
- onLogin
 - Application, 8
 - sampleApplication::ApplicationImpl, 9
- onLogout
 - Application, 8
 - sampleApplication::ApplicationImpl, 10
- onTick
 - Application, 8
 - sampleApplication::ApplicationImpl, 10
- Order, 28
 - getClOrdId, 30
 - getExchangeOrderId, 30
 - getFilledQuantity, 30
 - getOrderMode, 30
 - getOrderStatus, 30
 - getOrderType, 30
 - getOrderValidity, 30
 - getPrice, 31
 - getQuantity, 31

- getSecurityType, [31](#)
- getStopPrice, [31](#)
- getSymbol, [31](#)
- getTransactionType, [31](#)
- setClientId, [32](#)
- setOrderMode, [32](#)
- setOrderStatus, [32](#)
- setOrderType, [32](#)
- setPrice, [32](#)
- setQuantity, [32](#)
- setSymbol, [33](#)
- setTransactionType, [33](#)
- OrderBook, [34](#)
 - getOrder, [34](#)
 - insert, [34](#)
 - update, [34](#)
- placeOrder
 - Context, [13](#)
- Portfolio, [35](#)
 - getInstance, [35](#)
 - getOrderByTokenId, [35](#)
 - insert, [35](#)
- Position, [37](#)
 - Position, [38](#)
- Quote, [40](#)
- sampleApplication::ApplicationImpl, [9](#)
 - onExecutionReport, [9](#)
 - onLogin, [9](#)
 - onLogout, [10](#)
 - onTick, [10](#)
- sampleApplication::Program, [39](#)
- setApplication
 - Context, [14](#)
- setClientId
 - Order, [32](#)
- setLogLevel
 - Logger, [22](#)
- setOrderMode
 - Order, [32](#)
- setOrderStatus
 - Order, [32](#)
- setOrderType
 - Order, [32](#)
- setPrice
 - Order, [32](#)
- setQuantity
 - Order, [32](#)
- setSymbol
 - Order, [33](#)
- setTransactionType
 - Order, [33](#)
- subscribe
 - Context, [14](#)
- Trade, [41](#)
- TradeBook, [42](#)
 - getTradeQueue, [42](#)
 - update, [42](#)
- TradeQueue, [43](#)
- unsubscribe
 - Context, [14](#)
- update
 - NetPositions, [27](#)
 - OrderBook, [34](#)
 - TradeBook, [42](#)