

Dispatcher

Generated by Doxygen 1.8.1.2

Thu Jan 30 2014 17:06:49

Contents

1	muTrade Market Data Disptacher documentation	1
1.1	Introduction	1
2	Namespace Index	3
2.1	Namespace List	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	CMD Namespace Reference	9
5.1.1	Enumeration Type Documentation	10
5.1.1.1	CommandCategory	10
5.1.1.2	Exchangeld	11
5.2	MarketData Namespace Reference	12
5.2.1	Typedef Documentation	12
5.2.1.1	ConsolidatedIdFdMap	12
5.2.1.2	FdIdMap	12
5.2.1.3	IdFdMap	12
5.2.1.4	IdFdMapltr	12
5.2.2	Enumeration Type Documentation	13
5.2.2.1	FeedType	13
5.2.3	Function Documentation	13
5.2.3.1	findIndex	13
5.2.3.2	getCurrentLocalTimeStamp	13
6	Class Documentation	15
6.1	MarketData::FdDispatcher Class Reference	15
6.1.1	Detailed Description	18
6.1.2	Constructor & Destructor Documentation	18

6.1.2.1	FdDispatcher	18
6.1.2.2	~FdDispatcher	18
6.1.3	Member Function Documentation	18
6.1.3.1	createFifo	18
6.1.3.2	dispatchFd	19
6.1.3.3	dispatchFd	20
6.1.3.4	dispatchOnConsolidatedFd	20
6.1.3.5	fifoReadMcl	22
6.1.3.6	fifoReadTbt	23
6.1.3.7	getConsolidatedDepthAsk	24
6.1.3.8	getConsolidatedDepthBid	25
6.1.3.9	getMclUseDispatcher	25
6.1.3.10	getUseDispatcherCM	25
6.1.3.11	getUseDispatcherFO	26
6.1.3.12	invokeFifoReadMcl	26
6.1.3.13	invokeFifoReadTbt	26
6.1.3.14	openFifo	27
6.1.3.15	operator()	27
6.1.3.16	setMclUseDispatcher	28
6.1.3.17	setUseDispatcherCM	28
6.1.3.18	setUseDispatcherFO	29
6.1.3.19	writeOnFd	29
6.1.3.20	writeOnFd	30
6.1.4	Member Data Documentation	31
6.1.4.1	_mclUseDispatcher	31
6.1.4.2	_numberOfDispatcherThread	31
6.1.4.3	_threads	31
6.1.4.4	_useDispatcherCM	31
6.1.4.5	_useDispatcherFO	31
6.2	MarketData::FdState Struct Reference	31
6.2.1	Member Data Documentation	32
6.2.1.1	buffer	32
6.2.1.2	buffer_used	32
6.2.1.3	readEvent	32
6.3	CMD::InstrumentRequest Class Reference	32
6.3.1	Detailed Description	33
6.3.2	Constructor & Destructor Documentation	33
6.3.2.1	InstrumentRequest	33
6.3.2.2	InstrumentRequest	33
6.3.3	Member Function Documentation	33

6.3.3.1	getMnemonic	34
6.3.3.2	initialize	34
6.3.3.3	serialize	34
6.3.3.4	setMnemonic	34
6.3.4	Member Data Documentation	34
6.3.4.1	_mnemonic	34
6.3.4.2	MNEMONIC_LENGTH	34
6.4	CMD::MDQuote Class Reference	34
6.4.1	Detailed Description	37
6.4.2	Constructor & Destructor Documentation	37
6.4.2.1	MDQuote	37
6.4.2.2	MDQuote	37
6.4.3	Member Function Documentation	37
6.4.3.1	dump	37
6.4.3.2	dumpDepth	37
6.4.3.3	getAskPrice	37
6.4.3.4	getAskQty	37
6.4.3.5	getBidPrice	37
6.4.3.6	getBidQty	37
6.4.3.7	getClosePrice	37
6.4.3.8	getDepth	37
6.4.3.9	getHighPrice	37
6.4.3.10	getLastTradePrice	37
6.4.3.11	getLastTradeQty	37
6.4.3.12	getLowerCktLimit	37
6.4.3.13	getLowPrice	38
6.4.3.14	getNumberOfTrades	38
6.4.3.15	getOpenPrice	38
6.4.3.16	getSymbolId	38
6.4.3.17	getTotalAskQty	38
6.4.3.18	getTotalBidQty	38
6.4.3.19	getUpperCktLimit	38
6.4.3.20	getValue	38
6.4.3.21	getVolume	38
6.4.3.22	initialize	38
6.4.3.23	serialize	38
6.4.3.24	setAskPrice	38
6.4.3.25	setAskQty	38
6.4.3.26	setBidPrice	38
6.4.3.27	setBidQty	39

6.4.3.28	setClosePrice	39
6.4.3.29	setDepth	39
6.4.3.30	setHighPrice	39
6.4.3.31	setLastTradePrice	39
6.4.3.32	setLastTradeQty	39
6.4.3.33	setLowerCktLimit	39
6.4.3.34	setLowPrice	39
6.4.3.35	setNummberOfTrades	39
6.4.3.36	setOpenPrice	39
6.4.3.37	setSymbolId	39
6.4.3.38	setTotalAskQty	39
6.4.3.39	setTotalBidQty	39
6.4.3.40	setUpperCktLimit	39
6.4.3.41	setValue	39
6.4.3.42	setVolume	39
6.4.4	Member Data Documentation	39
6.4.4.1	_askPrice	39
6.4.4.2	_askQty	39
6.4.4.3	_bidPrice	39
6.4.4.4	_bidQty	39
6.4.4.5	_closePrice	39
6.4.4.6	_depth	39
6.4.4.7	_highPrice	39
6.4.4.8	_lastTradePrice	39
6.4.4.9	_lastTradeQty	39
6.4.4.10	_lowerCktLimit	39
6.4.4.11	_lowPrice	39
6.4.4.12	_numberOfTrades	40
6.4.4.13	_openPrice	40
6.4.4.14	_symbolId	40
6.4.4.15	_totalAskQty	40
6.4.4.16	_totalBidQty	40
6.4.4.17	_upperCktLimit	40
6.4.4.18	_value	40
6.4.4.19	_volume	40
6.4.4.20	MAX_LOOKUP_LEVEL	40
6.4.4.21	QUOTEDATA_LENGTH	40
6.5	CMD::MDQuoteConsolidated Class Reference	40
6.5.1	Detailed Description	42
6.5.2	Constructor & Destructor Documentation	43

6.5.2.1	MDQuoteConsolidated	43
6.5.2.2	MDQuoteConsolidated	43
6.5.3	Member Function Documentation	43
6.5.3.1	dump	43
6.5.3.2	dumpDepth	43
6.5.3.3	getAskPrice	43
6.5.3.4	getAskQty	43
6.5.3.5	getBidPrice	43
6.5.3.6	getBidQty	43
6.5.3.7	getClosePrice	43
6.5.3.8	getDepth	43
6.5.3.9	getHighPrice	43
6.5.3.10	getLastTradePrice	43
6.5.3.11	getLastTradeQty	43
6.5.3.12	getLowPrice	43
6.5.3.13	getNumberOfSymbols	43
6.5.3.14	getOpenPrice	43
6.5.3.15	getSymbolId	43
6.5.3.16	initialize	43
6.5.3.17	serialize	44
6.5.3.18	setAskPrice	44
6.5.3.19	setAskQty	44
6.5.3.20	setBidPrice	44
6.5.3.21	setBidQty	45
6.5.3.22	setClosePrice	45
6.5.3.23	setDepth	45
6.5.3.24	setHighPrice	45
6.5.3.25	setLastTradePrice	46
6.5.3.26	setLastTradeQty	46
6.5.3.27	setLowPrice	46
6.5.3.28	setNumberOfSymbols	46
6.5.3.29	setOpenPrice	47
6.5.3.30	setSymbolId	47
6.5.4	Member Data Documentation	47
6.5.4.1	_askPrice	47
6.5.4.2	_askQty	47
6.5.4.3	_bidPrice	47
6.5.4.4	_bidQty	47
6.5.4.5	_closePrice	47
6.5.4.6	_depth	47

6.5.4.7	_highPrice	47
6.5.4.8	_lastTradePrice	47
6.5.4.9	_lastTradeQty	47
6.5.4.10	_lowPrice	47
6.5.4.11	_numSymbols	47
6.5.4.12	_openPrice	47
6.5.4.13	_symbolId	47
6.5.4.14	MAX_LOOKUP_LEVEL	48
6.5.4.15	QUOTEDATA_LENGTH	48
6.6	MarketData::MDStorage Class Reference	48
6.6.1	Detailed Description	49
6.6.2	Member Typedef Documentation	49
6.6.2.1	SymbolQuoteMap	49
6.6.3	Constructor & Destructor Documentation	49
6.6.3.1	MDStorage	49
6.6.3.2	MDStorage	49
6.6.4	Member Function Documentation	49
6.6.4.1	get	49
6.6.4.2	getInstance	50
6.6.4.3	quoteToMDQuote	50
6.6.4.4	save	50
6.6.5	Member Data Documentation	51
6.6.5.1	_depth	51
6.6.5.2	_symbolQuoteMap	51
6.6.5.3	MAX_ORDER_DEPTH	51
6.7	CMD::MDSubscribeRequest Class Reference	51
6.7.1	Detailed Description	53
6.7.2	Constructor & Destructor Documentation	53
6.7.2.1	MDSubscribeRequest	53
6.7.2.2	MDSubscribeRequest	53
6.7.3	Member Function Documentation	53
6.7.3.1	getSymbolId	53
6.7.3.2	getUserId	53
6.7.3.3	initialize	53
6.7.3.4	isSubscriptionReq	53
6.7.3.5	serialize	53
6.7.3.6	setSubscriptionReq	53
6.7.3.7	setSymbolId	53
6.7.3.8	setUserId	53
6.7.4	Member Data Documentation	53

6.7.4.1	_isSubscription	53
6.7.4.2	_symbolId	53
6.7.4.3	_userId	53
6.7.4.4	TOKEN_LENGTH	53
6.8	CMD::ScripMasterDataRequest Class Reference	53
6.8.1	Constructor & Destructor Documentation	55
6.8.1.1	ScripMasterDataRequest	55
6.8.1.2	ScripMasterDataRequest	55
6.8.2	Member Function Documentation	55
6.8.2.1	dump	55
6.8.2.2	getClientId	55
6.8.2.3	getExchangeId	56
6.8.2.4	getExpiryDate	56
6.8.2.5	getExpiryYearMon	56
6.8.2.6	getMarketName	56
6.8.2.7	getNumberOfRecords	56
6.8.2.8	getOptionMode	56
6.8.2.9	getOptionType	56
6.8.2.10	getRecordNumber	56
6.8.2.11	getScripMasterDataRequestType	56
6.8.2.12	getSecurityId	56
6.8.2.13	getSecurityType	56
6.8.2.14	getSeries	56
6.8.2.15	getStrikePrice	56
6.8.2.16	getSymbol	56
6.8.2.17	getSymbolAlias	56
6.8.2.18	getSymbolId	56
6.8.2.19	serialize	56
6.8.2.20	setClientId	56
6.8.2.21	setExchangeId	56
6.8.2.22	setExpiryDate	56
6.8.2.23	setExpiryYearMon	56
6.8.2.24	setMarketName	56
6.8.2.25	setNumberOfRecords	56
6.8.2.26	setOptionMode	56
6.8.2.27	setOptionType	56
6.8.2.28	setRecordNumber	56
6.8.2.29	setScripMasterDataRequestType	56
6.8.2.30	setSecurityId	57
6.8.2.31	setSecurityType	57

6.8.2.32	setSeries	57
6.8.2.33	setStrikePrice	57
6.8.2.34	setSymbol	57
6.8.2.35	setSymbolAlias	57
6.8.2.36	setSymbolId	57
6.8.3	Member Data Documentation	57
6.8.3.1	_clientId	57
6.8.3.2	_exchangeId	57
6.8.3.3	_expiryDate	57
6.8.3.4	_expiryYearMon	57
6.8.3.5	_marketName	57
6.8.3.6	_numberOfRecords	57
6.8.3.7	_optionMode	57
6.8.3.8	_optionType	57
6.8.3.9	_recordNumber	57
6.8.3.10	_scripMasterDataRequestType	57
6.8.3.11	_securityId	57
6.8.3.12	_securityType	57
6.8.3.13	_series	57
6.8.3.14	_strikePrice	57
6.8.3.15	_symbol	57
6.8.3.16	_symbolAlias	57
6.8.3.17	_symbolId	57
6.9	Subscription Class Reference	58
6.9.1	Detailed Description	58
6.10	MarketData::SubscriptionServer Class Reference	58
6.10.1	Constructor & Destructor Documentation	60
6.10.1.1	SubscriptionServer	60
6.10.2	Member Function Documentation	60
6.10.2.1	acceptHandle	60
6.10.2.2	allocFdState	60
6.10.2.3	freeFdState	61
6.10.2.4	invokeAcceptHandler	61
6.10.2.5	invokeReadHandler	62
6.10.2.6	operator()	62
6.10.2.7	readHandle	63
6.10.3	Member Data Documentation	63
6.10.3.1	_pendingBuffer	63
6.10.3.2	_pendingBufferSize	63
6.10.3.3	_port	64

6.11 MarketData::SubscriptionServMsg Class Reference	64
6.11.1 Detailed Description	65
6.11.2 Member Typedef Documentation	66
6.11.2.1 FeedTypeMap	66
6.11.2.2 SymbolIdMap	66
6.11.3 Constructor & Destructor Documentation	66
6.11.3.1 SubscriptionServMsg	66
6.11.3.2 SubscriptionServMsg	66
6.11.4 Member Function Documentation	66
6.11.4.1 dump	66
6.11.4.2 getFdName	68
6.11.4.3 getFeedType	68
6.11.4.4 getIsConsolidatedFeed	68
6.11.4.5 getIsRunningLocally	69
6.11.4.6 getSymbolCount	69
6.11.4.7 getSymbolId	69
6.11.4.8 isSubscriptionReq	70
6.11.4.9 isUnlinkFifo	70
6.11.4.10 serialize	70
6.11.4.11 setFdName	71
6.11.4.12 setFeedType	71
6.11.4.13 setIsConsolidatedFeed	72
6.11.4.14 setIsRunningLocally	72
6.11.4.15 setSubscriptionReq	72
6.11.4.16 setSymbolCount	73
6.11.4.17 setSymbolId	73
6.11.4.18 setUnlinkFifo	73
6.11.5 Member Data Documentation	73
6.11.5.1 _fdName	73
6.11.5.2 _feedType	73
6.11.5.3 _isConsolidatedFeed	73
6.11.5.4 _isRunningLocally	74
6.11.5.5 _isSubscription	74
6.11.5.6 _isUnlinkFifo	74
6.11.5.7 _symbolCount	74
6.11.5.8 _symbolId	74
6.11.5.9 FD_LENGTH	74
6.11.5.10 MAX_SYMBOL_COUNT	74
6.12 MarketData::SymbolIdFdMap Class Reference	74
6.12.1 Detailed Description	76

6.12.2	Member Function Documentation	76
6.12.2.1	addFd	76
6.12.2.2	addFdInConsolidatedMap	76
6.12.2.3	deleteAllSymbolOfConsolidatedFd	77
6.12.2.4	deleteAllSymbolOfFd	77
6.12.2.5	deleteFd	78
6.12.2.6	deleteFdFromConsolidatedMap	78
6.12.2.7	getFdForSymbol	79
6.12.2.8	getFdForSymbolConsolidated	79
6.12.2.9	getInstance	79
6.12.2.10	getSymbolForFdConsolidated	80
6.12.2.11	initialize	80
6.12.3	Member Data Documentation	80
6.12.3.1	_consolidatedFdIdMap	80
6.12.3.2	_consolidatedIdFdMap	80
6.12.3.3	_fdIdMap	81
6.12.3.4	_idFdMap	81
6.12.3.5	_iter	81
6.13	MarketData::SymbolIdParser Class Reference	81
6.13.1	Member Function Documentation	81
6.13.1.1	getMcISymbolsFromFile	82
6.13.1.2	initializeSymbolIdNameMap	82
6.13.1.3	setExchangeld	83
6.13.2	Member Data Documentation	83
6.13.2.1	_exchangeld	83
6.14	MarketData::SymbolNameIdMap Class Reference	83
6.14.1	Constructor & Destructor Documentation	84
6.14.1.1	SymbolNameIdMap	84
6.14.2	Member Function Documentation	84
6.14.2.1	getId	84
6.14.2.2	getInstance	84
6.14.2.3	setId	84
6.14.3	Member Data Documentation	84
6.14.3.1	_symbolNameIdMap	84
6.15	MarketData::ThreadPool Class Reference	85
6.15.1	Constructor & Destructor Documentation	85
6.15.1.1	ThreadPool	85
6.15.1.2	~ThreadPool	85
6.15.2	Member Function Documentation	85
6.15.2.1	getIoService	85

6.15.2.2	post	86
6.15.3	Member Data Documentation	86
6.15.3.1	_ioService	86
6.15.3.2	_threads	86
6.15.3.3	_work	86
7	File Documentation	87
7.1	commands.h File Reference	87
7.1.1	Macro Definition Documentation	89
7.1.1.1	EXCHANGE_ID_BASE	89
7.1.1.2	GET_EXCHANGE_ID	89
7.1.1.3	GET_SCRIP_CODE	89
7.1.1.4	GET_SYMBOL_ID	89
7.2	defines.cpp File Reference	89
7.2.1	Macro Definition Documentation	90
7.2.1.1	_WINSOCKAPI_	90
7.2.2	Function Documentation	90
7.2.2.1	generateUniqueOrderId	90
7.2.2.2	htonl_32	90
7.2.2.3	htonl_64	90
7.2.2.4	htons_16	91
7.2.2.5	ntohl_32	91
7.2.2.6	ntohl_64	91
7.2.2.7	ntohs_16	91
7.2.2.8	signed_htonl_64	91
7.2.2.9	signed_ntohl_64	91
7.2.3	Variable Documentation	91
7.2.3.1	uniqueOrderIdMutex	91
7.3	defines.h File Reference	91
7.3.1	Macro Definition Documentation	93
7.3.1.1	ACCOUNT_FIELD_SIZE	93
7.3.1.2	CHARACTER_FIELD_SIZE	93
7.3.1.3	DESERIALIZE_16	93
7.3.1.4	DESERIALIZE_32	93
7.3.1.5	DESERIALIZE_64	93
7.3.1.6	DESERIALIZE_8	93
7.3.1.7	DESERIALIZE_SIGNED_64	93
7.3.1.8	IS_BIG_ENDIAN	94
7.3.1.9	ISIN_SIZE	94
7.3.1.10	MARKET_NAME_SIZE	94

