

Reference Manual

Generated by Doxygen 1.6.1

Sun Dec 22 13:35:00 2013

Contents

1	muTrade API documentation	1
1.1	Introduction	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	Application Class Reference	5
3.1.1	Detailed Description	6
3.1.2	Member Function Documentation	6
3.1.2.1	onExecutionReport	6
3.1.2.2	onLoadInstrumentEnd	6
3.1.2.3	onLogin	6
3.1.2.4	onLogout	6
3.1.2.5	onTick	6
3.2	Context Class Reference	7
3.2.1	Detailed Description	8
3.2.2	Member Function Documentation	8
3.2.2.1	enableLogging	8
3.2.2.2	getInstance	8
3.2.2.3	getInstrument	8
3.2.2.4	loadInstrument	8
3.2.2.5	login	8
3.2.2.6	login	8
3.2.2.7	logout	9
3.2.2.8	placeOrder	9
3.2.2.9	setApplication	10
3.2.2.10	subscribe	10
3.2.2.11	unsubscribe	10

3.3	ExecutionReport Class Reference	11
3.3.1	Detailed Description	12
3.3.2	Member Function Documentation	12
3.3.2.1	getClientId	12
3.3.2.2	getClOrderId	13
3.3.2.3	getErrorCode	13
3.3.2.4	getLastFillPrice	13
3.3.2.5	getLastFillQuantity	13
3.3.2.6	getOrderMode	13
3.3.2.7	getOrderQuantity	13
3.3.2.8	getOriginalClOrderId	14
3.3.2.9	getSymbolId	14
3.3.2.10	getTradeId	14
3.4	ExecutionResponse Class Reference	15
3.4.1	Detailed Description	15
3.5	Instrument Class Reference	16
3.5.1	Detailed Description	16
3.6	InstrumentType Class Reference	18
3.6.1	Detailed Description	18
3.7	Logger Class Reference	19
3.7.1	Detailed Description	19
3.7.2	Member Function Documentation	19
3.7.2.1	getInstance	19
3.7.2.2	setLogLevel	19
3.8	LogLevel Class Reference	20
3.8.1	Detailed Description	20
3.9	MarketData Class Reference	21
3.9.1	Detailed Description	22
3.9.2	Member Function Documentation	22
3.9.2.1	getAvgPrice	22
3.9.2.2	getAvgPriceForQty	22
3.9.2.3	getCount	22
3.9.2.4	getDayClose	22
3.9.2.5	getDayHigh	22
3.9.2.6	getDayLow	22
3.9.2.7	getDayOpen	23

3.9.2.8	getDepth	23
3.9.2.9	getLastPrice	23
3.9.2.10	getLastQty	23
3.9.2.11	getLastTime	23
3.9.2.12	getPrice	23
3.9.2.13	getQty	23
3.9.2.14	getQtyForAvgPrice	23
3.9.2.15	getQtyForWorstPrice	23
3.9.2.16	getRank	23
3.9.2.17	getTotalQty	23
3.9.2.18	getWorstPriceForQty	24
3.9.2.19	hasQty	24
3.10	NetPositions Class Reference	25
3.10.1	Detailed Description	25
3.10.2	Member Function Documentation	25
3.10.2.1	getPosition	25
3.10.2.2	update	25
3.11	OptionType Class Reference	26
3.11.1	Detailed Description	26
3.12	Order Class Reference	27
3.12.1	Detailed Description	28
3.12.2	Member Function Documentation	29
3.12.2.1	getClOrdId	29
3.12.2.2	getExchangeOrderId	29
3.12.2.3	getFilledQuantity	29
3.12.2.4	getOrderMode	29
3.12.2.5	getOrderStatus	29
3.12.2.6	getOrderType	29
3.12.2.7	getOrderValidity	30
3.12.2.8	getPrice	30
3.12.2.9	getQuantity	30
3.12.2.10	getSecurityType	30
3.12.2.11	getStopPrice	30
3.12.2.12	getSymbol	30
3.12.2.13	getTransactionType	31
3.12.2.14	setClientId	31