7.3.1.11	MARKET_PICTURE_BROADCAST_FLAG_SIZE	94
7.3.1.12	PASSWORD_SALT_SIZE	94
7.3.1.13	PASSWORD_SIZE	94
7.3.1.14	PRODUCT_CODE_SIZE	94
7.3.1.15	SCRIP_NAME_SIZE	94
7.3.1.16	SERIALIZE_16	94
7.3.1.17	SERIALIZE_32	94
7.3.1.18	SERIALIZE_64	94
7.3.1.19	SERIALIZE_8	94
7.3.1.20	SERIALIZE_SIGNED_64	94
7.3.1.21	SERIES_SIZE	95
7.3.1.22	SWAP16	95
7.3.1.23	SWAP32	95
7.3.1.24	SWAP64	95
7.3.1.25	SYMBOL_ALIAS_SIZE	95
7.3.1.26	SYMBOL_SIZE	95
7.3.2	Typedef Documentation	95
7.3.2.1	SIGNED_LONG	95
7.3.2.2	UNSIGNED_CHARACTER	95
7.3.2.3	UNSIGNED_INTEGER	95
7.3.2.4	UNSIGNED_LONG	95
7.3.2.5	UNSIGNED_SHORT	95
7.3.3	Function Documentation	95
7.3.3.1	getPacket	95
7.3.3.2	htonl_32	95
7.3.3.3	htonl_64	95
7.3.3.4	htons_16	96
7.3.3.5	ntohl_32	96
7.3.3.6	ntohl_64	96
7.3.3.7	ntohs_16	96
7.3.3.8	signed_htonl_64	96
7.3.3.9	signed_ntohl_64	96
7.4	dispatcherMain.cpp File Reference	96
7.4.1	Function Documentation	96
7.4.1.1	main	97
7.5	fdDispatcher.cpp File Reference	97
7.6	fdDispatcher.h File Reference	98
7.7	mdStorage.cpp File Reference	99
7.8	mdStorage.h File Reference	99
7.9	subscriptionServer.cpp File Reference	100

7.10	subscriptionServer.h File Reference	101
7.10.1	Macro Definition Documentation	102
7.10.1.1	MAXDATABUFFER	102
7.10.1.2	MAXLINE	102
7.11	subscriptionServMsg.cpp File Reference	102
7.12	subscriptionServMsg.h File Reference	102
7.13	symbolIdFdMap.cpp File Reference	103
7.14	symbolIdFdMap.h File Reference	104
7.15	symbolIdParser.cpp File Reference	105
7.16	symbolIdParser.h File Reference	106
7.17	threadPool.h File Reference	107

Chapter 1

muTrade Market Data Disptacher documentation

1.1 Introduction

It is subscription based market data event provider. Developer need to subscribe the symbol from our subscription server. Once symbol is subscribed user will get the event as snapshot for every tick update of respective symbol.

Code Flow

- 1) Developer need to subscribe the symbol using [SubscriptionServMsg](#) with TCP based connection.
- 2) Once subscription is done appliaction will get event as [MDQuote](#) on the same socket.

Currently market data feed server supports four types of event.

- 1) Event as symbol Id for every snapshot update.
- 2) Event as symbol Id for every TBT update.
- 3) Event as top 5 snapshot market depth for every snapshot update.
- 4) Snapshot of consolidated order depth of two or more symbols.

Sanpshot for single symbol is available as [MDQuote](#) where as consolidated feed is available as [MDQuote-Consolidated](#) .

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

CMD	9
MarketData	12

Chapter 3

Class Index

3.1 Class List

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

MarketData::FdDispatcher	
FdDispatcher Class	15
MarketData::FdState	31
CMD::InstrumentRequest	
Instrument search request	32
CMD::MDQuote	
MarketData Quote	34
CMD::MDQuoteConsolidated	
MarketData Quote Consolidated	40
MarketData::MDStorage	
MDStorage Class	48
CMD::MDSubscribeRequest	
MarketData Subscribe/Unsubscribe Request	51
CMD::ScripMasterDataRequest	53
Subscription	
It accepts request for symbol subscription from the client and updates symbol-fds map. Which is used by disppatcher to send event for every tick update	58
MarketData::SubscriptionServer	58
MarketData::SubscriptionServMsg	
This class is responsible for serialization of request data to the server	64
MarketData::SymbolIdFdMap	
SymbolIdFdMap class This is the data structure which is used by tbt to send event on each subscriber for each symbol id	74
MarketData::SymbolIdParser	81
MarketData::SymbolNameIdMap	83
MarketData::ThreadPool	85

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

commands.h	87
defines.cpp	89
defines.h	91
dispatcherMain.cpp	96
fdDispatcher.cpp	97
fdDispatcher.h	98
mdStorage.cpp	99
mdStorage.h	99
subscriptionServer.cpp	100
subscriptionServer.h	101
subscriptionServMsg.cpp	102
subscriptionServMsg.h	102
symbolIdFdMap.cpp	103
symbolIdFdMap.h	104
symbolIdParser.cpp	105
symbolIdParser.h	106
threadPool.h	107

Chapter 5

Namespace Documentation

5.1 CMD Namespace Reference

Classes

- class [ScripMasterDataRequest](#)
- class [InstrumentRequest](#)
Instrument search request.
- class [MDSubscribeRequest](#)
MarketData Subscribe/Unsubscribe Request.
- class [MDQuote](#)
MarketData Quote.
- class [MDQuoteConsolidated](#)
MarketData Quote Consolidated.

Enumerations

- enum [Exchangeld](#) {
[Exchangeld_BSE](#) = 1, [Exchangeld_NSE](#), [Exchangeld_ESMNSE](#), [Exchangeld_SGX](#),
[Exchangeld_NSECDS](#), [Exchangeld_BSEETI](#), [Exchangeld_CFH](#), [Exchangeld_MAX](#) }
- enum [CommandCategory](#) {
[CommandCategory_SINGLE_ORDER](#), [CommandCategory_DUMMY_TWO_LEG_STRATEGY](#), [CommandCategory_TWO_LEG_ARBITRAGE_STRATEGY](#), [CommandCategory_MARKET_MAKING_STRATEGY](#),
[CommandCategory_STRATEGY_COMMAND](#), [CommandCategory_PAUSE_STRATEGY](#), [CommandCategory_RUN_STRATEGY](#), [CommandCategory_TERMINATE_STRATEGY](#),
[CommandCategory_TERMINATE_SQUAREOFF](#), [CommandCategory_LOGIN](#), [CommandCategory_LOGOUT](#), [CommandCategory_RMS_UPDATE](#),
[CommandCategory_REQUEST_RMS_CONFIGURATION](#), [CommandCategory_REQUEST_OFFLINE_DATA](#), [CommandCategory_HEART_BEAT_MESSAGE](#), [CommandCategory_THREE_LEG_ARBITRAGE_STRATEGY](#),
[CommandCategory_DUMMY_THREE_LEG_ARBITRAGE_STRATEGY](#), [CommandCategory_BSE_DISCONNECTED](#), [CommandCategory_NSECM_DISCONNECTED](#), [CommandCategory_NSEFO_DISCONNECTED](#),
[CommandCategory_TBTCM_DISCONNECTED](#), [CommandCategory_TBTFO_DISCONNECTED](#), [CommandCategory_BSE_SNAPSHOTFEED_DISCONNECTED](#), [CommandCategory_NSEFO_SNAPSHOTFEED_DISCONNECTED](#),
[CommandCategory_NSECM_SNAPSHOTFEED_DISCONNECTED](#), [CommandCategory_MARKET_MAKING_TURNOVER_STRATEGY](#), [CommandCategory_TBT_MARKET_DATA](#), [CommandCategory_FOUR_LEG_BUTTERFLY_STRATEGY](#),
[CommandCategory_BOX_STRATEGY](#), [CommandCategory_TWO_LEG_THREE_LEG_ARBITRAGE_STR-](#)

```

ATEGY, CommandCategory_MODIFY_STRATEGY, CommandCategory_RMS_LOGIN_PARAMETERS,
CommandCategory_RMS_ORDER_LIMITS, CommandCategory_RMS_GLOBAL_PARAMETERS, Command-
Category_RMS_RISK_PERMISSIONS, CommandCategory_GET_RMS_PARAMETERS,
CommandCategory_CHANGE_PASSWORD, CommandCategory_SEND_SCRIP_MASTER_DATA,
CommandCategory_GET_TOTAL_RECORDS_SCRIP_MASTER, CommandCategory_GET_TOTAL_O-
RDER_CONFIRMATIONS,
CommandCategory_GET_CONNECTIVITY_STATUS, CommandCategory_RMS_SECURITY_WISE_LIMI-
TS, CommandCategory_IMPLIED_VOLATILITY_STRATEGY, CommandCategory_CASH_CASH_STRAT-
EGY,
CommandCategory_GET_CLIENT_CONNECTIVITY_STATUS, CommandCategory_FORCE_LOG_OFF_-
CLIENT, CommandCategory_PAIRS_STRATEGY, CommandCategory_PAIRS_CLOSE_POSITIONS_NO-
_TERMINATE,
CommandCategory_TBTCDS_DISCONNECTED, CommandCategory_NSECDS_DISCONNECTED,
CommandCategory_BSEETI_DISCONNECTED, CommandCategory_SGX_DISCONNECTED,
CommandCategory_INSTRUMENT_SEARCH_REQUEST, CommandCategory_MARKET_DATA_SUBSC-
RIPTION, CommandCategory_API_TRADED_SYMBOLS_COUNT_REQUEST, CommandCategory_API_-
REPLAY_TRADED_SYMBOLS_REQUEST,
CommandCategory_MAX }

```

5.1.1 Enumeration Type Documentation

5.1.1.1 enum CMD::CommandCategory

Enumerator:

```

CommandCategory_SINGLE_ORDER
CommandCategory_DUMMY_TWO_LEG_STRATEGY
CommandCategory_TWO_LEG_ARBITRAGE_STRATEGY
CommandCategory_MARKET_MAKING_STRATEGY
CommandCategory_STRATEGY_COMMAND
CommandCategory_PAUSE_STRATEGY
CommandCategory_RUN_STRATEGY
CommandCategory_TERMINATE_STRATEGY
CommandCategory_TERMINATE_SQUAREOFF
CommandCategory_LOGIN
CommandCategory_LOGOUT
CommandCategory_RMS_UPDATE
CommandCategory_REQUEST_RMS_CONFIGURATION
CommandCategory_REQUEST_OFFLINE_DATA
CommandCategory_HEART_BEAT_MESSAGE
CommandCategory_THREE_LEG_ARBITRAGE_STRATEGY
CommandCategory_DUMMY_THREE_LEG_ARBITRAGE_STRATEGY
CommandCategory_BSE_DISCONNECTED
CommandCategory_NSECM_DISCONNECTED
CommandCategory_NSEFO_DISCONNECTED
CommandCategory_TBTCM_DISCONNECTED
CommandCategory_TBTFO_DISCONNECTED
CommandCategory_BSE_SNAPSHOTFEED_DISCONNECTED
CommandCategory_NSEFO_SNAPSHOTFEED_DISCONNECTED
CommandCategory_NSECM_SNAPSHOTFEED_DISCONNECTED

```

CommandCategory_MARKET_MAKING_TURNOVER_STRATEGY
CommandCategory_TBT_MARKET_DATA
CommandCategory_FOUR_LEG_BUTTERFLY_STRATEGY
CommandCategory_BOX_STRATEGY
CommandCategory_TWO_LEG_THREE_LEG_ARBITRAGE_STRATEGY
CommandCategory_MODIFY_STRATEGY
CommandCategory_RMS_LOGIN_PARAMETERS
CommandCategory_RMS_ORDER_LIMITS
CommandCategory_RMS_GLOBAL_PARAMETERS
CommandCategory_RMS_RISK_PERMISSIONS
CommandCategory_GET_RMS_PARAMETERS
CommandCategory_CHANGE_PASSWORD
CommandCategory_SEND_SCRIP_MASTER_DATA
CommandCategory_GET_TOTAL_RECORDS_SCRIP_MASTER
CommandCategory_GET_TOTAL_ORDER_CONFIRMATIONS
CommandCategory_GET_CONNECTIVITY_STATUS
CommandCategory_RMS_SECURITY_WISE_LIMITS
CommandCategory_IMPLIED_VOLATILITY_STRATEGY
CommandCategory_CASH_CASH_STRATEGY
CommandCategory_GET_CLIENT_CONNECTIVITY_STATUS
CommandCategory_FORCE_LOG_OFF_CLIENT
CommandCategory_PAIRS_STRATEGY
CommandCategory_PAIRS_CLOSE_POSITIONS_NO_TERMINATE
CommandCategory_TBTCDS_DISCONNECTED
CommandCategory_NSECDS_DISCONNECTED
CommandCategory_BSEETI_DISCONNECTED
CommandCategory_SGX_DISCONNECTED
CommandCategory_INSTRUMENT_SEARCH_REQUEST
CommandCategory_MARKET_DATA_SUBSCRIPTION
CommandCategory_API_TRADED_SYMBOLS_COUNT_REQUEST
CommandCategory_API_REPLAY_TRADED_SYMBOLS_REQUEST
CommandCategory_MAX

5.1.1.2 enum CMD::Exchangeld

Enumerator:

Exchangeld_BSE
Exchangeld_NSE
Exchangeld_ESMNSE
Exchangeld_SGX
Exchangeld_NSECDS
Exchangeld_BSEETI
Exchangeld_CFH
Exchangeld_MAX

5.2 MarketData Namespace Reference

Classes

- class [FdDispatcher](#)
FdDispatcher Class.
- class [MDStorage](#)
MDStorage Class.
- struct [FdState](#)
- class [SubscriptionServer](#)
- class [SubscriptionServMsg](#)
This class is responsible for serialization of request data to the server.
- class [SymbolIdFdMap](#)
SymbolIdFdMap class This is the data structure which is used by tbt to send event on each subscriber for each symbol id.
- class [SymbolNameIdMap](#)
- class [SymbolIdParser](#)
- class [ThreadPool](#)

Typedefs

- typedef std::unordered_map< int, std::deque< std::pair< int, int > > > [IdFdMap](#)
- typedef std::unordered_map< int, std::deque< std::pair< int, int > > >::iterator [IdFdMapItr](#)
- typedef std::unordered_map< int, std::vector< int > > [FdIdMap](#)
- typedef std::unordered_map< int, std::deque< int > > [ConsolidatedIdFdMap](#)

Enumerations

- enum [FeedType](#) { [TBT](#) = 0, [MCL](#) = 1 }

Functions

- template<class T >
int [findIndex](#) (T &vec, int id)
- void [getCurrentLocalTimeStamp](#) ()

5.2.1 Typedef Documentation

5.2.1.1 typedef std::unordered_map<int, std::deque<int> > **MarketData::ConsolidatedIdFdMap**

5.2.1.2 typedef std::unordered_map<int, std::vector<int> > **MarketData::FdIdMap**

5.2.1.3 typedef std::unordered_map<int, std::deque<std::pair<int, int> > > **MarketData::IdFdMap**

5.2.1.4 typedef std::unordered_map<int, std::deque<std::pair<int, int> > >::iterator **MarketData::IdFdMapItr**

5.2.2 Enumeration Type Documentation

5.2.2.1 enum MarketData::FeedType

Enumerator:

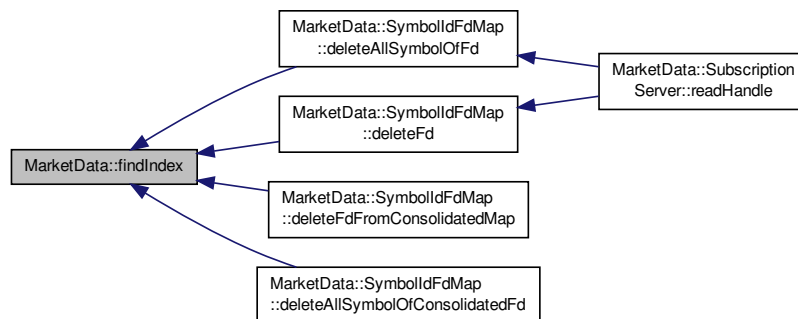
TBT

MCL

5.2.3 Function Documentation

5.2.3.1 template<class T> int MarketData::findIndex (T & vec, int id)

Here is the caller graph for this function:



5.2.3.2 void MarketData::getCurrentLocalTimeStamp () [inline]

Chapter 6

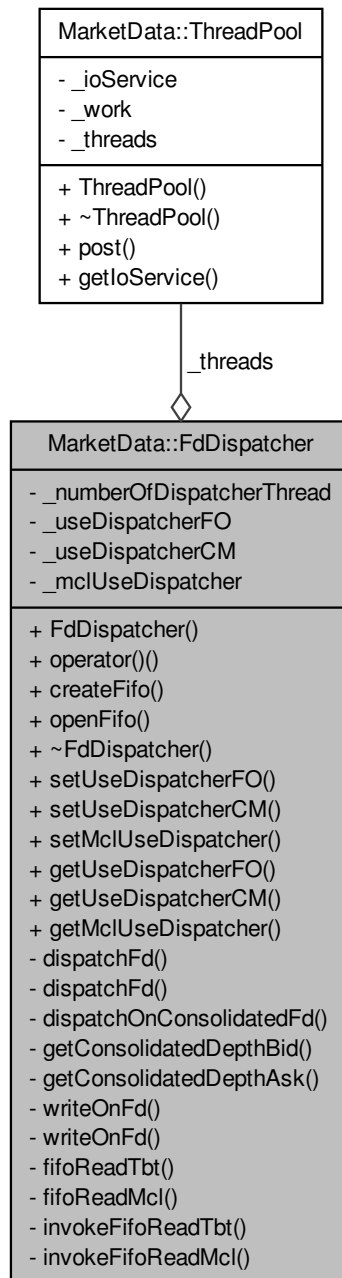
Class Documentation

6.1 MarketData::FdDispatcher Class Reference

[FdDispatcher](#) Class.

```
#include <fdDispatcher.h>
```

Collaboration diagram for MarketData::FdDispatcher:



Public Member Functions

- [FdDispatcher](#) (int num=5)
Constructor.
- void [operator](#)() ()
Overloaded function operator.
- int [createFifo](#) (const char *fifo)

- *Function to create fifo if dispatcher and client both are on the same system.*
- `int openFifo (const char *fifo)`
Function to open fifo if dispatcher and client both are on the same system.
- `~FdDispatcher ()`
Destructor.

Static Public Member Functions

- `static void setUseDispatcherFO (bool val)`
Internally used function to set whether to use TBT FO dispatcher or not. Value is updated from setting file.
- `static void setUseDispatcherCM (bool val)`
Internally used function to set whether to use TBT CM dispatcher or not. Value is updated from setting file.
- `static void setMclUseDispatcher (bool val)`
Internally used function to set whether to use MCL dispatcher or not. Value is updated from setting file.
- `static bool getUseDispatcherFO ()`
Internally used function to check whether to use TBT FO dispatcher or not. Value is updated from setting file.
- `static bool getUseDispatcherCM ()`
Internally used function to check whether to use TBT CM dispatcher or not. Value is updated from setting file.
- `static bool getMclUseDispatcher ()`
Internally used function to check whether to use MCL dispatcher or not. Value is updated from setting file.

Private Member Functions

- `void dispatchFd (int, int)`
Feed Dispatcher Function Function Called when there is an event for market data feed.
- `void dispatchFd (int, int, CMD::MDQuote &)`
Feed Dispatcher Function Function Called when there is an event for market data feed.
- `void dispatchOnConsolidatedFd (int, CMD::MDQuote &)`
Feed Dispatcher Function Function Called when there is an event for market data feed.
- `void getConsolidatedDepthBid (std::map< int, int > &symbolIdDepthMap, std::vector< CMD::MDQuote > &mdq, CMD::MDQuoteConsolidated &mdqConsolidated)`
- `void getConsolidatedDepthAsk (std::map< int, int > &symbolIdDepthMap, std::vector< CMD::MDQuote > &mdq, CMD::MDQuoteConsolidated &mdqConsolidated)`
- `void writeOnFd (int, int, int, int)`
Write event on fd Function responsible for writing the event with symbol id on socket/fd.
- `void writeOnFd (int, int, int, int, CMD::MDQuote &)`
Write event on fd Function responsible for writing the event with symbol id and snapshot on socket/fd.
- `void fifoReadTbt (evutil_socket_t, short, void *)`
Read Tbt Data From Fifo Function is called to read the feed when there is an event for tbt feed.
- `void fifoReadMcl (evutil_socket_t, short, void *)`
Fifo Read Mcl Function is called to read the feed when there is an event for mcl feed.

Static Private Member Functions

- `static void invokeFifoReadTbt (evutil_socket_t, short, void *)`
Invoke Fifo Read Tbt Since libevent is a C library, it doesn't directly provide a mechanism to associate an instance and an instance method, so we need to do it ourself.
- `static void invokeFifoReadMcl (evutil_socket_t, short, void *)`
Invoke Fifo Read Since libevent is a C library, it doesn't directly provide a mechanism to associate an instance and an instance method, so we need to do it ourself.

Private Attributes

- `int _numberOfDispatcherThread`
- `ThreadPool _threads`

Static Private Attributes

- `static bool _useDispatcherFO = true`
- `static bool _useDispatcherCM = true`
- `static bool _mclUseDispatcher = true`

6.1.1 Detailed Description

`FdDispatcher` Class.

This class is responsible for writing event on every subscribed symbols.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `MarketData::FdDispatcher::FdDispatcher (int num = 5)`

Constructor.

Parameters

<i>num</i>	number of dispatcher threads
------------	------------------------------

6.1.2.2 `MarketData::FdDispatcher::~~FdDispatcher ()`

Destructor.

6.1.3 Member Function Documentation

6.1.3.1 `int MarketData::FdDispatcher::createFifo (const char * fifo)`

Function to create fifo if dispatcher and client both are on the same system.

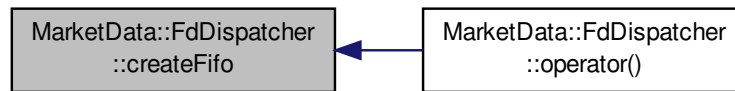
Parameters

<i>fifo</i>	name
-------------	------

Returns

0/-1 Success/Failure

Here is the caller graph for this function:



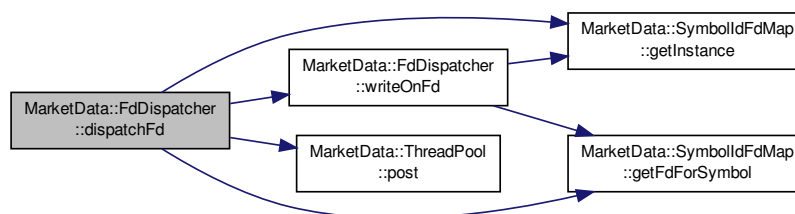
6.1.3.2 void MarketData::FdDispatcher::dispatchFd (int *symbolId*, int *feedType*) [private]

Feed Dispatcher Function Function Called when there is an event for market data feed.

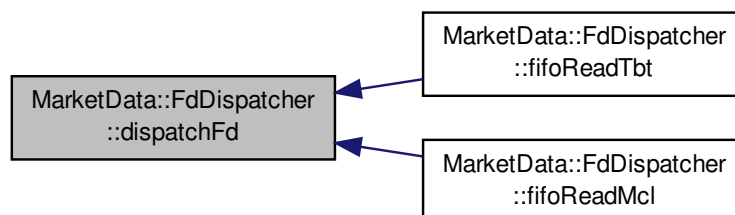
Parameters

<i>int</i>	Symbol Id
<i>int</i>	Feed Type(TBT/Snapshot)

Here is the call graph for this function:



Here is the caller graph for this function:



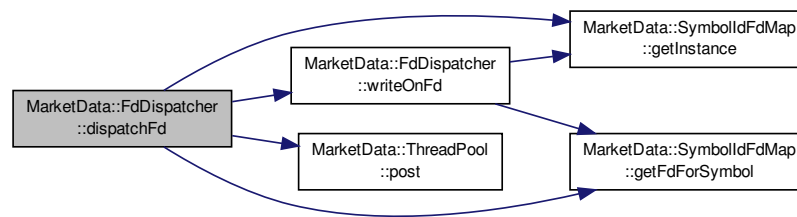
6.1.3.3 void MarketData::FdDispatcher::dispatchFd (int *symbolId*, int *feedType*, CMD::MDQuote & *mdq*) [private]

Feed Dispatcher Function Function Called when there is an event for market data feed.

Parameters

<i>int</i>	Symbol Id
<i>int</i>	Feed Type(TBT/Snapshot)
<i>MDQuote</i>	reference

Here is the call graph for this function:



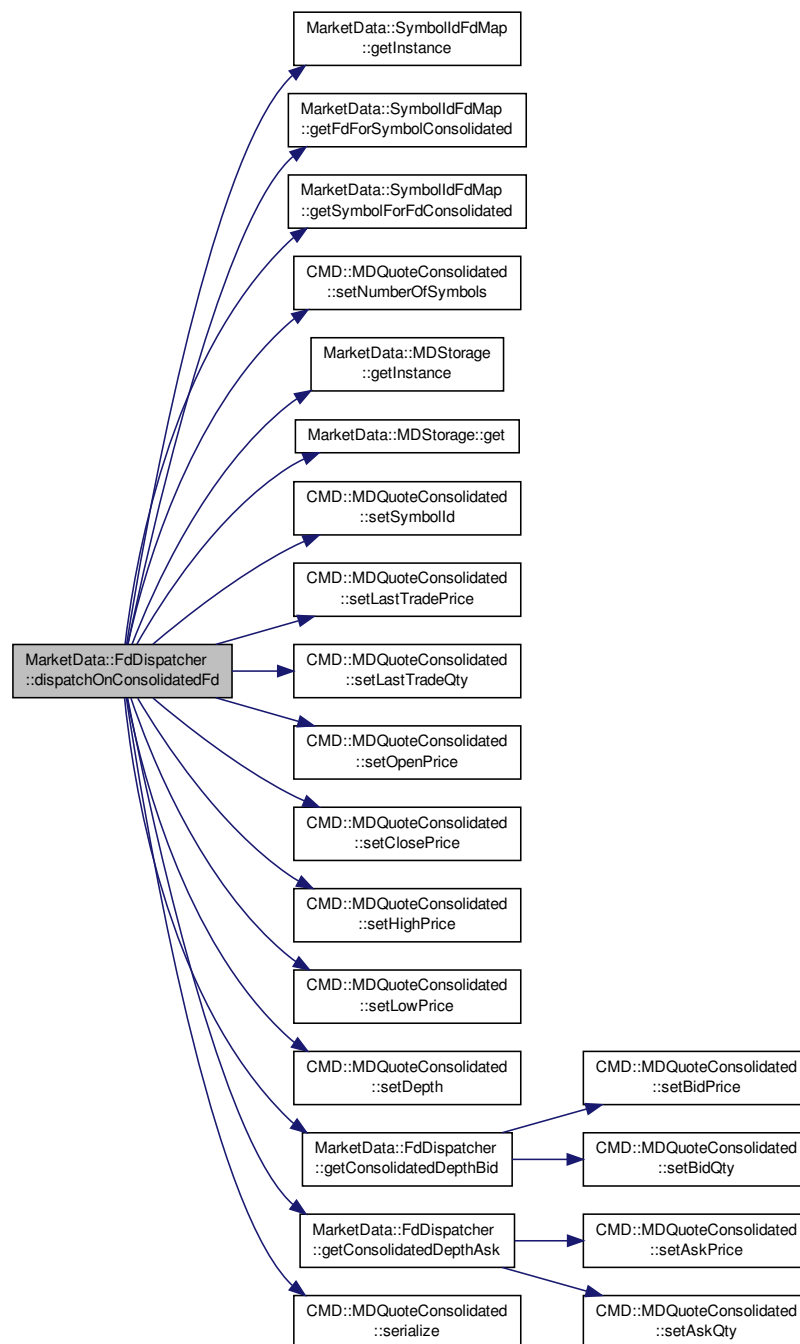
6.1.3.4 void MarketData::FdDispatcher::dispatchOnConsolidatedFd (int *symbolId*, CMD::MDQuote & *mdq*) [private]

Feed Dispatcher Function Function Called when there is an event for market data feed.

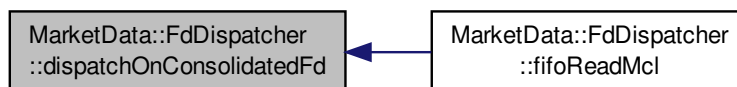
Parameters

<i>int</i>	Symbol Id
<i>MDQuote</i>	reference

Here is the call graph for this function:



Here is the caller graph for this function:



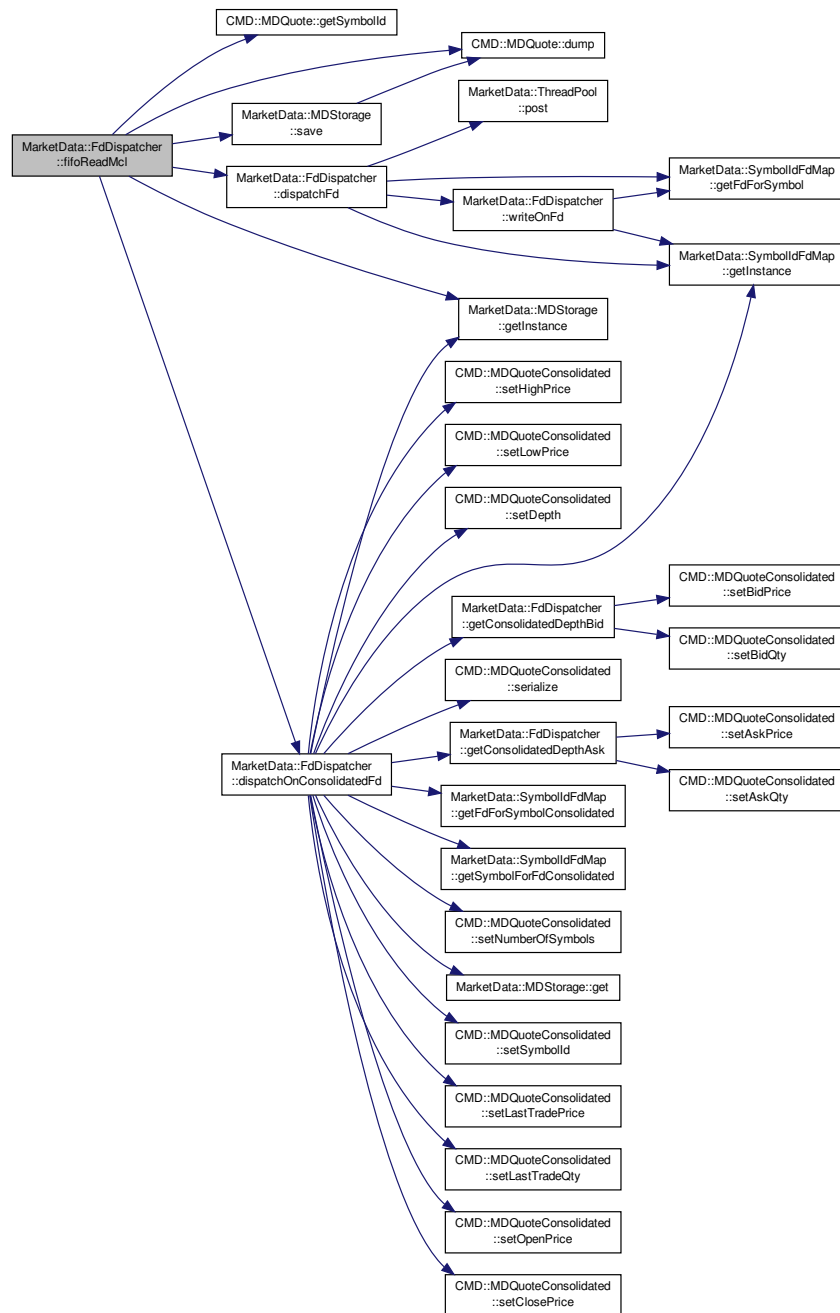
6.1.3.5 `void MarketData::FdDispatcher::fifoReadMcl (evutil_socket_t fd, short event, void * arg)` [private]

Fifo Read Mcl Function is called to read the feed when there is an event for mcl feed.

Parameters

<i>evutil_socket_t</i>	<i>fd</i>
<i>short</i>	<i>event</i>
<i>void</i>	pointer (struct event)

Here is the call graph for this function:



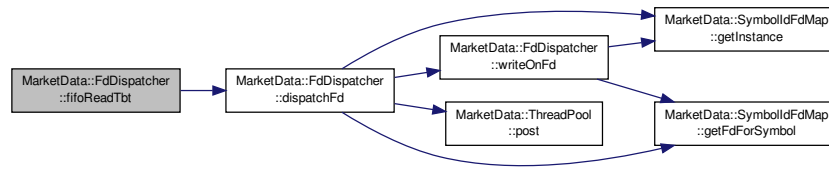
6.1.3.6 void MarketData::FdDispatcher::fifoReadTbt (evutil_socket_t fd, short event, void * arg) [private]

Read Tbt Data From Fifo Function is called to read the feed when there is an event for tbt feed.

Parameters

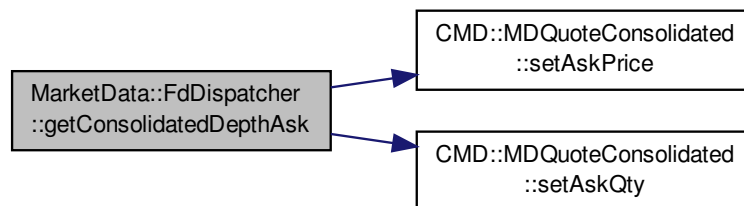
<i>evutil_socket_t</i>	
<i>short</i>	
<i>@return</i>	

Here is the call graph for this function:

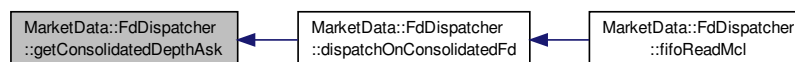


6.1.3.7 `void MarketData::FdDispatcher::getConsolidatedDepthAsk (std::map< int, int > & symbolIdDepthMap, std::vector< CMD::MDQuote > & mdq, CMD::MDQuoteConsolidated & mdqConsolidated) [private]`

Here is the call graph for this function:

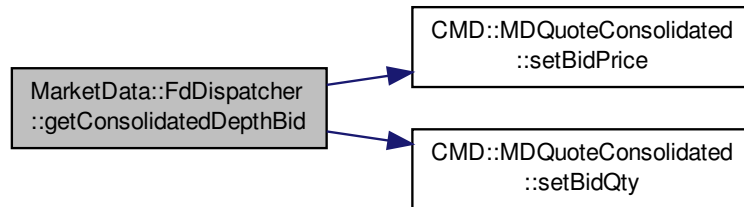


Here is the caller graph for this function:

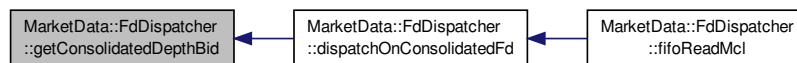


6.1.3.8 void MarketData::FdDispatcher::getConsolidatedDepthBid (std::map< int, int > & *symbolIdDepthMap*, std::vector< CMD::MDQuote > & *mdq*, CMD::MDQuoteConsolidated & *mdqConsolidated*) [private]

Here is the call graph for this function:



Here is the caller graph for this function:



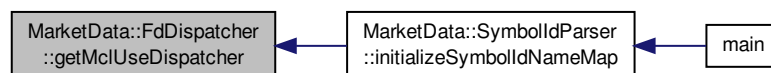
6.1.3.9 static bool MarketData::FdDispatcher::getMclUseDispatcher () [inline],[static]

Internally used function to check whether to use MCL dispatcher or not. Value is updated from setting file.

Parameters

<i>val</i>	True/False
------------	------------

Here is the caller graph for this function:



6.1.3.10 static bool MarketData::FdDispatcher::getUseDispatcherCM () [inline],[static]

Internally used function to check whether to use TBT CM dispatcher or not. Value is updated from setting file.

Parameters

<i>val</i>	True/False
------------	------------

6.1.3.11 static bool MarketData::FdDispatcher::getUseDispatcherFO () [inline],[static]

Internally used function to check whether to use TBT FO dispatcher or not. Value is updated from setting file.