3.12.2.15	setClOrdId	31
3.12.2.16	setOrderMode	31
3.12.2.17	setOrderStatus	31
3.12.2.18	setOrderType	31
3.12.2.19	setPrice	32
3.12.2.20	setQuantity	32
3.12.2.21	setSymbol	32
3.12.2.22	setTransactionType	32
3.13	OrderBook Class Reference	33
3.13.1	Detailed Description	33
3.13.2	Member Function Documentation	33
3.13.2.1	getOrder	33
3.13.2.2	insert	33
3.13.2.3	update	33
3.14	OrderMode Class Reference	34
3.14.1	Detailed Description	34
3.15	OrderStatus Class Reference	35
3.15.1	Detailed Description	35
3.16	OrderType Class Reference	36
3.16.1	Detailed Description	36
3.17	Portfolio Class Reference	37
3.17.1	Detailed Description	37
3.17.2	Member Function Documentation	37
3.17.2.1	getInstance	37
3.17.2.2	getOrderByTokenId	37
3.17.2.3	insert	38
3.18	Position Class Reference	39
3.18.1	Detailed Description	39
3.18.2	Constructor & Destructor Documentation	40
3.18.2.1	Position	40
3.18.2.2	Position	40
3.19	Quote Class Reference	41
3.20	ResponseType Class Reference	43
3.20.1	Detailed Description	43
3.21	Side Class Reference	44
3.21.1	Detailed Description	44

3.22	TimeInForce Class Reference	45
3.22.1	Detailed Description	45
3.23	Trade Class Reference	46
3.24	TradeBook Class Reference	47
3.24.1	Detailed Description	47
3.24.2	Member Function Documentation	47
3.24.2.1	getTradeQue	47
3.24.2.2	update	47
3.25	TradeQue Class Reference	48
3.25.1	Detailed Description	49
3.26	TransactionType Class Reference	50
3.26.1	Detailed Description	50

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 List

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

Application (Abstract Application class, to be overridden by the developer)	5
Context (Context class for the algorithm)	7
ExecutionReport (Execution Report Class)	11
ExecutionResponse (Execution Response)	15
Instrument (Instrument class)	16
InstrumentType (Instrument Type of the trade (Stock/Future/Option))	18
Logger (Singleton Logging class)	19
LogLevel	20
MarketData (MarketData Class)	21
NetPositions (NetPositions class)	25
OptionType (Option Type (Call/Put))	26
Order (Order Class)	27
OrderBook (OrderBook class)	33
OrderMode (Order Mode)	34
OrderStatus (Order Status)	35
OrderType (Order Type)	36
Portfolio (Portfolio class)	37
Position (Position class)	39
Quote	41
ResponseType (Response Type Internally used by API)	43
Side (Side of the trade (Buy/Sell))	44
TimeInForce (Time In Force (DAY/IOC))	45
Trade	46
TradeBook (TradeBook class)	47
TradeQue (TradeQue class)	48
TransactionType (Transaction Type)	50

Chapter 3

Class Documentation

3.1 Application Class Reference

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

Public Member Functions

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

Protected Member Functions

- [Application](#) (long cPtr, boolean cMemoryOwn)

Protected Attributes

- boolean [swigCMemOwn](#)

3.1.1 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.

3.1.2 Member Function Documentation

3.1.2.1 void Application::onExecutionReport (ExecutionReport *report*)

Event called when an execution is received from Server.

Parameters:

3.1.2.2 void Application::onLoadInstrumentEnd (String *instrumentName*, boolean *success*)

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

Parameters:

3.1.2.3 void Application::onLogin (boolean *status*)

Event called when Login message is returned.

Parameters:

3.1.2.4 void Application::onLogout (boolean *status*)

Event called when Logout message is returned.

Parameters:

3.1.2.5 void Application::onTick (MarketData *arg0*)

Event called when a tick is received.

Parameters:

3.2 Context Class Reference

[Context](#) class for the algorithm.

Public Member Functions

- void [login](#) (int userId, String password, String host, short port, boolean 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.
- boolean [placeOrder](#) ([Order](#) order)
Send an order to the muTrade server.
- void [enableLogging](#) ([LogLevel](#) level)
Enable logging of various events.
- void [enableLogging](#) ()
- 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.

3.2.1 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.

3.2.2 Member Function Documentation

3.2.2.1 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

3.2.2.2 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.

3.2.2.3 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.

3.2.2.4 void Context::loadInstrument (String *s*)

Load static data for an instrument from the muTrade server.

3.2.2.5 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

3.2.2.6 void Context::login (int *userId*, String *password*, String *host*, short *port*, boolean *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

3.2.2.7 void Context::logout ()

Logout from the muTrade server.

3.2.2.8 boolean 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.

3.2.2.9 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.

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

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

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

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

3.3 ExecutionReport Class Reference

Execution Report Class.

Public Member Functions

- **ExecutionReport** (String buf)
- 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** ()
- short **getIsOffline** ()
- long **getSequenceNumber** ()
- long [getTradeId](#) ()
Get Trade Id.
- int [getErrorCode](#) ()
Get Error Code.

- int **getReasonText** ()
- short **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** (short isOffline)
- void **setSequenceNumber** (long sequenceNumber)
- void **setTradeId** (long tradeId)
- void **setErrorCode** (int errorCode)
- void **setReasonText** (int reasonText)
- void **setUnknownOrder** (short unknownOrder)
- void **setInstrumentName** (String instrumentName)
- void **dump** ()
- void **dumpCSV** (SWIGTYPE_p_std__ofstream csvFile)
- int **serialize** (String buf)

3.3.1 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.

3.3.2 Member Function Documentation

3.3.2.1 int ExecutionReport::getClientId ()

Get Client Id.

Returns:

User Id.

3.3.2.2 long ExecutionReport::getClOrderId ()

Get Client [Order](#) Id.

Returns:

Client [Order](#) Id.

3.3.2.3 int ExecutionReport::getErrorCode ()

Get Error Code.

Returns:

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

3.3.2.4 int ExecutionReport::getLastFillPrice ()

Get Last Fill Price.

Returns:

Last Fill Price.

3.3.2.5 int ExecutionReport::getLastFillQuantity ()

Get Last Fill Quantity.

Returns:

Filled Quantity.

3.3.2.6 OrderMode ExecutionReport::getOrderMode ()

Get [Order](#) Mode.

Returns:

[OrderMode](#) Buy or sell order.

3.3.2.7 int ExecutionReport::getOrderQuantity ()

Get Order Quantity.

Returns:

Ordered qty.

3.3.2.8 long ExecutionReport::getOriginalClOrderId ()

Get Original Original Id.

Returns:

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

3.3.2.9 long ExecutionReport::getSymbolId ()

Get Symbol Id.

Returns:

Symbol Id.

3.3.2.10 long ExecutionReport::getTradeId ()

Get [Trade](#) Id.

Returns:

[Trade](#) Id.

3.4 ExecutionResponse Class Reference

Execution Response.

Public Member Functions

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

3.4.1 Detailed Description

Execution Response. Internally used by API.

3.5 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.

3.5.1 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

3.6 InstrumentType Class Reference

[Instrument](#) Type of the trade (Stock/Future/Option).

Static Public Attributes

- static final [InstrumentType](#) **InstrumentType_STOCK** = new [InstrumentType](#)("InstrumentType_STOCK")
- static final [InstrumentType](#) **InstrumentType_FUTURE** = new [InstrumentType](#)("InstrumentType_FUTURE")
- static final [InstrumentType](#) **InstrumentType_OPTION** = new [InstrumentType](#)("InstrumentType_OPTION")

3.6.1 Detailed Description

[Instrument](#) Type of the trade (Stock/Future/Option). It can have values `InstrumentType_STOCK`, `InstrumentType_FUTURE` or `InstrumentType_OPTION`.

3.7 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.

3.7.1 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.

3.7.2 Member Function Documentation

3.7.2.1 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.

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

Set Log Level.

Parameters:

level

3.8 LogLevel Class Reference

Static Public Attributes

- static final [LogLevel](#) **INFO** = new [LogLevel](#)("INFO")
- static final [LogLevel](#) **WARN** = new [LogLevel](#)("WARN")
- static final [LogLevel](#) **ERROR_** = new [LogLevel](#)("ERROR_")
- static final [LogLevel](#) **FATAL** = new [LogLevel](#)("FATAL")

3.8.1 Detailed Description

Various Logging levels supported

3.9 MarketData Class Reference

[MarketData](#) Class.

Public Member Functions

- **MarketData** ([Quote](#) arg0)
- **Instrument** [getInstrument](#) ()
- **int** [getLastPrice](#) ()
Last Traded Price of the [Instrument](#).
- **int** [getLastQty](#) ()
Last Traded Quantity of the [Instrument](#).
- **int** [getLastTime](#) ()
Time of last trade.
- **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.
- **boolean** [getCount](#) ([Side](#) side, **int** rank)
get [Order](#) count at Bid/Ask side
- **boolean** [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.

3.9.1 Detailed Description

[MarketData](#) Class.