Parameters

<i>val</i>	True/False
------------	------------

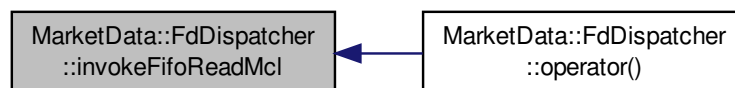
6.1.3.12 void MarketData::FdDispatcher::invokeFifoReadMcl (evutil_socket_t *fd*, short *events*, void * *arg*) [static],[private]

Invoke Fifo Read Since libevent is a C library, it doesn't directly provide a mechanism to associate an instance and an instance method, so we need to do it ourself.

Parameters

<i>evutil_socket_t</i>	<i>fd</i>
<i>short</i>	<i>event</i>
<i>void</i>	pointer(struct event)

Here is the caller graph for this function:

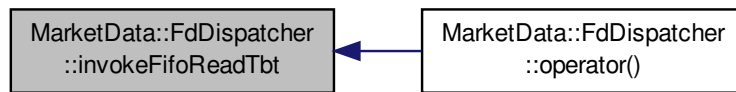
6.1.3.13 void MarketData::FdDispatcher::invokeFifoReadTbt (evutil_socket_t *fd*, short *events*, void * *arg*) [static],[private]

Invoke Fifo Read Tbt Since libevent is a C library, it doesn't directly provide a mechanism to associate an instance and an instance method, so we need to do it ourself.

Parameters

<i>evutil_socket_t</i>	<i>fd</i>
<i>short</i>	<i>event</i>
<i>void</i>	pointer(struct event)

Here is the caller graph for this function:



6.1.3.14 int MarketData::FdDispatcher::openFifo (const char * *fifo*)

Function to open fifo if dispatcher and client both are on the same system.

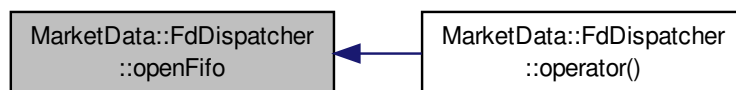
Parameters

<i>fifo</i>	fifo name.
-------------	------------

Returns

0/-1 Success/Failure

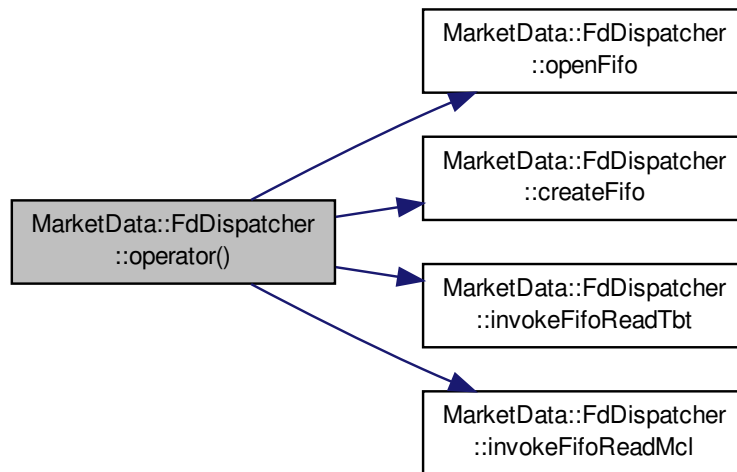
Here is the caller graph for this function:



6.1.3.15 void MarketData::FdDispatcher::operator() ()

Overloaded function operator.

Here is the call graph for this function:



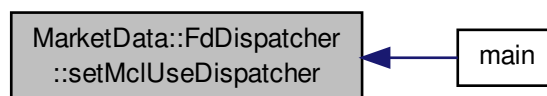
6.1.3.16 `static void MarketData::FdDispatcher::setMclUseDispatcher (bool val)` `[inline],[static]`

Internally used function to set whether to use MCL dispatcher or not. Value is updated from setting file.

Parameters

<i>val</i>	True/False
------------	------------

Here is the caller graph for this function:



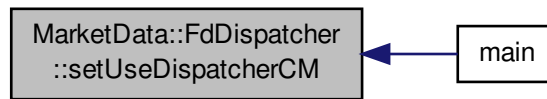
6.1.3.17 `static void MarketData::FdDispatcher::setUseDispatcherCM (bool val)` `[inline],[static]`

Internally used function to set whether to use TBT CM dispatcher or not. Value is updated from setting file.

Parameters

<i>val</i>	True/False
------------	------------

Here is the caller graph for this function:



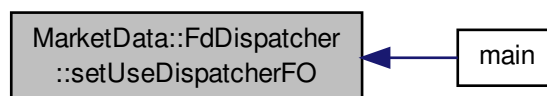
6.1.3.18 `static void MarketData::FdDispatcher::setUseDispatcherFO (bool val)` `[inline],[static]`

Internally used function to set whether to use TBT FO dispatcher or not. Value is updated from setting file.

Parameters

<i>val</i>	True/False
------------	------------

Here is the caller graph for this function:



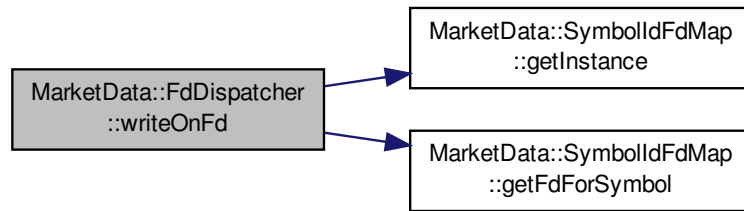
6.1.3.19 `void MarketData::FdDispatcher::writeOnFd (int from, int to, int symbolId, int feedType)` `[private]`

Write event on fd Function responsible for writing the event with symbol id on socket/fd.

Parameters

<i>int</i>	<i>from</i>
<i>int</i>	<i>to</i>
<i>int</i>	Symbol Id
<i>int</i>	Feed Type

Here is the call graph for this function:



Here is the caller graph for this function:



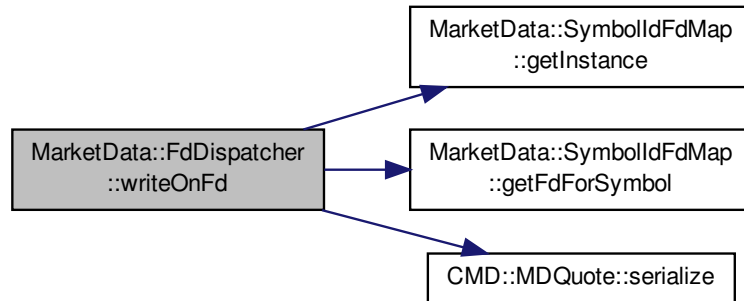
6.1.3.20 `void MarketData::FdDispatcher::writeOnFd (int from, int to, int symbolId, int feedType, CMD::MDQuote & mdq)`
`[private]`

Write event on fd Function responsible for writing the event with symbol id and snapshot on socket/fd.

Parameters

<i>int</i>	from
<i>int</i>	to
<i>int</i>	Symbol Id
<i>int</i>	Feed Type
<i>MDQuote</i>	

Here is the call graph for this function:



6.1.4 Member Data Documentation

6.1.4.1 `bool MarketData::FdDispatcher::_mclUseDispatcher = true` `[static]`, `[private]`

6.1.4.2 `int MarketData::FdDispatcher::_numberOfDispatcherThread` `[private]`

6.1.4.3 `ThreadPool MarketData::FdDispatcher::_threads` `[private]`

6.1.4.4 `bool MarketData::FdDispatcher::_useDispatcherCM = true` `[static]`, `[private]`

6.1.4.5 `bool MarketData::FdDispatcher::_useDispatcherFO = true` `[static]`, `[private]`

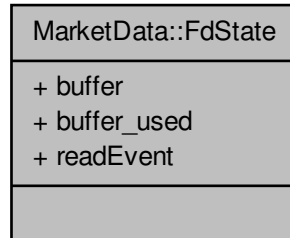
The documentation for this class was generated from the following files:

- [fdDispatcher.h](#)
- [fdDispatcher.cpp](#)

6.2 MarketData::FdState Struct Reference

```
#include <subscriptionServer.h>
```

Collaboration diagram for MarketData::FdState:



Public Attributes

- char [buffer](#) [[MAXLINE](#)]
- size_t [buffer_used](#)
- struct event * [readEvent](#)

6.2.1 Member Data Documentation

6.2.1.1 char MarketData::FdState::buffer[MAXLINE]

6.2.1.2 size_t MarketData::FdState::buffer_used

6.2.1.3 struct event* MarketData::FdState::readEvent

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

- [subscriptionServer.h](#)

6.3 CMD::InstrumentRequest Class Reference

Instrument search request.

```
#include <commands.h>
```


Collaboration diagram for CMD::InstrumentRequest:

CMD::InstrumentRequest
<ul style="list-style-type: none"> - <code>_mnemonic</code> - <code>MNEMONIC_LENGTH</code>
<ul style="list-style-type: none"> + <code>InstrumentRequest()</code> + <code>InstrumentRequest()</code> + <code>serialize()</code> + <code>initialize()</code> + <code>getMnemonic()</code> + <code>setMnemonic()</code>

Public Member Functions

- [InstrumentRequest](#) ()
- [InstrumentRequest](#) (const char *buf)
- int [serialize](#) (char *buf)
- void [initialize](#) ()
- const char * [getMnemonic](#) () const
- void [setMnemonic](#) (const char *m)

Private Attributes

- char [_mnemonic](#) [[MNEMONIC_LENGTH](#)]

Static Private Attributes

- static const int [MNEMONIC_LENGTH](#) = 100

6.3.1 Detailed Description

Instrument search request.

In response, the server will send SymbolStaticData if the instrument was successfully found, otherwise it will just return a FAILURE message.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 CMD::InstrumentRequest::InstrumentRequest ()

6.3.2.2 CMD::InstrumentRequest::InstrumentRequest (const char * *buf*)

6.3.3 Member Function Documentation

6.3.3.1 `const char* CMD::InstrumentRequest::getMnemonic () const` `[inline]`

6.3.3.2 `void CMD::InstrumentRequest::initialize ()`

6.3.3.3 `int CMD::InstrumentRequest::serialize (char * buf)`

6.3.3.4 `void CMD::InstrumentRequest::setMnemonic (const char * m)` `[inline]`

6.3.4 Member Data Documentation

6.3.4.1 `char CMD::InstrumentRequest::_mnemonic[MNEMONIC_LENGTH]` `[private]`

6.3.4.2 `const int CMD::InstrumentRequest::MNEMONIC_LENGTH = 100` `[static], [private]`

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

- [commands.h](#)

6.4 CMD::MDQuote Class Reference

[MarketData](#) Quote.

```
#include <commands.h>
```

Collaboration diagram for CMD::MDQuote:

CMD::MDQuote
<ul style="list-style-type: none"> - _symbolId - _numberOfTrades - _volume - _value - _lastTradePrice - _lastTradeQty - _openPrice - _closePrice - _highPrice - _lowPrice and 9 more... - MAX_LOOKUP_LEVEL - QUOTEDATA_LENGTH
<ul style="list-style-type: none"> + MDQuote() + MDQuote() + dumpDepth() + dump() + serialize() + initialize() + setSymbolId() + setNummberOfTrades() + setVolume() + setValue() and 34 more...

Public Member Functions

- [MDQuote](#) ()
- [MDQuote](#) (const char *buf)
- void [dumpDepth](#) (UNSIGNED_LONG *data, UNSIGNED_CHARACTER depth)
- void [dump](#) ()
- int [serialize](#) (char *buf)
- void [initialize](#) ()
- void [setSymbolId](#) (UNSIGNED_LONG val)
- void [setNummberOfTrades](#) (UNSIGNED_LONG val)
- void [setVolume](#) (UNSIGNED_LONG val)
- void [setValue](#) (UNSIGNED_LONG val)
- void [setLastTradePrice](#) (UNSIGNED_LONG val)
- void [setLastTradeQty](#) (UNSIGNED_LONG val)
- void [setOpenPrice](#) (UNSIGNED_LONG val)
- void [setClosePrice](#) (UNSIGNED_LONG val)
- void [setHighPrice](#) (UNSIGNED_LONG val)

- void [setLowPrice](#) (UNSIGNED_LONG val)
- void [setTotalBidQty](#) (UNSIGNED_LONG val)
- void [setTotalAskQty](#) (UNSIGNED_LONG val)
- void [setLowerCktLimit](#) (UNSIGNED_LONG val)
- void [setUpperCktLimit](#) (UNSIGNED_LONG val)
- void [setDepth](#) (UNSIGNED_CHARACTER val)
- void [setBidPrice](#) (UNSIGNED_LONG val[])
- void [setBidQty](#) (UNSIGNED_LONG val[])
- void [setAskPrice](#) (UNSIGNED_LONG val[])
- void [setAskQty](#) (UNSIGNED_LONG val[])
- [UNSIGNED_LONG](#) [getSymbolId](#) ()
- [UNSIGNED_LONG](#) [getNumberOfTrades](#) ()
- [UNSIGNED_LONG](#) [getVolume](#) ()
- [UNSIGNED_LONG](#) [getValue](#) ()
- [UNSIGNED_LONG](#) [getLastTradePrice](#) ()
- [UNSIGNED_LONG](#) [getLastTradeQty](#) ()
- [UNSIGNED_LONG](#) [getOpenPrice](#) ()
- [UNSIGNED_LONG](#) [getClosePrice](#) ()
- [UNSIGNED_LONG](#) [getHighPrice](#) ()
- [UNSIGNED_LONG](#) [getLowPrice](#) ()
- [UNSIGNED_LONG](#) [getTotalBidQty](#) ()
- [UNSIGNED_LONG](#) [getTotalAskQty](#) ()
- [UNSIGNED_LONG](#) [getLowerCktLimit](#) ()
- [UNSIGNED_LONG](#) [getUpperCktLimit](#) ()
- [UNSIGNED_CHARACTER](#) [getDepth](#) ()
- [UNSIGNED_LONG](#) * [getBidPrice](#) ()
- [UNSIGNED_LONG](#) * [getBidQty](#) ()
- [UNSIGNED_LONG](#) * [getAskPrice](#) ()
- [UNSIGNED_LONG](#) * [getAskQty](#) ()

Private Attributes

- [UNSIGNED_LONG](#) [_symbolId](#)
- [UNSIGNED_LONG](#) [_numberOfTrades](#)
- [UNSIGNED_LONG](#) [_volume](#)
- [UNSIGNED_LONG](#) [_value](#)
- [UNSIGNED_LONG](#) [_lastTradePrice](#)
- [UNSIGNED_LONG](#) [_lastTradeQty](#)
- [UNSIGNED_LONG](#) [_openPrice](#)
- [UNSIGNED_LONG](#) [_closePrice](#)
- [UNSIGNED_LONG](#) [_highPrice](#)
- [UNSIGNED_LONG](#) [_lowPrice](#)
- [UNSIGNED_LONG](#) [_totalBidQty](#)
- [UNSIGNED_LONG](#) [_totalAskQty](#)
- [UNSIGNED_LONG](#) [_lowerCktLimit](#)
- [UNSIGNED_LONG](#) [_upperCktLimit](#)
- [UNSIGNED_CHARACTER](#) [_depth](#)
- [UNSIGNED_LONG](#) [_bidPrice](#) [MAX_LOOKUP_LEVEL]
- [UNSIGNED_LONG](#) [_bidQty](#) [MAX_LOOKUP_LEVEL]
- [UNSIGNED_LONG](#) [_askPrice](#) [MAX_LOOKUP_LEVEL]
- [UNSIGNED_LONG](#) [_askQty](#) [MAX_LOOKUP_LEVEL]

Static Private Attributes

- static const int [MAX_LOOKUP_LEVEL](#) = 10
- static const int [QUOTEDATA_LENGTH](#) = 80

6.4.1 Detailed Description

[MarketData](#) Quote.

Server will send the quote to the subscribed members.

6.4.2 Constructor & Destructor Documentation

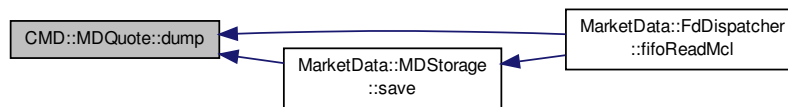
6.4.2.1 `CMD::MDQuote::MDQuote ()`

6.4.2.2 `CMD::MDQuote::MDQuote (const char * buf)`

6.4.3 Member Function Documentation

6.4.3.1 `void CMD::MDQuote::dump ()`

Here is the caller graph for this function:



6.4.3.2 `void CMD::MDQuote::dumpDepth (UNSIGNED_LONG * data, UNSIGNED_CHARACTER depth)`

6.4.3.3 `UNSIGNED_LONG* CMD::MDQuote::getAskPrice ()` [inline]

6.4.3.4 `UNSIGNED_LONG* CMD::MDQuote::getAskQty ()` [inline]

6.4.3.5 `UNSIGNED_LONG* CMD::MDQuote::getBidPrice ()` [inline]

6.4.3.6 `UNSIGNED_LONG* CMD::MDQuote::getBidQty ()` [inline]

6.4.3.7 `UNSIGNED_LONG CMD::MDQuote::getClosePrice ()` [inline]

6.4.3.8 `UNSIGNED_CHARACTER CMD::MDQuote::getDepth ()` [inline]

6.4.3.9 `UNSIGNED_LONG CMD::MDQuote::getHighPrice ()` [inline]

6.4.3.10 `UNSIGNED_LONG CMD::MDQuote::getLastTradePrice ()` [inline]

6.4.3.11 `UNSIGNED_LONG CMD::MDQuote::getLastTradeQty ()` [inline]

6.4.3.12 `UNSIGNED_LONG CMD::MDQuote::getLowerCktLimit ()` [inline]

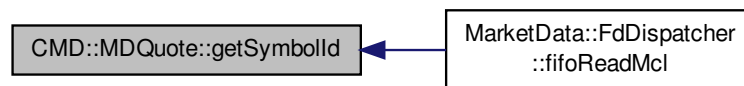
6.4.3.13 `UNSIGNED_LONG CMD::MDQuote::getLowPrice () [inline]`

6.4.3.14 `UNSIGNED_LONG CMD::MDQuote::getNumberOfTrades () [inline]`

6.4.3.15 `UNSIGNED_LONG CMD::MDQuote::getOpenPrice () [inline]`

6.4.3.16 `UNSIGNED_LONG CMD::MDQuote::getSymbolId () [inline]`

Here is the caller graph for this function:



6.4.3.17 `UNSIGNED_LONG CMD::MDQuote::getTotalAskQty () [inline]`

6.4.3.18 `UNSIGNED_LONG CMD::MDQuote::getTotalBidQty () [inline]`

6.4.3.19 `UNSIGNED_LONG CMD::MDQuote::getUpperCktLimit () [inline]`

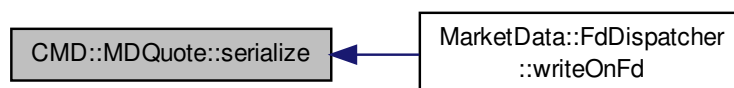
6.4.3.20 `UNSIGNED_LONG CMD::MDQuote::getValue () [inline]`

6.4.3.21 `UNSIGNED_LONG CMD::MDQuote::getVolume () [inline]`

6.4.3.22 `void CMD::MDQuote::initialize ()`

6.4.3.23 `int CMD::MDQuote::serialize (char * buf)`

Here is the caller graph for this function:



6.4.3.24 `void CMD::MDQuote::setAskPrice (UNSIGNED_LONG val[]) [inline]`

6.4.3.25 `void CMD::MDQuote::setAskQty (UNSIGNED_LONG val[]) [inline]`

6.4.3.26 `void CMD::MDQuote::setBidPrice (UNSIGNED_LONG val[]) [inline]`

- 6.4.3.27 void CMD::MDQuote::setBidQty (UNSIGNED_LONG val) [inline]
- 6.4.3.28 void CMD::MDQuote::setClosePrice (UNSIGNED_LONG val) [inline]
- 6.4.3.29 void CMD::MDQuote::setDepth (UNSIGNED_CHARACTER val) [inline]
- 6.4.3.30 void CMD::MDQuote::setHighPrice (UNSIGNED_LONG val) [inline]
- 6.4.3.31 void CMD::MDQuote::setLastTradePrice (UNSIGNED_LONG val) [inline]
- 6.4.3.32 void CMD::MDQuote::setLastTradeQty (UNSIGNED_LONG val) [inline]
- 6.4.3.33 void CMD::MDQuote::setLowerCktLimit (UNSIGNED_LONG val) [inline]
- 6.4.3.34 void CMD::MDQuote::setLowPrice (UNSIGNED_LONG val) [inline]
- 6.4.3.35 void CMD::MDQuote::setNummberOfTrades (UNSIGNED_LONG val) [inline]
- 6.4.3.36 void CMD::MDQuote::setOpenPrice (UNSIGNED_LONG val) [inline]
- 6.4.3.37 void CMD::MDQuote::setSymbolId (UNSIGNED_LONG val) [inline]
- 6.4.3.38 void CMD::MDQuote::setTotalAskQty (UNSIGNED_LONG val) [inline]
- 6.4.3.39 void CMD::MDQuote::setTotalBidQty (UNSIGNED_LONG val) [inline]
- 6.4.3.40 void CMD::MDQuote::setUpperCktLimit (UNSIGNED_LONG val) [inline]
- 6.4.3.41 void CMD::MDQuote::setValue (UNSIGNED_LONG val) [inline]
- 6.4.3.42 void CMD::MDQuote::setVolume (UNSIGNED_LONG val) [inline]

6.4.4 Member Data Documentation

- 6.4.4.1 UNSIGNED_LONG CMD::MDQuote::_askPrice[MAX_LOOKUP_LEVEL] [private]
- 6.4.4.2 UNSIGNED_LONG CMD::MDQuote::_askQty[MAX_LOOKUP_LEVEL] [private]
- 6.4.4.3 UNSIGNED_LONG CMD::MDQuote::_bidPrice[MAX_LOOKUP_LEVEL] [private]
- 6.4.4.4 UNSIGNED_LONG CMD::MDQuote::_bidQty[MAX_LOOKUP_LEVEL] [private]
- 6.4.4.5 UNSIGNED_LONG CMD::MDQuote::_closePrice [private]
- 6.4.4.6 UNSIGNED_CHARACTER CMD::MDQuote::_depth [private]
- 6.4.4.7 UNSIGNED_LONG CMD::MDQuote::_highPrice [private]
- 6.4.4.8 UNSIGNED_LONG CMD::MDQuote::_lastTradePrice [private]
- 6.4.4.9 UNSIGNED_LONG CMD::MDQuote::_lastTradeQty [private]
- 6.4.4.10 UNSIGNED_LONG CMD::MDQuote::_lowerCktLimit [private]
- 6.4.4.11 UNSIGNED_LONG CMD::MDQuote::_lowPrice [private]

6.4.4.12 **UNSIGNED_LONG** CMD::MDQuote::_numberOfTrades [private]

6.4.4.13 **UNSIGNED_LONG** CMD::MDQuote::_openPrice [private]

6.4.4.14 **UNSIGNED_LONG** CMD::MDQuote::_symbolId [private]

6.4.4.15 **UNSIGNED_LONG** CMD::MDQuote::_totalAskQty [private]

6.4.4.16 **UNSIGNED_LONG** CMD::MDQuote::_totalBidQty [private]

6.4.4.17 **UNSIGNED_LONG** CMD::MDQuote::_upperCktLimit [private]

6.4.4.18 **UNSIGNED_LONG** CMD::MDQuote::_value [private]

6.4.4.19 **UNSIGNED_LONG** CMD::MDQuote::_volume [private]

6.4.4.20 **const int** CMD::MDQuote::MAX_LOOKUP_LEVEL = 10 [static], [private]

6.4.4.21 **const int** CMD::MDQuote::QUOTEDATA_LENGTH = 80 [static], [private]

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

- [commands.h](#)

6.5 CMD::MDQuoteConsolidated Class Reference

[MarketData](#) Quote Consolidated.

```
#include <commands.h>
```


Collaboration diagram for CMD::MDQuoteConsolidated:

CMD::MDQuoteConsolidated
<ul style="list-style-type: none"> - _numSymbols - _symbolId - _lastTradePrice - _lastTradeQty - _openPrice - _closePrice - _highPrice - _lowPrice - _depth - _bidPrice - _bidQty - _askPrice - _askQty - MAX_LOOKUP_LEVEL - QUOTEDATA_LENGTH
<ul style="list-style-type: none"> + MDQuoteConsolidated() + MDQuoteConsolidated() + dumpDepth() + dump() + serialize() + initialize() + setNumberOfSymbols() + setSymbolId() + setLastTradePrice() + setLastTradeQty() and 22 more...

Public Member Functions

- [MDQuoteConsolidated](#) ()
- [MDQuoteConsolidated](#) (const char *buf)
- void [dumpDepth](#) (std::vector< [UNSIGNED_LONG](#) > &data, [UNSIGNED_CHARACTER](#) depth)
- void [dump](#) ()
- int [serialize](#) (char *buf)
- void [initialize](#) ()
- void [setNumberOfSymbols](#) ([UNSIGNED_SHORT](#) val)
- void [setSymbolId](#) ([UNSIGNED_SHORT](#) index, [UNSIGNED_LONG](#) val)
- void [setLastTradePrice](#) ([UNSIGNED_SHORT](#) index, [UNSIGNED_LONG](#) val)
- void [setLastTradeQty](#) ([UNSIGNED_SHORT](#) index, [UNSIGNED_LONG](#) val)
- void [setOpenPrice](#) ([UNSIGNED_SHORT](#) index, [UNSIGNED_LONG](#) val)
- void [setClosePrice](#) ([UNSIGNED_SHORT](#) index, [UNSIGNED_LONG](#) val)
- void [setHighPrice](#) ([UNSIGNED_SHORT](#) index, [UNSIGNED_LONG](#) val)

- void [setLowPrice](#) (UNSIGNED_SHORT index, UNSIGNED_LONG val)
- void [setDepth](#) (UNSIGNED_CHARACTER val)
- void [setBidPrice](#) (UNSIGNED_SHORT index, UNSIGNED_LONG val[])
- void [setBidQty](#) (UNSIGNED_SHORT index, UNSIGNED_LONG val[])
- void [setAskPrice](#) (UNSIGNED_SHORT index, UNSIGNED_LONG val[])
- void [setAskQty](#) (UNSIGNED_SHORT index, UNSIGNED_LONG val[])
- UNSIGNED_SHORT [getNumberOfSymbols](#) ()
- UNSIGNED_LONG [getSymbolId](#) (UNSIGNED_SHORT index)
- UNSIGNED_LONG [getLastTradePrice](#) (UNSIGNED_SHORT index)
- UNSIGNED_LONG [getLastTradeQty](#) (UNSIGNED_SHORT index)
- UNSIGNED_LONG [getOpenPrice](#) (UNSIGNED_SHORT index)
- UNSIGNED_LONG [getClosePrice](#) (UNSIGNED_SHORT index)
- UNSIGNED_LONG [getHighPrice](#) (UNSIGNED_SHORT index)
- UNSIGNED_LONG [getLowPrice](#) (UNSIGNED_SHORT index)
- UNSIGNED_CHARACTER [getDepth](#) ()
- std::vector< UNSIGNED_LONG > & [getBidPrice](#) (UNSIGNED_SHORT index)
- std::vector< UNSIGNED_LONG > & [getBidQty](#) (UNSIGNED_SHORT index)
- std::vector< UNSIGNED_LONG > & [getAskPrice](#) (UNSIGNED_SHORT index)
- std::vector< UNSIGNED_LONG > & [getAskQty](#) (UNSIGNED_SHORT index)

Private Attributes

- UNSIGNED_SHORT [_numSymbols](#)
- std::vector< UNSIGNED_LONG > [_symbolId](#)
- std::vector< UNSIGNED_LONG > [_lastTradePrice](#)
- std::vector< UNSIGNED_LONG > [_lastTradeQty](#)
- std::vector< UNSIGNED_LONG > [_openPrice](#)
- std::vector< UNSIGNED_LONG > [_closePrice](#)
- std::vector< UNSIGNED_LONG > [_highPrice](#)
- std::vector< UNSIGNED_LONG > [_lowPrice](#)
- UNSIGNED_CHARACTER [_depth](#)
- std::vector< std::vector
< UNSIGNED_LONG > > [_bidPrice](#)
- std::vector< std::vector
< UNSIGNED_LONG > > [_bidQty](#)
- std::vector< std::vector
< UNSIGNED_LONG > > [_askPrice](#)
- std::vector< std::vector
< UNSIGNED_LONG > > [_askQty](#)

Static Private Attributes

- static const int [MAX_LOOKUP_LEVEL](#) = 10
- static const int [QUOTEDATA_LENGTH](#) = 80

6.5.1 Detailed Description

[MarketData](#) Quote Consolidated.

Server will send the consolidated quote to the subscribed members.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 CMD::MDQuoteConsolidated::MDQuoteConsolidated ()

6.5.2.2 CMD::MDQuoteConsolidated::MDQuoteConsolidated (const char * *buf*)

6.5.3 Member Function Documentation

6.5.3.1 void CMD::MDQuoteConsolidated::dump ()

6.5.3.2 void CMD::MDQuoteConsolidated::dumpDepth (std::vector< UNSIGNED_LONG > & *data*,
UNSIGNED_CHARACTER *depth*)

6.5.3.3 std::vector<UNSIGNED_LONG>& CMD::MDQuoteConsolidated::getAskPrice (UNSIGNED_SHORT *index*)
[inline]

6.5.3.4 std::vector<UNSIGNED_LONG>& CMD::MDQuoteConsolidated::getAskQty (UNSIGNED_SHORT *index*)
[inline]

6.5.3.5 std::vector<UNSIGNED_LONG>& CMD::MDQuoteConsolidated::getBidPrice (UNSIGNED_SHORT *index*)
[inline]

6.5.3.6 std::vector<UNSIGNED_LONG>& CMD::MDQuoteConsolidated::getBidQty (UNSIGNED_SHORT *index*)
[inline]

6.5.3.7 UNSIGNED_LONG CMD::MDQuoteConsolidated::getClosePrice (UNSIGNED_SHORT *index*) [inline]

6.5.3.8 UNSIGNED_CHARACTER CMD::MDQuoteConsolidated::getDepth () [inline]

6.5.3.9 UNSIGNED_LONG CMD::MDQuoteConsolidated::getHighPrice (UNSIGNED_SHORT *index*) [inline]

6.5.3.10 UNSIGNED_LONG CMD::MDQuoteConsolidated::getLastTradePrice (UNSIGNED_SHORT *index*)
[inline]

6.5.3.11 UNSIGNED_LONG CMD::MDQuoteConsolidated::getLastTradeQty (UNSIGNED_SHORT *index*)
[inline]

6.5.3.12 UNSIGNED_LONG CMD::MDQuoteConsolidated::getLowPrice (UNSIGNED_SHORT *index*) [inline]

6.5.3.13 UNSIGNED_SHORT CMD::MDQuoteConsolidated::getNumberOfSymbols () [inline]

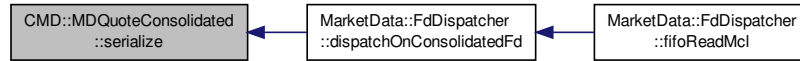
6.5.3.14 UNSIGNED_LONG CMD::MDQuoteConsolidated::getOpenPrice (UNSIGNED_SHORT *index*) [inline]

6.5.3.15 UNSIGNED_LONG CMD::MDQuoteConsolidated::getSymbolId (UNSIGNED_SHORT *index*) [inline]

6.5.3.16 void CMD::MDQuoteConsolidated::initialize ()

6.5.3.17 `int CMD::MDQuoteConsolidated::serialize (char * buf)`

Here is the caller graph for this function:



6.5.3.18 `void CMD::MDQuoteConsolidated::setAskPrice (UNSIGNED_SHORT index, UNSIGNED_LONG val[])` [inline]

Here is the caller graph for this function:



6.5.3.19 `void CMD::MDQuoteConsolidated::setAskQty (UNSIGNED_SHORT index, UNSIGNED_LONG val[])` [inline]

Here is the caller graph for this function:



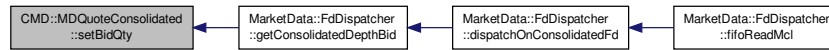
6.5.3.20 `void CMD::MDQuoteConsolidated::setBidPrice (UNSIGNED_SHORT index, UNSIGNED_LONG val[])` [inline]

Here is the caller graph for this function:



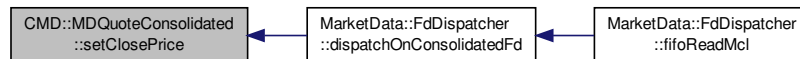
6.5.3.21 void CMD::MDQuoteConsolidated::setBidQty (UNSIGNED_SHORT index, UNSIGNED_LONG val)
[inline]

Here is the caller graph for this function:



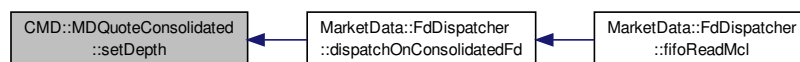
6.5.3.22 void CMD::MDQuoteConsolidated::setClosePrice (UNSIGNED_SHORT index, UNSIGNED_LONG val)
[inline]

Here is the caller graph for this function:



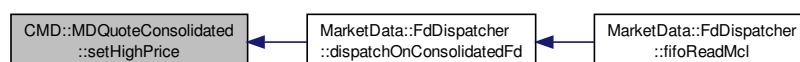
6.5.3.23 void CMD::MDQuoteConsolidated::setDepth (UNSIGNED_CHARACTER val) [inline]

Here is the caller graph for this function:



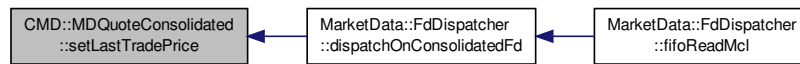
6.5.3.24 void CMD::MDQuoteConsolidated::setHighPrice (UNSIGNED_SHORT index, UNSIGNED_LONG val)
[inline]

Here is the caller graph for this function:



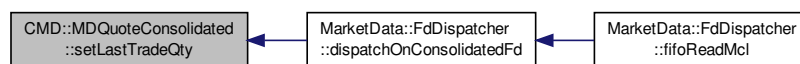
6.5.3.25 `void CMD::MDQuoteConsolidated::setLastTradePrice (UNSIGNED_SHORT index, UNSIGNED_LONG val)`
`[inline]`

Here is the caller graph for this function:



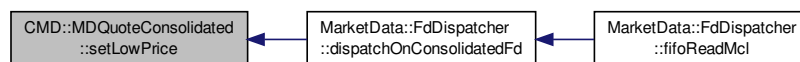
6.5.3.26 `void CMD::MDQuoteConsolidated::setLastTradeQty (UNSIGNED_SHORT index, UNSIGNED_LONG val)`
`[inline]`

Here is the caller graph for this function:



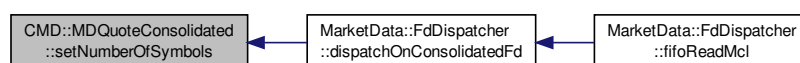
6.5.3.27 `void CMD::MDQuoteConsolidated::setLowPrice (UNSIGNED_SHORT index, UNSIGNED_LONG val)`
`[inline]`

Here is the caller graph for this function:



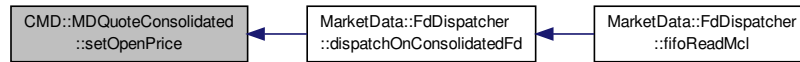
6.5.3.28 `void CMD::MDQuoteConsolidated::setNumberOfSymbols (UNSIGNED_SHORT val)` `[inline]`

Here is the caller graph for this function:



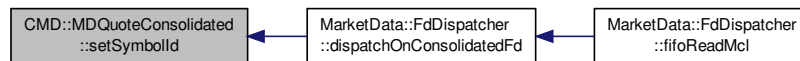
6.5.3.29 `void CMD::MDQuoteConsolidated::setOpenPrice (UNSIGNED_SHORT index, UNSIGNED_LONG val)`
`[inline]`

Here is the caller graph for this function:



6.5.3.30 `void CMD::MDQuoteConsolidated::setSymbolId (UNSIGNED_SHORT index, UNSIGNED_LONG val)`
`[inline]`

Here is the caller graph for this function:



6.5.4 Member Data Documentation

6.5.4.1 `std::vector< std::vector<UNSIGNED_LONG> > CMD::MDQuoteConsolidated::_askPrice` `[private]`

6.5.4.2 `std::vector< std::vector<UNSIGNED_LONG> > CMD::MDQuoteConsolidated::_askQty` `[private]`

6.5.4.3 `std::vector< std::vector<UNSIGNED_LONG> > CMD::MDQuoteConsolidated::_bidPrice` `[private]`

6.5.4.4 `std::vector< std::vector<UNSIGNED_LONG> > CMD::MDQuoteConsolidated::_bidQty` `[private]`

6.5.4.5 `std::vector<UNSIGNED_LONG> CMD::MDQuoteConsolidated::_closePrice` `[private]`

6.5.4.6 `UNSIGNED_CHARACTER CMD::MDQuoteConsolidated::_depth` `[private]`

6.5.4.7 `std::vector<UNSIGNED_LONG> CMD::MDQuoteConsolidated::_highPrice` `[private]`

6.5.4.8 `std::vector<UNSIGNED_LONG> CMD::MDQuoteConsolidated::_lastTradePrice` `[private]`

6.5.4.9 `std::vector<UNSIGNED_LONG> CMD::MDQuoteConsolidated::_lastTradeQty` `[private]`

6.5.4.10 `std::vector<UNSIGNED_LONG> CMD::MDQuoteConsolidated::_lowPrice` `[private]`

6.5.4.11 `UNSIGNED_SHORT CMD::MDQuoteConsolidated::_numSymbols` `[private]`

6.5.4.12 `std::vector<UNSIGNED_LONG> CMD::MDQuoteConsolidated::_openPrice` `[private]`

6.5.4.13 `std::vector<UNSIGNED_LONG> CMD::MDQuoteConsolidated::_symbolId` `[private]`

6.5.4.14 `const int CMD::MDQuoteConsolidated::MAX_LOOKUP_LEVEL = 10` `[static], [private]`

6.5.4.15 `const int CMD::MDQuoteConsolidated::QUOTEDATA_LENGTH = 80` `[static], [private]`

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

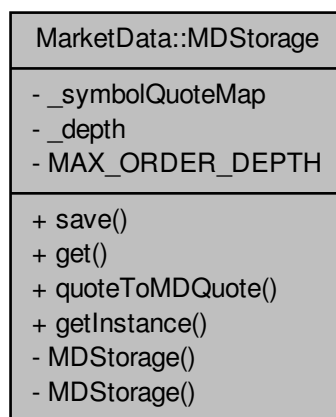
- [commands.h](#)

6.6 MarketData::MDStorage Class Reference

[MDStorage](#) Class.

```
#include <mdStorage.h>
```

Collaboration diagram for MarketData::MDStorage:



Public Member Functions

- void [save](#) (const long, const [CMD::MDQuote](#) *)
Function to save market quote for every symbol.
- [CMD::MDQuote](#) [get](#) (const long symbolId)
Function to get market quote for given symbol.
- [CMD::MDQuote](#) * [quoteToMDQuote](#) (const long symbolId, const [CMD::MDQuote](#) *quote)

Static Public Member Functions

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

Private Types

- typedef boost::unordered_map
 < long, [CMD::MDQuote](#) * > [SymbolQuoteMap](#)

Private Member Functions

- [MDStorage](#) ()
- [MDStorage](#) (const [MDStorage](#) &)

Private Attributes

- [SymbolQuoteMap](#) _symbolQuoteMap
- int _depth

Static Private Attributes

- static const int [MAX_ORDER_DEPTH](#) = 5

6.6.1 Detailed Description

[MDStorage](#) Class.

This class is responsible for storing the quote. This is useful for getting consolidated feed for multiple symbols.

6.6.2 Member Typedef Documentation

6.6.2.1 `typedef boost::unordered_map< long, CMD::MDQuote* > MarketData::MDStorage::SymbolQuoteMap`
[private]

6.6.3 Constructor & Destructor Documentation

6.6.3.1 `MarketData::MDStorage::MDStorage ()` [inline], [private]

6.6.3.2 `MarketData::MDStorage::MDStorage (const MDStorage &)` [private]

6.6.4 Member Function Documentation

6.6.4.1 `CMD::MDQuote MarketData::MDStorage::get (const long symbolId)`

Function to get market quote for given symbol.

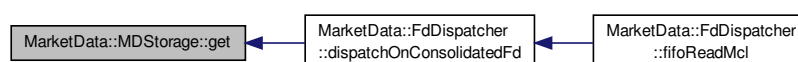
Parameters

<i>symbol</i>	Id
---------------	----

Returns

MDQuote

Here is the caller graph for this function:



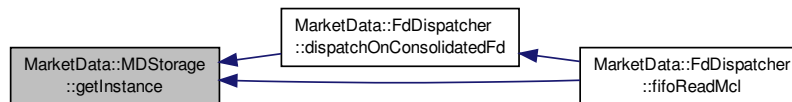
6.6.4.2 `static MDStorage* MarketData::MDStorage::getInstance () [inline],[static]`

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

Returns

MDStorage*

Here is the caller graph for this function:



6.6.4.3 `CMD::MDQuote* MarketData::MDStorage::quoteToMDQuote (const long symbolId, const CMD::MDQuote * quote)`

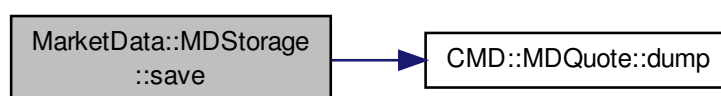
6.6.4.4 `void MarketData::MDStorage::save (const long symbolId, const CMD::MDQuote * quote)`

Function to save market quote for every symbol.

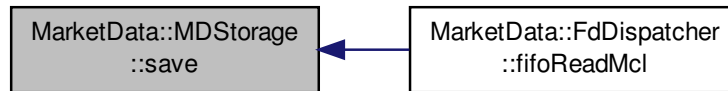
Parameters

<i>symbol</i>	Id
<i>MDQuote</i>	

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.5 Member Data Documentation

6.6.5.1 `int MarketData::MDStorage::depth` `[private]`

6.6.5.2 `SymbolQuoteMap MarketData::MDStorage::symbolQuoteMap` `[private]`

6.6.5.3 `const int MarketData::MDStorage::MAX_ORDER_DEPTH = 5` `[static], [private]`

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

- [mdStorage.h](#)
- [mdStorage.cpp](#)

6.7 CMD::MDSubscribeRequest Class Reference

[MarketData](#) Subscribe/Unsubscribe Request.

```
#include <commands.h>
```

Collaboration diagram for CMD::MDSubscribeRequest:

CMD::MDSubscribeRequest
<ul style="list-style-type: none"> - <code>_isSubscription</code> - <code>_symbolId</code> - <code>_userId</code> - <code>TOKEN_LENGTH</code>
<ul style="list-style-type: none"> + <code>MDSubscribeRequest()</code> + <code>MDSubscribeRequest()</code> + <code>serialize()</code> + <code>initialize()</code> + <code>getSymbolId()</code> + <code>setSymbolId()</code> + <code>isSubscriptionReq()</code> + <code>setSubscriptionReq()</code> + <code>getUserId()</code> + <code>setUserId()</code>

Public Member Functions

- [MDSubscribeRequest](#) ()
- [MDSubscribeRequest](#) (const char *buf)
- int [serialize](#) (char *buf)
- void [initialize](#) ()
- [UNSIGNED_LONG](#) [getSymbolId](#) () const
- void [setSymbolId](#) ([UNSIGNED_LONG](#) val)
- [UNSIGNED_CHARACTER](#) [isSubscriptionReq](#) () const
- void [setSubscriptionReq](#) ([UNSIGNED_CHARACTER](#) val)
- [UNSIGNED_SHORT](#) [getUserId](#) () const
- void [setUserId](#) ([UNSIGNED_SHORT](#) val)

Private Attributes

- [UNSIGNED_SHORT](#) `_isSubscription`
- [UNSIGNED_LONG](#) `_symbolId`
- [UNSIGNED_SHORT](#) `_userId`

Static Private Attributes

- static const int [TOKEN_LENGTH](#) = 100

6.7.1 Detailed Description

[MarketData](#) Subscribe/Unsubscribe Request.

Server will send a successful response if the subscription was successful, otherwise it send a unsuccessful response.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 `CMD::MDSubscribeRequest::MDSubscribeRequest ()` `[inline]`

6.7.2.2 `CMD::MDSubscribeRequest::MDSubscribeRequest (const char * buf)`

6.7.3 Member Function Documentation

6.7.3.1 `UNSIGNED_LONG CMD::MDSubscribeRequest::getSymbolId () const` `[inline]`

6.7.3.2 `UNSIGNED_SHORT CMD::MDSubscribeRequest::getUserId () const` `[inline]`

6.7.3.3 `void CMD::MDSubscribeRequest::initialize ()`

6.7.3.4 `UNSIGNED_CHARACTER CMD::MDSubscribeRequest::isSubscriptionReq () const` `[inline]`

6.7.3.5 `int CMD::MDSubscribeRequest::serialize (char * buf)`

6.7.3.6 `void CMD::MDSubscribeRequest::setSubscriptionReq (UNSIGNED_CHARACTER val)` `[inline]`

6.7.3.7 `void CMD::MDSubscribeRequest::setSymbolId (UNSIGNED_LONG val)` `[inline]`

6.7.3.8 `void CMD::MDSubscribeRequest::setUserId (UNSIGNED_SHORT val)` `[inline]`

6.7.4 Member Data Documentation

6.7.4.1 `UNSIGNED_SHORT CMD::MDSubscribeRequest::_isSubscription` `[private]`

6.7.4.2 `UNSIGNED_LONG CMD::MDSubscribeRequest::_symbolId` `[private]`

6.7.4.3 `UNSIGNED_SHORT CMD::MDSubscribeRequest::_userId` `[private]`

6.7.4.4 `const int CMD::MDSubscribeRequest::TOKEN_LENGTH = 100` `[static], [private]`

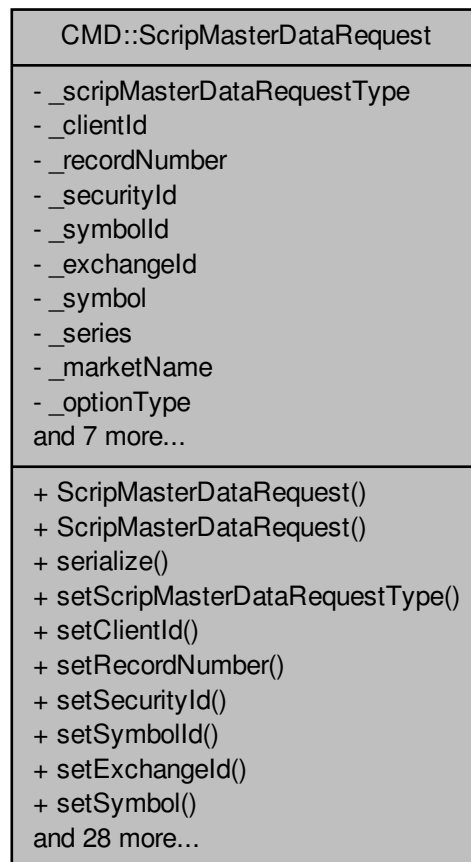
The documentation for this class was generated from the following file:

- [commands.h](#)

6.8 CMD::ScripMasterDataRequest Class Reference

```
#include <commands.h>
```

Collaboration diagram for CMD::ScripMasterDataRequest:



Public Member Functions

- [ScripMasterDataRequest](#) ()
- [ScripMasterDataRequest](#) (const char *buf)
- int [serialize](#) (char *buf)
- void [setScripMasterDataRequestType](#) (UNSIGNED_CHARACTER scripMasterDataRequestType)
- void [setClientId](#) (UNSIGNED_INTEGER clientId)
- void [setRecordNumber](#) (UNSIGNED_LONG recordNumber)
- void [setSecurityId](#) (UNSIGNED_LONG securityId)
- void [setSymbolId](#) (UNSIGNED_LONG symbolId)
- void [setExchangeId](#) (UNSIGNED_SHORT exchangeId)
- void [setSymbol](#) (const char *symbol)
- void [setSeries](#) (const char *series)
- void [setMarketName](#) (const char *marketName)
- void [setSymbolAlias](#) (const char *symbolAlias)
- void [setOptionType](#) (UNSIGNED_CHARACTER optionType)
- void [setOptionMode](#) (UNSIGNED_CHARACTER optionMode)
- void [setSecurityType](#) (UNSIGNED_CHARACTER securityType)
- void [setStrikePrice](#) (UNSIGNED_LONG strikePrice)

- void `setExpiryYearMon` (UNSIGNED_INTEGER expiryYearMon)
- void `setExpiryDate` (UNSIGNED_INTEGER expiryDate)
- void `setNumberOfRecords` (UNSIGNED_INTEGER numberOfRecords)
- UNSIGNED_CHARACTER `getScripMasterDataRequestType` ()
- UNSIGNED_INTEGER `getClientId` ()
- UNSIGNED_LONG `getRecordNumber` ()
- UNSIGNED_LONG `getSecurityId` ()
- UNSIGNED_LONG `getSymbolId` ()
- UNSIGNED_SHORT `getExchangeld` ()
- char * `getSymbol` ()
- char * `getSeries` ()
- char * `getMarketName` ()
- char * `getSymbolAlias` ()
- UNSIGNED_CHARACTER `getOptionType` ()
- UNSIGNED_CHARACTER `getOptionMode` ()
- UNSIGNED_CHARACTER `getSecurityType` ()
- UNSIGNED_LONG `getStrikePrice` ()
- UNSIGNED_INTEGER `getExpiryYearMon` ()
- UNSIGNED_INTEGER `getExpiryDate` ()
- UNSIGNED_INTEGER `getNumberOfRecords` ()
- void `dump` ()

Private Attributes

- UNSIGNED_CHARACTER `_scripMasterDataRequestType`
- UNSIGNED_INTEGER `_clientId`
- UNSIGNED_LONG `_recordNumber`
- UNSIGNED_LONG `_securityId`
- UNSIGNED_LONG `_symbolId`
- UNSIGNED_SHORT `_exchangeld`
- char `_symbol` [SYMBOL_SIZE]
- char `_series` [SERIES_SIZE]
- char `_marketName` [MARKET_NAME_SIZE]
- UNSIGNED_CHARACTER `_optionType`
- UNSIGNED_CHARACTER `_optionMode`
- UNSIGNED_CHARACTER `_securityType`
- UNSIGNED_LONG `_strikePrice`
- UNSIGNED_INTEGER `_expiryYearMon`
- UNSIGNED_INTEGER `_expiryDate`
- UNSIGNED_INTEGER `_numberOfRecords`
- char `_symbolAlias` [SYMBOL_ALIAS_SIZE]

6.8.1 Constructor & Destructor Documentation

6.8.1.1 CMD::ScripMasterDataRequest::ScripMasterDataRequest () [inline]

6.8.1.2 CMD::ScripMasterDataRequest::ScripMasterDataRequest (const char * *buf*)

6.8.2 Member Function Documentation

6.8.2.1 void CMD::ScripMasterDataRequest::dump ()

6.8.2.2 UNSIGNED_INTEGER CMD::ScripMasterDataRequest::getClientId () [inline]

- 6.8.2.3 **UNSIGNED_SHORT** **CMD::ScripMasterDataRequest::getExchanged**() [inline]
- 6.8.2.4 **UNSIGNED_INTEGER** **CMD::ScripMasterDataRequest::getExpiryDate**() [inline]
- 6.8.2.5 **UNSIGNED_INTEGER** **CMD::ScripMasterDataRequest::getExpiryYearMon**() [inline]
- 6.8.2.6 **char*** **CMD::ScripMasterDataRequest::getMarketName**() [inline]
- 6.8.2.7 **UNSIGNED_INTEGER** **CMD::ScripMasterDataRequest::getNumberOfRecords**() [inline]
- 6.8.2.8 **UNSIGNED_CHARACTER** **CMD::ScripMasterDataRequest::getOptionMode**() [inline]
- 6.8.2.9 **UNSIGNED_CHARACTER** **CMD::ScripMasterDataRequest::getOptionType**() [inline]
- 6.8.2.10 **UNSIGNED_LONG** **CMD::ScripMasterDataRequest::getRecordNumber**() [inline]
- 6.8.2.11 **UNSIGNED_CHARACTER** **CMD::ScripMasterDataRequest::getScripMasterDataRequestType**() [inline]
- 6.8.2.12 **UNSIGNED_LONG** **CMD::ScripMasterDataRequest::getSecurityId**() [inline]
- 6.8.2.13 **UNSIGNED_CHARACTER** **CMD::ScripMasterDataRequest::getSecurityType**() [inline]
- 6.8.2.14 **char*** **CMD::ScripMasterDataRequest::getSeries**() [inline]
- 6.8.2.15 **UNSIGNED_LONG** **CMD::ScripMasterDataRequest::getStrikePrice**() [inline]
- 6.8.2.16 **char*** **CMD::ScripMasterDataRequest::getSymbol**() [inline]
- 6.8.2.17 **char*** **CMD::ScripMasterDataRequest::getSymbolAlias**() [inline]
- 6.8.2.18 **UNSIGNED_LONG** **CMD::ScripMasterDataRequest::getSymbolId**() [inline]
- 6.8.2.19 **int** **CMD::ScripMasterDataRequest::serialize**(**char * buf**)
- 6.8.2.20 **void** **CMD::ScripMasterDataRequest::setClientId**(**UNSIGNED_INTEGER clientId**) [inline]
- 6.8.2.21 **void** **CMD::ScripMasterDataRequest::setExchanged**(**UNSIGNED_SHORT exchanged**) [inline]
- 6.8.2.22 **void** **CMD::ScripMasterDataRequest::setExpiryDate**(**UNSIGNED_INTEGER expiryDate**) [inline]
- 6.8.2.23 **void** **CMD::ScripMasterDataRequest::setExpiryYearMon**(**UNSIGNED_INTEGER expiryYearMon**) [inline]
- 6.8.2.24 **void** **CMD::ScripMasterDataRequest::setMarketName**(**const char * marketName**) [inline]
- 6.8.2.25 **void** **CMD::ScripMasterDataRequest::setNumberOfRecords**(**UNSIGNED_INTEGER numberOfRecords**) [inline]
- 6.8.2.26 **void** **CMD::ScripMasterDataRequest::setOptionMode**(**UNSIGNED_CHARACTER optionMode**) [inline]
- 6.8.2.27 **void** **CMD::ScripMasterDataRequest::setOptionType**(**UNSIGNED_CHARACTER optionType**) [inline]
- 6.8.2.28 **void** **CMD::ScripMasterDataRequest::setRecordNumber**(**UNSIGNED_LONG recordNumber**) [inline]
- 6.8.2.29 **void** **CMD::ScripMasterDataRequest::setScripMasterDataRequestType**(**UNSIGNED_CHARACTER scripMasterDataRequestType**) [inline]