3.9.2 Member Function Documentation

3.9.2.1 `int` [MarketData::getAvgPrice](#) (`Side` *side*, `int` *qty*)

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

3.9.2.2 `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

3.9.2.3 `boolean` [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.

3.9.2.4 `int` [MarketData::getDayClose](#) ()

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

3.9.2.5 `int` [MarketData::getDayHigh](#) ()

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

3.9.2.6 `int` [MarketData::getDayLow](#) ()

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

3.9.2.7 int MarketData::getDayOpen ()

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

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

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

3.9.2.9 int MarketData::getLastPrice ()

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

3.9.2.10 int MarketData::getLastQty ()

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

3.9.2.11 int MarketData::getLastTime ()

Time of last trade. Time of last trade

3.9.2.12 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

3.9.2.13 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

3.9.2.14 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.

3.9.2.15 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.

3.9.2.16 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

3.9.2.17 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.

3.9.2.18 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

3.9.2.19 boolean 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

3.10 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](#).

3.10.1 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.

3.10.2 Member Function Documentation

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

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

Parameters:

instrument

side (BUY/SELL)

3.10.2.2 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.

3.11 OptionType Class Reference

Option Type (Call/Put).

Static Public Attributes

- static final [OptionType](#) **OptionType_PUT** = new [OptionType](#)("OptionType_PUT")
- static final [OptionType](#) **OptionType_CALL** = new [OptionType](#)("OptionType_CALL")

3.11.1 Detailed Description

Option Type (Call/Put). You can use OptionType_PUT, OptionType_CALL.

3.12 Order Class Reference

[Order](#) Class.

Public Member Functions

- `int getClientId ()`
- `TransactionType getTransactionType ()`
Trasnsaction 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)
Set Symbol.
- 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** ()

3.12.1 Detailed Description

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

3.12.2 Member Function Documentation

3.12.2.1 long Order::getClOrdId ()

Client order Id.

Returns:

Client order Id.

3.12.2.2 String Order::getExchangeOrderId ()

Exchange [Order](#) Id.

Returns:

Exchange order Id.

3.12.2.3 int Order::getFilledQuantity ()

Filled quantity.

Returns:

Filled qty.

3.12.2.4 Side Order::getOrderMode ()

[Order](#) Mode.

Returns:

[OrderMode](#)

3.12.2.5 OrderStatus Order::getOrderStatus ()

[Order](#) Status.

Returns:

[OrderStatus](#)

3.12.2.6 OrderType Order::getOrderType ()

[Order](#) Type.

Returns:

OrderType_LIMIT/OrderType_MARKET/OrderType_STOP_LIMIT.

3.12.2.7 TimeInForce Order::getOrderValidity ()

Time in force.

Returns:

TimeInForce_DAY/TimeInForce_IOC.

3.12.2.8 int Order::getPrice ()

[Order](#) Price.

Returns:

[Order](#) Price.

3.12.2.9 int Order::getQuantity ()

[Order](#) Quantity.

Returns:

[Order](#) quantity.

3.12.2.10 InstrumentType Order::getSecurityType ()

[Instrument](#) Type.

Returns:

InstrumentType_STOCK/InstrumentType_FUTURE/ InstrumentType_OPTION.

3.12.2.11 int Order::getStopPrice ()

Stop Price.

Returns:

Stop Price.

3.12.2.12 String Order::getSymbol ()

[Instrument](#) name.

Returns:

[Instrument](#) name.

3.12.2.13 TransactionType Order::getTransactionType ()

Trasnaction Type.

Returns:

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

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

Set Client [Order](#) Id.

Parameters:

val

3.12.2.15 void Order::setClOrdId (long *val*)

Set Symbol.

Parameters:

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

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

Set [Order](#) Mode.

Parameters:

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

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

Set [Order](#) Status.

Parameters:

val Interanally updated by API.

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

Set [Order](#) Type.

Parameters:

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

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

Set [Order](#) Price.

Parameters:

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

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

Set [Order](#) Quantity.

Parameters:

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

3.12.2.21 void Order::setSymbol (String *val*)

Set Symbol.

Parameters:

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

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

Set Transaction Type.

Parameters:

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

3.13 OrderBook Class Reference

[OrderBook](#) class.

Public Member Functions

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

3.13.1 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.

3.13.2 Member Function Documentation

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

Get the order details.

Parameters:

clOrderId (client order ID generated by the server)

3.13.2.2 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.