- 6.8.2.30 void CMD::ScripMasterDataRequest::setSecurityId (UNSIGNED_LONG *securityId*) [inline]
- 6.8.2.31 void CMD::ScripMasterDataRequest::setSecurityType (UNSIGNED_CHARACTER *securityType*) [inline]
- 6.8.2.32 void CMD::ScripMasterDataRequest::setSeries (const char * *series*) [inline]
- 6.8.2.33 void CMD::ScripMasterDataRequest::setStrikePrice (UNSIGNED_LONG *strikePrice*) [inline]
- 6.8.2.34 void CMD::ScripMasterDataRequest::setSymbol (const char * *symbol*) [inline]
- 6.8.2.35 void CMD::ScripMasterDataRequest::setSymbolAlias (const char * *symbolAlias*) [inline]
- 6.8.2.36 void CMD::ScripMasterDataRequest::setSymbolId (UNSIGNED_LONG *symbolId*) [inline]

6.8.3 Member Data Documentation

- 6.8.3.1 UNSIGNED_INTEGER CMD::ScripMasterDataRequest::_clientId [private]
- 6.8.3.2 UNSIGNED_SHORT CMD::ScripMasterDataRequest::_exchangeId [private]
- 6.8.3.3 UNSIGNED_INTEGER CMD::ScripMasterDataRequest::_expiryDate [private]
- 6.8.3.4 UNSIGNED_INTEGER CMD::ScripMasterDataRequest::_expiryYearMon [private]
- 6.8.3.5 char CMD::ScripMasterDataRequest::_marketName[MARKET_NAME_SIZE] [private]
- 6.8.3.6 UNSIGNED_INTEGER CMD::ScripMasterDataRequest::_numberOfRecords [private]
- 6.8.3.7 UNSIGNED_CHARACTER CMD::ScripMasterDataRequest::_optionMode [private]
- 6.8.3.8 UNSIGNED_CHARACTER CMD::ScripMasterDataRequest::_optionType [private]
- 6.8.3.9 UNSIGNED_LONG CMD::ScripMasterDataRequest::_recordNumber [private]
- 6.8.3.10 UNSIGNED_CHARACTER CMD::ScripMasterDataRequest::_scripMasterDataRequestType [private]
- 6.8.3.11 UNSIGNED_LONG CMD::ScripMasterDataRequest::_securityId [private]
- 6.8.3.12 UNSIGNED_CHARACTER CMD::ScripMasterDataRequest::_securityType [private]
- 6.8.3.13 char CMD::ScripMasterDataRequest::_series[SERIES_SIZE] [private]
- 6.8.3.14 UNSIGNED_LONG CMD::ScripMasterDataRequest::_strikePrice [private]
- 6.8.3.15 char CMD::ScripMasterDataRequest::_symbol[SYMBOL_SIZE] [private]
- 6.8.3.16 char CMD::ScripMasterDataRequest::_symbolAlias[SYMBOL_ALIAS_SIZE] [private]
- 6.8.3.17 UNSIGNED_LONG CMD::ScripMasterDataRequest::_symbolId [private]

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

- [commands.h](#)

6.9 Subscription Class Reference

It accepts request for symbol subscription from the client and updates symbol-fds map. Which is used by disppatcher to send event for every tick update.

Collaboration diagram for Subscription:



6.9.1 Detailed Description

It accepts request for symbol subscription from the client and updates symbol-fds map. Which is used by disppatcher to send event for every tick update.

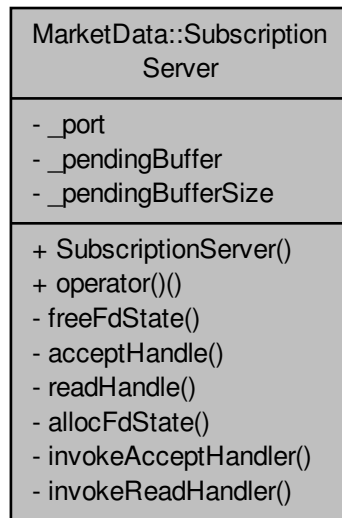
The documentation for this class was generated from the following file:

- [subscriptionServer.h](#)

6.10 MarketData::SubscriptionServer Class Reference

```
#include <subscriptionServer.h>
```

Collaboration diagram for MarketData::SubscriptionServer:



Public Member Functions

- [SubscriptionServer](#) (int)
- void [operator](#)() ()
overloaded function operator.

Private Member Functions

- void [freeFdState](#) ([FdState](#) *state)
Free allocated state memory.
- void [acceptHandle](#) (evutil_socket_t listener, short event, event_base *base)
Accept Handle.
- void [readHandle](#) (evutil_socket_t fd, short events, [FdState](#) *state)
Read Handle.
- [FdState](#) * [allocFdState](#) (struct event_base *base, evutil_socket_t fd)
Allocate [FdState](#).

Static Private Member Functions

- static void [invokeAcceptHandler](#) (evutil_socket_t fd, short events, void *arg)
Invoke Accept Handler Since libevent is a C library, it doesn't directly provide a mechanism to associate an instance and an instance method, so we need to do it ourself.
- static void [invokeReadHandler](#) (evutil_socket_t fd, short events, void *arg)
Invoke Read Handler Since libevent is a C library, it doesn't directly provide a mechanism to associate an instance and an instance method, so we need to do it ourself.

Private Attributes

- `int _port`
- `char _pendingBuffer [MAXDATABUFFER]`
- `int _pendingBufferSize`

6.10.1 Constructor & Destructor Documentation

6.10.1.1 `MarketData::SubscriptionServer::SubscriptionServer (int port)`

6.10.2 Member Function Documentation

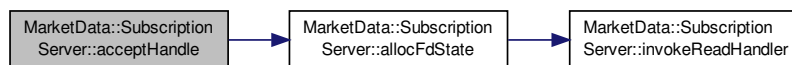
6.10.2.1 `void MarketData::SubscriptionServer::acceptHandle (evutil_socket_t listener, short event, event_base * base)` [private]

Accept Handle.

Parameters

<i>listener</i>	Socket fd
<i>event</i>	Flags
<i>*base</i>	Pointer to event base

Here is the call graph for this function:



6.10.2.2 `FdState * MarketData::SubscriptionServer::allocFdState (struct event_base * base, evutil_socket_t fd)` [private]

Allocate `FdState`.

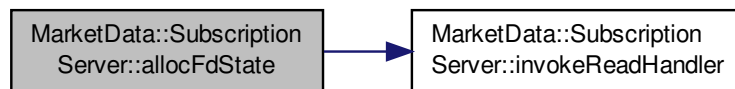
Parameters

<i>base</i>	
<i>fd</i>	

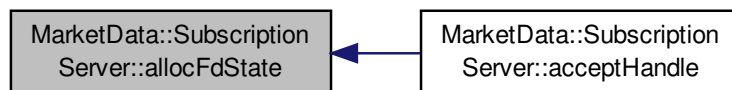
Returns

[FdState](#)

Here is the call graph for this function:



Here is the caller graph for this function:



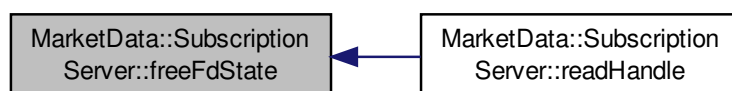
6.10.2.3 `void MarketData::SubscriptionServer::freeFdState (FdState * state)` [private]

Free allocated state memory.

Parameters

<i>state</i>	
--------------	--

Here is the caller graph for this function:



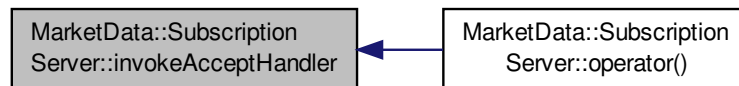
6.10.2.4 `void MarketData::SubscriptionServer::invokeAcceptHandler (evutil_socket_t fd, short events, void * arg)`
[static], [private]

Invoke Accept Handler Since libevent is a C library, it doesn't directly provide a mechanism to associate an instance and an instance method, so we need to do it ourself.

Parameters

<i>fd</i>	Socket fd
<i>events</i>	Flags
<i>*arg</i>	Pointer to event base

Here is the caller graph for this function:



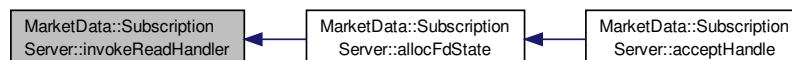
6.10.2.5 `void MarketData::SubscriptionServer::invokeReadHandler (evutil_socket_t fd, short events, void * arg)`
`[static], [private]`

Invoke Read Handler Since libevent is a C library, it doesn't directly provide a mechanism to associate an instance and an instance method, so we need to do it ourself.

Parameters

<i>fd</i>	Socket fd
<i>events</i>	Flags
<i>*arg</i>	Pointer to event base

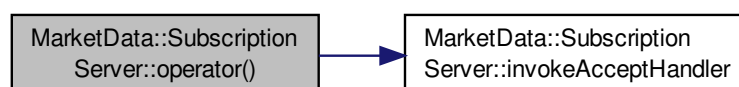
Here is the caller graph for this function:



6.10.2.6 `void MarketData::SubscriptionServer::operator() ()`

overloaded function operator.

Here is the call graph for this function:



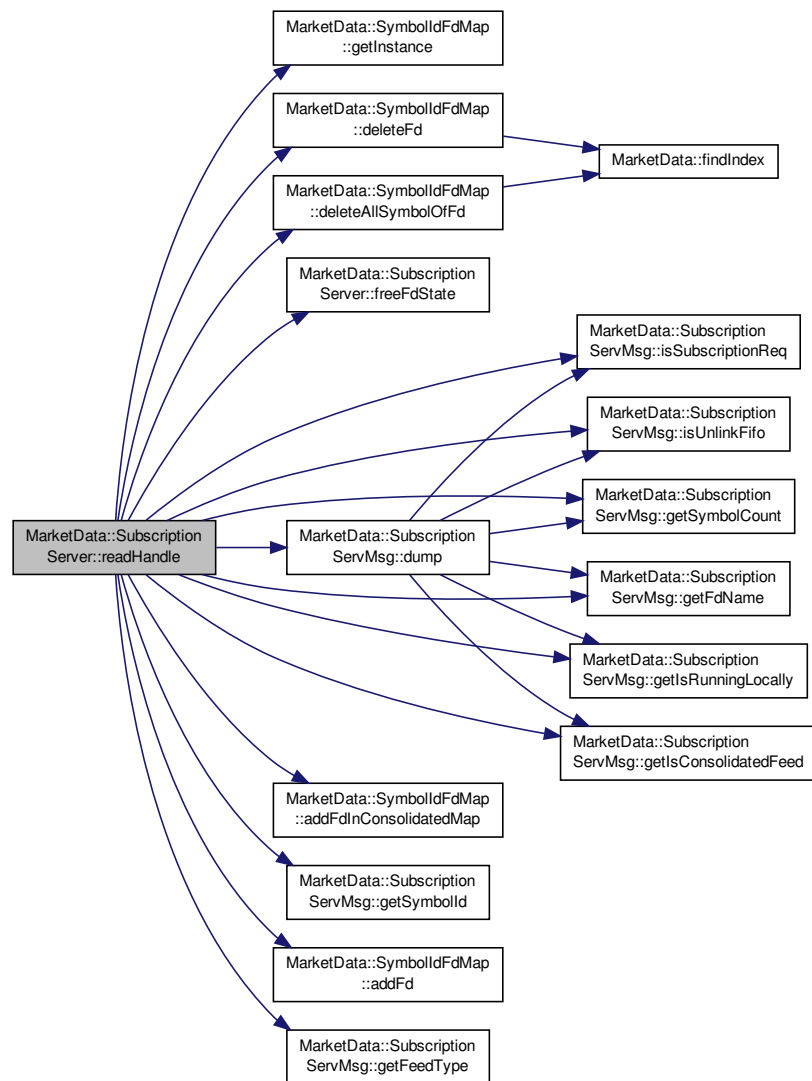
6.10.2.7 `void MarketData::SubscriptionServer::readHandle (evutil_socket_t fd, short events, FdState * state)`
`[private]`

Read Handle.

Parameters

<i>fd</i>	Socket fd
<i>events</i>	Flags
<i>*state</i>	

Here is the call graph for this function:



6.10.3 Member Data Documentation

6.10.3.1 `char MarketData::SubscriptionServer::_pendingBuffer[MAXDATABUFFER]` `[private]`

6.10.3.2 `int MarketData::SubscriptionServer::_pendingBufferSize` `[private]`

6.10.3.3 `int MarketData::SubscriptionServer::_port` [private]

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

- [subscriptionServer.h](#)
- [subscriptionServer.cpp](#)

6.11 `MarketData::SubscriptionServMsg` Class Reference

This class is responsible for serialization of request data to the server.

```
#include <subscriptionServMsg.h>
```

Collaboration diagram for `MarketData::SubscriptionServMsg`:

<code>MarketData::SubscriptionServMsg</code>
<ul style="list-style-type: none"> - <code>_isSubscription</code> - <code>_isUnlinkFifo</code> - <code>_symbolCount</code> - <code>_symbolId</code> - <code>_feedType</code> - <code>_fdName</code> - <code>_isRunningLocally</code> - <code>_isConsolidatedFeed</code> - <code>FD_LENGTH</code> - <code>MAX_SYMBOL_COUNT</code>
<ul style="list-style-type: none"> + <code>SubscriptionServMsg()</code> + <code>SubscriptionServMsg()</code> + <code>serialize()</code> + <code>dump()</code> + <code>isSubscriptionReq()</code> + <code>setSubscriptionReq()</code> + <code>isUnlinkFifo()</code> + <code>setUnlinkFifo()</code> + <code>getSymbolCount()</code> + <code>setSymbolCount()</code> and 10 more...

Public Member Functions

- [SubscriptionServMsg](#) ()
- [SubscriptionServMsg](#) (const char *buf)
- int [serialize](#) (char *buf)
Serialize data.

- void `dump` ()
Dump data.
- `UNSIGNED_CHARACTER` `isSubscriptionReq` () const
To check whether the request is to subscribe or unsubscribe.
- void `setSubscriptionReq` (`UNSIGNED_CHARACTER` val)
To check whether the request is to subscribe or unsubscribe.
- `UNSIGNED_CHARACTER` `isUnlinkFifo` () const
- void `setUnlinkFifo` (`UNSIGNED_CHARACTER` val)
- `UNSIGNED_SHORT` `getSymbolCount` () const
- void `setSymbolCount` (`UNSIGNED_SHORT` val)
- `UNSIGNED_LONG` `getSymbolId` (int subs)
- void `setSymbolId` (int subs, int val)
- `UNSIGNED_CHARACTER` `getFeedType` (int subs)
- void `setFeedType` (int subs, `UNSIGNED_CHARACTER` val)
- const char * `getFdName` () const
- void `setFdName` (const char *m)
- `UNSIGNED_SHORT` `getIsRunningLocally` ()
- void `setIsRunningLocally` (`UNSIGNED_SHORT` val=1)
- `UNSIGNED_SHORT` `getIsConsolidatedFeed` ()
- void `setIsConsolidatedFeed` (`UNSIGNED_SHORT` val=0)

Private Types

- typedef std::map
 < `UNSIGNED_LONG`, `UNSIGNED_LONG` > `SymbolIdMap`
- typedef std::map
 < `UNSIGNED_LONG`, `UNSIGNED_LONG` > `FeedTypeMap`

Private Attributes

- `UNSIGNED_SHORT` `_isSubscription`
- `UNSIGNED_SHORT` `_isUnlinkFifo`
- `UNSIGNED_SHORT` `_symbolCount`
- `SymbolIdMap` `_symbolId`
- `FeedTypeMap` `_feedType`
- char `_fdName` [`FD_LENGTH`]
- `UNSIGNED_SHORT` `_isRunningLocally`
- `UNSIGNED_SHORT` `_isConsolidatedFeed`

Static Private Attributes

- static const int `FD_LENGTH` = 100
- static const int `MAX_SYMBOL_COUNT` = 10

6.11.1 Detailed Description

This class is responsible for serialization of request data to the server.

6.11.2 Member Typedef Documentation

6.11.2.1 `typedef std::map<UNSIGNED_LONG, UNSIGNED_LONG> MarketData::SubscriptionServMsg::Feed-
TypeMap [private]`

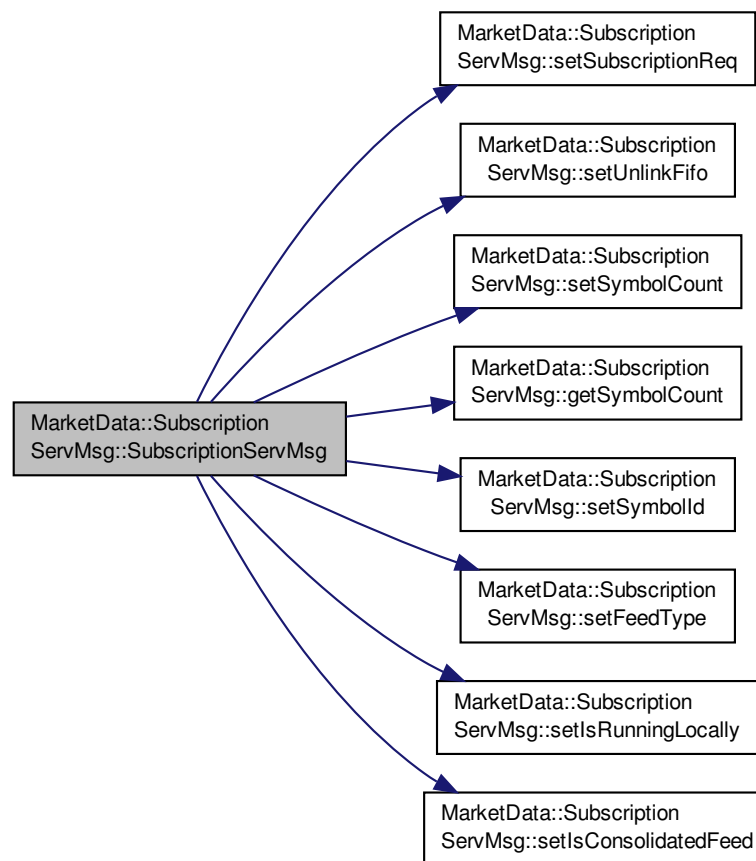
6.11.2.2 `typedef std::map<UNSIGNED_LONG, UNSIGNED_LONG> MarketData::SubscriptionServMsg::-
SymbolIdMap [private]`

6.11.3 Constructor & Destructor Documentation

6.11.3.1 `MarketData::SubscriptionServMsg::SubscriptionServMsg () [inline]`

6.11.3.2 `MarketData::SubscriptionServMsg::SubscriptionServMsg (const char * buf)`

Here is the call graph for this function:

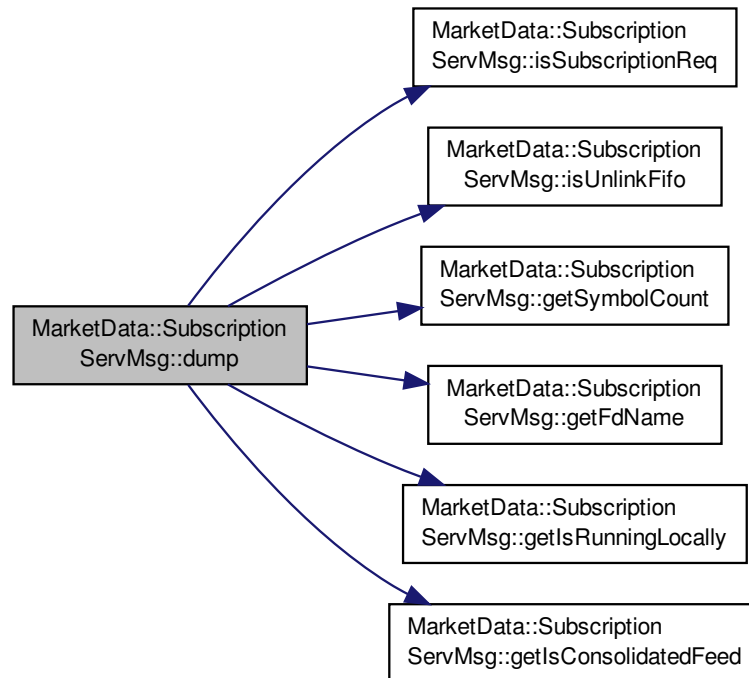


6.11.4 Member Function Documentation

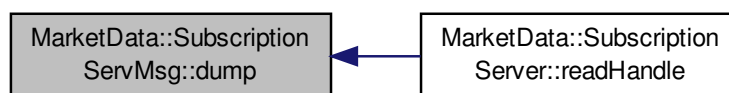
6.11.4.1 `void MarketData::SubscriptionServMsg::dump ()`

Dump data.