3.13.2.3 int [OrderBook::update](#) ([ExecutionReport](#) *report*, boolean *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.

3.14 OrderMode Class Reference

[Order](#) Mode.

Static Public Attributes

- static final [OrderMode](#) **OrderMode_BUY** = new [OrderMode](#)("OrderMode_BUY", 0)
- static final [OrderMode](#) **OrderMode_SELL** = new [OrderMode](#)("OrderMode_SELL", 1)
- static final [OrderMode](#) **OrderMode_MAX** = new [OrderMode](#)("OrderMode_MAX")

3.14.1 Detailed Description

[Order](#) Mode.

3.15 OrderStatus Class Reference

[Order](#) Status.

Static Public Attributes

- static final [OrderStatus](#) **OrderStatus_PENDING** = new [OrderStatus](#)("OrderStatus_PENDING")
- static final [OrderStatus](#) **OrderStatus_CONFIRMED** = new [OrderStatus](#)("OrderStatus_CONFIRMED")
- static final [OrderStatus](#) **OrderStatus_FILLED** = new [OrderStatus](#)("OrderStatus_FILLED")
- static final [OrderStatus](#) **OrderStatus_CANCELED** = new [OrderStatus](#)("OrderStatus_CANCELED")
- static final [OrderStatus](#) **OrderStatus_REPLACED** = new [OrderStatus](#)("OrderStatus_REPLACED")
- static final [OrderStatus](#) **OrderStatus_NEW_REJECTED** = new [OrderStatus](#)("OrderStatus_NEW_REJECTED")
- static final [OrderStatus](#) **OrderStatus_CANCEL_REJECTED** = new [OrderStatus](#)("OrderStatus_CANCEL_REJECTED")
- static final [OrderStatus](#) **OrderStatus_REPLACE_REJECTED** = new [OrderStatus](#)("OrderStatus_REPLACE_REJECTED")
- static final [OrderStatus](#) **OrderStatus_FROZEN** = new [OrderStatus](#)("OrderStatus_FROZEN")
- static final [OrderStatus](#) **OrderStatus_MARKET_TO_LIMIT** = new [OrderStatus](#)("OrderStatus_MARKET_TO_LIMIT")
- static final [OrderStatus](#) **OrderStatus_TRIGGERED** = new [OrderStatus](#)("OrderStatus_TRIGGERED")
- static final [OrderStatus](#) **OrderStatus_PARTIALLY_FILLED** = new [OrderStatus](#)("OrderStatus_PARTIALLY_FILLED")
- static final [OrderStatus](#) **OrderStatus_CANCELED_OF_IOC** = new [OrderStatus](#)("OrderStatus_CANCELED_OF_IOC")
- static final [OrderStatus](#) **OrderStatus_RMS_REJECT** = new [OrderStatus](#)("OrderStatus_RMS_REJECT")
- static final [OrderStatus](#) **OrderStatus_MAX** = new [OrderStatus](#)("OrderStatus_MAX")

3.15.1 Detailed Description

[Order](#) Status.

3.16 OrderType Class Reference

[Order](#) Type.

Static Public Attributes

- static final [OrderType](#) **OrderType_LIMIT** = new [OrderType](#)("OrderType_LIMIT")
- static final [OrderType](#) **OrderType_MARKET** = new [OrderType](#)("OrderType_MARKET")
- static final [OrderType](#) **OrderType_STOP_LIMIT** = new [OrderType](#)("OrderType_STOP_LIMIT")
- static final [OrderType](#) **OrderType_STOP** = new [OrderType](#)("OrderType_STOP")
- static final [OrderType](#) **OrderType_MAX** = new [OrderType](#)("OrderType_MAX")

3.16.1 Detailed Description

[Order](#) Type. You can use OrderType_LIMIT, OrderType_MARKET, OrderType_STOP_LIMIT and OrderType_STOP, while placing oredr.

3.17 Portfolio Class Reference

[Portfolio](#) class.

Public Member Functions

- void [insert](#) ([Order](#) order)
Inserts 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.

3.17.1 Detailed Description

[Portfolio](#) class.

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

3.17.2 Member Function Documentation

3.17.2.1 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.

3.17.2.2 [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.

3.17.2.3 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.

3.18 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.

3.18.1 Detailed Description

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

Note:

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

3.18.2 Constructor & Destructor Documentation

3.18.2.1 Position::Position (Instrument *instrument*)

[Position](#).

Parameters:

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

3.18.2.2 Position::Position (Position *arg0*)

[Position](#) class copy constructor.

Parameters:

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

3.19 Quote Class Reference

Public Member Functions

- synchronized void **delete** ()
- **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** (short 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** ()
- short **getDepth** ()
- SWIGTYPE_p_long_long **getBidPrice** ()
- SWIGTYPE_p_long_long **getBidQty** ()
- SWIGTYPE_p_long_long **getAskPrice** ()
- SWIGTYPE_p_long_long **getAskQty** ()

Protected Member Functions

- **Quote** (long cPtr, boolean cMemoryOwn)
- void **finalize** ()

Static Protected Member Functions

- static long **getCPtr** ([Quote](#) obj)

Protected Attributes

- boolean **swigCMemOwn**

3.20 ResponseType Class Reference

Response Type Internally used by API.

Static Public Attributes

- static final [ResponseType](#) **ResponseType_SUCCESS** = new [ResponseType](#)("ResponseType_SUCCESS")
- static final [ResponseType](#) **ResponseType_FAILURE** = new [ResponseType](#)("ResponseType_FAILURE")
- static final [ResponseType](#) **ResponseType_TERMINATE_SUCCESS** = new [ResponseType](#)("ResponseType_TERMINATE_SUCCESS")
- static final [ResponseType](#) **ResponseType_TERMINATE_FAILURE** = new [ResponseType](#)("ResponseType_TERMINATE_FAILURE")
- static final [ResponseType](#) **ResponseType_TERMINATE_SQUAREOFF_SUCCESS** = new [ResponseType](#)("ResponseType_TERMINATE_SQUAREOFF_SUCCESS")
- static final [ResponseType](#) **ResponseType_TERMINATE_SQUAREOFF_FAILURE** = new [ResponseType](#)("ResponseType_TERMINATE_SQUAREOFF_FAILURE")
- static final [ResponseType](#) **ResponseType_PAUSE_STRATEGY_SUCCESS** = new [ResponseType](#)("ResponseType_PAUSE_STRATEGY_SUCCESS")
- static final [ResponseType](#) **ResponseType_PAUSE_STRATEGY_FAILURE** = new [ResponseType](#)("ResponseType_PAUSE_STRATEGY_FAILURE")
- static final [ResponseType](#) **ResponseType_STRATEGY_RUNNING** = new [ResponseType](#)("ResponseType_STRATEGY_RUNNING")
- static final [ResponseType](#) **ResponseType_TERMINATE_SPREAD_BREACHED** = new [ResponseType](#)("ResponseType_TERMINATE_SPREAD_BREACHED")
- static final [ResponseType](#) **ResponseType_ALGO_NOT_ALLOWED** = new [ResponseType](#)("ResponseType_ALGO_NOT_ALLOWED")
- static final [ResponseType](#) **ResponseType_MAX** = new [ResponseType](#)("ResponseType_MAX")

3.20.1 Detailed Description

Response Type Internally used by API.

3.21 Side Class Reference

[Side](#) of the trade (Buy/Sell).

Static Public Attributes

- static final [Side](#) **Side_BUY** = new [Side](#)("Side_BUY", 0)
- static final [Side](#) **Side_SELL** = new [Side](#)("Side_SELL", 1)
- static final [Side](#) **Side_BID** = new [Side](#)("Side_BID", 0)
- static final [Side](#) **Side_ASK** = new [Side](#)("Side_ASK", 1)

3.21.1 Detailed Description

[Side](#) of the trade (Buy/Sell). You can use Side_BUY, Side_BID interchangeably.

3.22 TimeInForce Class Reference

Time In Force (DAY/IOC).

Static Public Attributes

- static final [TimeInForce](#) **TimeInForce_DAY** = new [TimeInForce](#)("TimeInForce_DAY")
- static final [TimeInForce](#) **TimeInForce_IOC** = new [TimeInForce](#)("TimeInForce_IOC")
- static final [TimeInForce](#) **TimeInForce_MAX** = new [TimeInForce](#)("TimeInForce_MAX")

3.22.1 Detailed Description

Time In Force (DAY/IOC). You can use TimeInForce_DAY, TimeInForce_IOC interchangeably.

3.23 Trade Class Reference

Public Member Functions

- synchronized void **delete** ()
- **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)

Protected Member Functions

- **Trade** (long cPtr, boolean cMemoryOwn)
- void **finalize** ()

Static Protected Member Functions

- static long **getCPtr** ([Trade](#) obj)

Protected Attributes

- boolean **swigCMemOwn**

3.24 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](#).

3.24.1 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.

3.24.2 Member Function Documentation

3.24.2.1 TradeQue TradeBook::getTradeQue (long *clOrderId*)

Get List of trades.

Parameters:

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

3.24.2.2 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.

3.25 TradeQue Class Reference

[TradeQue](#) class.

Public Member Functions

- [TradeQue](#) (long size, [Trade](#) value)
Creates a [TradeQue](#) of specified size and initialize them with given value.
- [TradeQue](#) (long size)
Creates a [TradeQue](#) of specified size.
- **TradeQue** ([TradeQue](#) arg0)
- void **assign** (long n, [Trade](#) value)
- void **swap** ([TradeQue](#) x)
- long **size** ()
Get size of [TradeQue](#).
- long **max_size** ()
- void **resize** (long n, [Trade](#) c)
- void **resize** (long n)
Resize [TradeQue](#).
- boolean **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)

3.25.1 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.

3.26 TransactionType Class Reference

Transaction Type.

Static Public Attributes

- static final [TransactionType](#) **TransactionType_NEW** = new [TransactionType](#)("TransactionType_NEW")
- static final [TransactionType](#) **TransactionType_MODIFY** = new [TransactionType](#)("TransactionType_MODIFY")
- static final [TransactionType](#) **TransactionType_CANCEL** = new [TransactionType](#)("TransactionType_CANCEL")
- static final [TransactionType](#) **TransactionType_MAX** = new [TransactionType](#)("TransactionType_MAX")

3.26.1 Detailed Description

Transaction Type. You can use TransactionType_NEW, TransactionType_MODIFY and TransactionType_CANCEL while placing order.

Index

- Application, [5](#)
 - onExecutionReport, [6](#)
 - onLoadInstrumentEnd, [6](#)
 - onLogin, [6](#)
 - onLogout, [6](#)
 - onTick, [6](#)
- Context, [7](#)
 - enableLogging, [8](#)
 - getInstance, [8](#)
 - getInstrument, [8](#)
 - loadInstrument, [8](#)
 - login, [8](#)
 - logout, [9](#)
 - placeOrder, [9](#)
 - setApplication, [10](#)
 - subscribe, [10](#)
 - unsubscribe, [10](#)
- enableLogging
 - Context, [8](#)
- ExecutionReport, [11](#)
 - getClientId, [12](#)
 - getClOrderId, [12](#)
 - getErrorCode, [13](#)
 - getLastFillPrice, [13](#)
 - getLastFillQuantity, [13](#)
 - getOrderMode, [13](#)
 - getOrderQuantity, [13](#)
 - getOriginalClOrderId, [13](#)
 - getSymbolId, [14](#)
 - getTradeId, [14](#)
- ExecutionResponse, [15](#)
- getAvgPrice
 - MarketData, [22](#)
- getAvgPriceForQty
 - MarketData, [22](#)
- getClientId
 - ExecutionReport, [12](#)
- getClOrderId
 - ExecutionReport, [12](#)
- getClOrdId
 - Order, [29](#)
- getCount
 - MarketData, [22](#)
- getDayClose
 - MarketData, [22](#)
- getDayHigh
 - MarketData, [22](#)
- getDayLow
 - MarketData, [22](#)
- getDayOpen
 - MarketData, [22](#)
- getDepth
 - MarketData, [23](#)
- getErrorCode
 - ExecutionReport, [13](#)
- getExchangeOrderId
 - Order, [29](#)
- getFilledQuantity
 - Order, [29](#)
- getInstance
 - Context, [8](#)
 - Logger, [19](#)
 - Portfolio, [37](#)
- getInstrument
 - Context, [8](#)
- getLastFillPrice
 - ExecutionReport, [13](#)
- getLastFillQuantity
 - ExecutionReport, [13](#)
- getLastPrice
 - MarketData, [23](#)
- getLastQty
 - MarketData, [23](#)
- getLastTime
 - MarketData, [23](#)
- getOrder
 - OrderBook, [33](#)
- getOrderByTokenId
 - Portfolio, [37](#)
- getOrderMode
 - ExecutionReport, [13](#)
 - Order, [29](#)
- getOrderQuantity
 - ExecutionReport, [13](#)
- getOrderStatus
 - Order, [29](#)
- getOrderType

- Order, 29
- getOrderValidity
 - Order, 29
- getOriginalCfOrderId
 - ExecutionReport, 13
- getPosition
 - NetPositions, 25
- getPrice
 - MarketData, 23
 - Order, 30
- getQty
 - MarketData, 23
- getQtyForAvgPrice
 - MarketData, 23
- getQtyForWorstPrice
 - MarketData, 23
- getQuantity
 - Order, 30
- getRank
 - MarketData, 23
- getSecurityType
 - Order, 30
- getStopPrice
 - Order, 30
- getSymbol
 - Order, 30
- getSymbolId
 - ExecutionReport, 14
- getTotalQty
 - MarketData, 23
- getTradeId
 - ExecutionReport, 14
- getTradeQue
 - TradeBook, 47
- getTransactionType
 - Order, 30
- getWorstPriceForQty
 - MarketData, 23
- hasQty
 - MarketData, 24
- insert
 - OrderBook, 33
 - Portfolio, 37
- Instrument, 16
- InstrumentType, 18
- loadInstrument
 - Context, 8
- Logger, 19
 - getInstance, 19
 - setLogLevel, 19
- login
 - Context, 8
- LogLevel, 20
- logout
 - Context, 9
- MarketData, 21
 - getAvgPrice, 22
 - getAvgPriceForQty, 22
 - getCount, 22
 - getDayClose, 22
 - getDayHigh, 22
 - getDayLow, 22
 - getDayOpen, 22
 - getDepth, 23
 - getLastPrice, 23
 - getLastQty, 23
 - getLastTime, 23
 - getPrice, 23
 - getQty, 23
 - getQtyForAvgPrice, 23
 - getQtyForWorstPrice, 23
 - getRank, 23
 - getTotalQty, 23
 - getWorstPriceForQty, 23
 - hasQty, 24
- NetPositions, 25
 - getPosition, 25
 - update, 25
- onExecutionReport
 - Application, 6
- onLoadInstrumentEnd
 - Application, 6
- onLogin
 - Application, 6
- onLogout
 - Application, 6
- onTick
 - Application, 6
- OptionType, 26
- Order, 27
 - getCfOrderId, 29
 - getExchangeOrderId, 29
 - getFilledQuantity, 29
 - getOrderMode, 29
 - getOrderStatus, 29
 - getOrderType, 29
 - getOrderValidity, 29
 - getPrice, 30
 - getQuantity, 30
 - getSecurityType, 30
 - getStopPrice, 30
 - getSymbol, 30

- getTransactionType, [30](#)
- setClientId, [31](#)
- setClOrdId, [31](#)
- setOrderMode, [31](#)
- setOrderStatus, [31](#)
- setOrderType, [31](#)
- setPrice, [31](#)
- setQuantity, [32](#)
- setSymbol, [32](#)
- setTransactionType, [32](#)
- OrderBook, [33](#)
 - getOrder, [33](#)
 - insert, [33](#)
 - update, [33](#)
- OrderMode, [34](#)
- OrderStatus, [35](#)
- OrderType, [36](#)
- placeOrder
 - Context, [9](#)
- Portfolio, [37](#)
 - getInstance, [37](#)
 - getOrderByTokenId, [37](#)
 - insert, [37](#)
- Position, [39](#)
 - Position, [40](#)
- Quote, [41](#)
- ResponseType, [43](#)
- setApplication
 - Context, [10](#)
- setClientId
 - Order, [31](#)
- setClOrdId
 - Order, [31](#)
- setLogLevel
 - Logger, [19](#)
- setOrderMode
 - Order, [31](#)
- setOrderStatus
 - Order, [31](#)
- setOrderType
 - Order, [31](#)
- setPrice
 - Order, [31](#)
- setQuantity
 - Order, [32](#)
- setSymbol
 - Order, [32](#)
- setTransactionType
 - Order, [32](#)
- Side, [44](#)
- subscribe
 - Context, [10](#)
- TimeInForce, [45](#)
- Trade, [46](#)
- TradeBook, [47](#)
 - getTradeQueue, [47](#)
 - update, [47](#)
- TradeQueue, [48](#)
- TransactionType, [50](#)
- unsubscribe
 - Context, [10](#)
- update
 - NetPositions, [25](#)
 - OrderBook, [33](#)
 - TradeBook, [47](#)