Here is the call graph for this function:

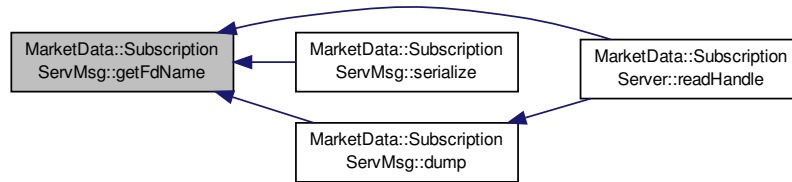


Here is the caller graph for this function:



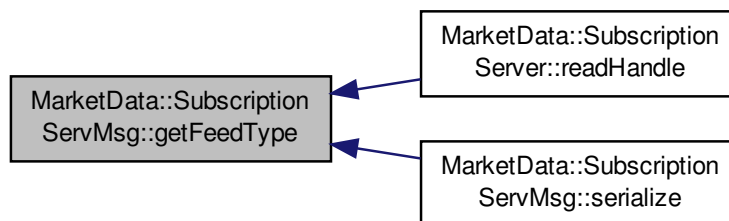
6.11.4.2 `const char* MarketData::SubscriptionServMsg::getFdName () const` `[inline]`

Here is the caller graph for this function:



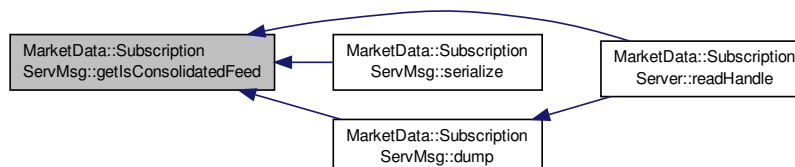
6.11.4.3 `UNSIGNED_CHARACTER MarketData::SubscriptionServMsg::getFeedType (int subs)` `[inline]`

Here is the caller graph for this function:



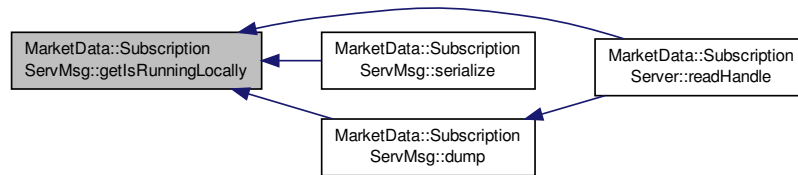
6.11.4.4 `UNSIGNED_SHORT MarketData::SubscriptionServMsg::getIsConsolidatedFeed ()` `[inline]`

Here is the caller graph for this function:



6.11.4.5 UNSIGNED_SHORT MarketData::SubscriptionServMsg::getIsRunningLocally () [inline]

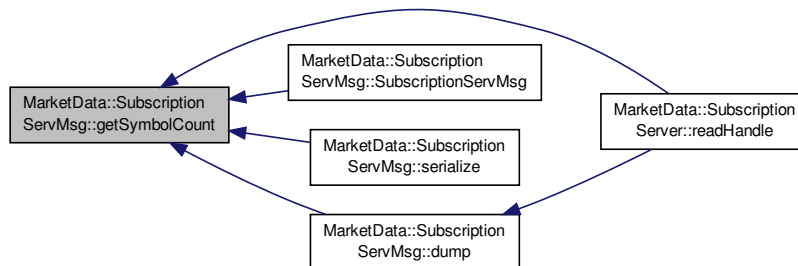
Here is the caller graph for this function:



6.11.4.6 UNSIGNED_SHORT MarketData::SubscriptionServMsg::getSymbolCount () const [inline]

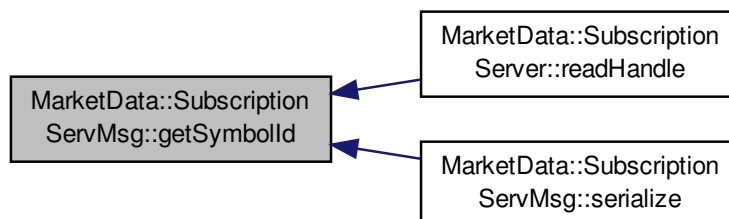
Returns

Here is the caller graph for this function:



6.11.4.7 UNSIGNED_LONG MarketData::SubscriptionServMsg::getSymbolId (int subs) [inline]

Here is the caller graph for this function:



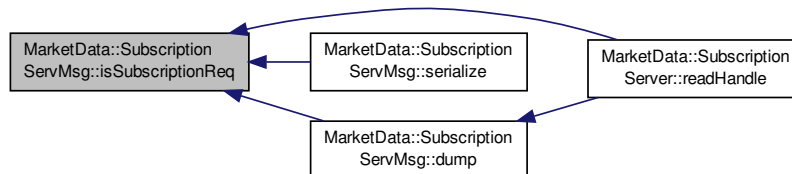
6.11.4.8 `UNSIGNED_CHARACTER` `MarketData::SubscriptionServMsg::isSubscriptionReq () const` `[inline]`

To check whether the request is to subscribe or unsubscribe.

Returns

1 to Subscribe 0 to Unsubscribe

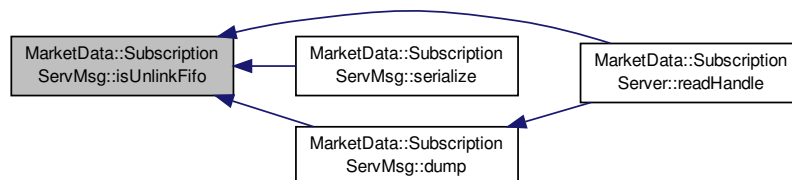
Here is the caller graph for this function:



6.11.4.9 `UNSIGNED_CHARACTER` `MarketData::SubscriptionServMsg::isUnlinkFifo () const` `[inline]`

Returns

Here is the caller graph for this function:



6.11.4.10 `int` `MarketData::SubscriptionServMsg::serialize (char * buf)`

Serialize data.

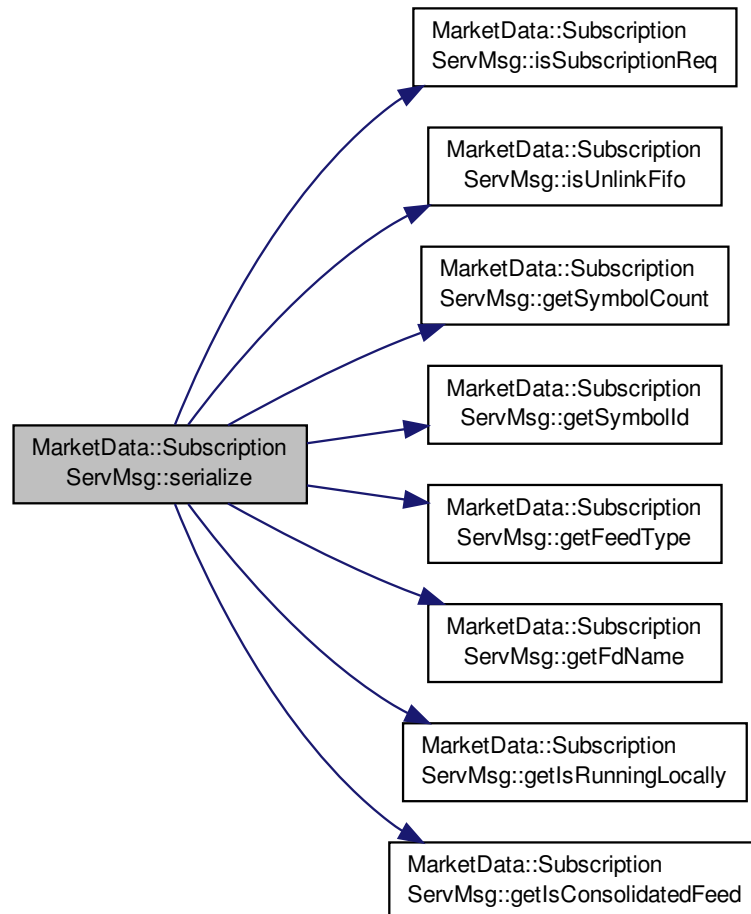
Parameters

<i>buf</i>	
------------	--

Returns

Size of serialized data

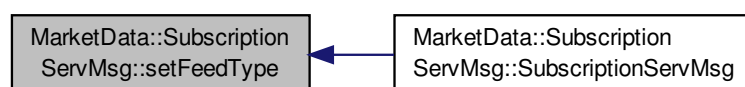
Here is the call graph for this function:



6.11.4.11 `void MarketData::SubscriptionServMsg::setFdName (const char * m) [inline]`

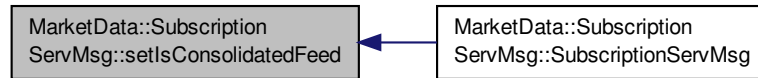
6.11.4.12 `void MarketData::SubscriptionServMsg::setFeedType (int subs, UNSIGNED_CHARACTER val) [inline]`

Here is the caller graph for this function:



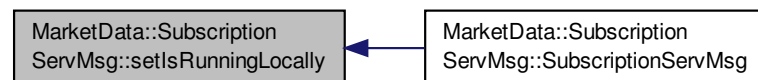
6.11.4.13 `void MarketData::SubscriptionServMsg::setIsConsolidatedFeed (UNSIGNED_SHORT val = 0) [inline]`

Here is the caller graph for this function:



6.11.4.14 `void MarketData::SubscriptionServMsg::setIsRunningLocally (UNSIGNED_SHORT val = 1) [inline]`

Here is the caller graph for this function:



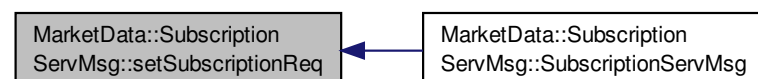
6.11.4.15 `void MarketData::SubscriptionServMsg::setSubscriptionReq (UNSIGNED_CHARACTER val) [inline]`

To check whether the request is to subscribe or unsubscribe.

Parameters

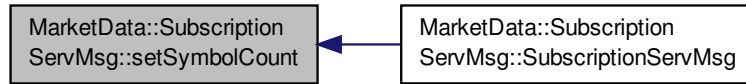
<i>val</i>	
------------	--

Here is the caller graph for this function:



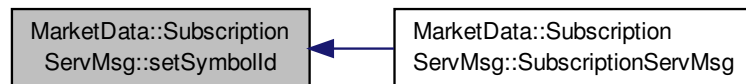
6.11.4.16 `void MarketData::SubscriptionServMsg::setSymbolCount (UNSIGNED_SHORT val) [inline]`

Here is the caller graph for this function:



6.11.4.17 `void MarketData::SubscriptionServMsg::setSymbolId (int subs, int val) [inline]`

Here is the caller graph for this function:



6.11.4.18 `void MarketData::SubscriptionServMsg::setUnlinkFifo (UNSIGNED_CHARACTER val) [inline]`

Here is the caller graph for this function:



6.11.5 Member Data Documentation

6.11.5.1 `char MarketData::SubscriptionServMsg::_fdName[FD_LENGTH] [private]`

6.11.5.2 `FeedTypeMap MarketData::SubscriptionServMsg::_feedType [private]`

6.11.5.3 `UNSIGNED_SHORT MarketData::SubscriptionServMsg::_isConsolidatedFeed [private]`

value 1 - Request Consolidated Data value 0 - Normal Data

6.11.5.4 **UNSIGNED_SHORT** `MarketData::SubscriptionServMsg::isRunningLocally` `[private]`

value 1 - Both are on same system(Default value) value 0 - Different

6.11.5.5 **UNSIGNED_SHORT** `MarketData::SubscriptionServMsg::isSubscription` `[private]`

value 0 - unsubscribe request value 1 - subscribe request

6.11.5.6 **UNSIGNED_SHORT** `MarketData::SubscriptionServMsg::isUnlinkFifo` `[private]`

value 0 - Leave value 1 - Unlink

6.11.5.7 **UNSIGNED_SHORT** `MarketData::SubscriptionServMsg::symbolCount` `[private]`

6.11.5.8 **SymbolIdMap** `MarketData::SubscriptionServMsg::symbolId` `[private]`

6.11.5.9 **const int** `MarketData::SubscriptionServMsg::FD_LENGTH = 100` `[static]`, `[private]`

6.11.5.10 **const int** `MarketData::SubscriptionServMsg::MAX_SYMBOL_COUNT = 10` `[static]`, `[private]`

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

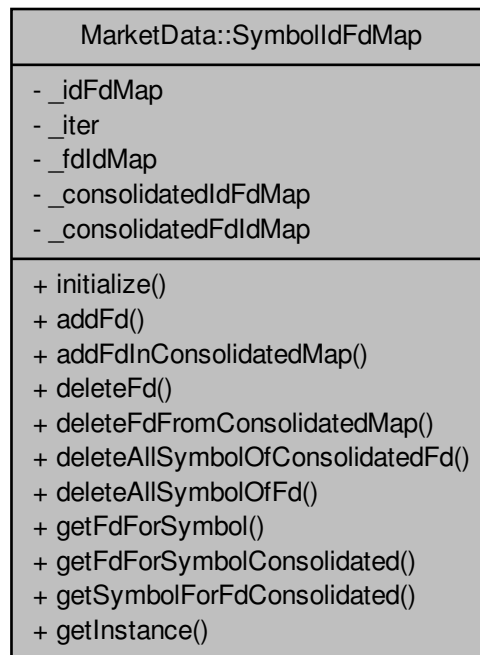
- [subscriptionServMsg.h](#)
- [subscriptionServMsg.cpp](#)

6.12 `MarketData::SymbolIdFdMap` Class Reference

[SymbolIdFdMap](#) class This is the data structure which is used by tbt to send event on each subscriber for each symbol id.

```
#include <symbolIdFdMap.h>
```

Collaboration diagram for MarketData::SymbolIdFdMap:



Public Member Functions

- void [initialize](#) (int symbolId)
Iniitalize symbol id map. This function will be called when tbt will start.
- void [addFd](#) (int symbolId, int feedType, int fd)
Add member in the subscribed list.
- void [addFdInConsolidatedMap](#) (int symbolId, int fd)
Add member in the subscribed list.
- void [deleteFd](#) (int symbolId, int fd, int feedType)
Delete member from subscribed list.
- void [deleteFdFromConsolidatedMap](#) (int symbolId, int fd)
Delete member from subscribed list.
- void [deleteAllSymbolOfConsolidatedFd](#) (int fd)
- void [deleteAllSymbolOfFd](#) (int fd)
- std::deque< std::pair< int, int > > & [getFdForSymbol](#) (int symbolId) throw (std::domain_error)
Get Fds associated to symbol id.
- std::deque< int > & [getFdForSymbolConsolidated](#) (int symbolId) throw (std::domain_error)
- std::deque< int > & [getSymbolForFdConsolidated](#) (int fd) throw (std::domain_error)

Static Public Member Functions

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

Private Attributes

- [IdFdMap_idFdMap](#)
- [IdFdMapItr_iter](#)
- [FdIdMap_fdIdMap](#)
- [ConsolidatedIdFdMap_consolidatedIdFdMap](#)
- [ConsolidatedIdFdMap_consolidatedFdIdMap](#)

6.12.1 Detailed Description

[SymbolIdFdMap](#) class This is the data structure which is used by tbt to send event on each subscriber for each symbol id.

6.12.2 Member Function Documentation

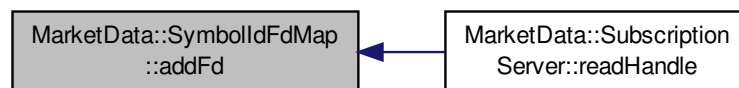
6.12.2.1 void `MarketData::SymbolIdFdMap::addFd (int symbolId, int feedType, int fd)`

Add member in the subscribed list.

Parameters

<i>symbolId</i>	
<i>fd</i>	

Here is the caller graph for this function:



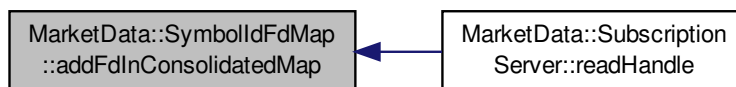
6.12.2.2 void `MarketData::SymbolIdFdMap::addFdInConsolidatedMap (int symbolId, int fd)`

Add member in the subscribed list.

Parameters

<i>symbolId</i>	
<i>fd</i>	

Here is the caller graph for this function:



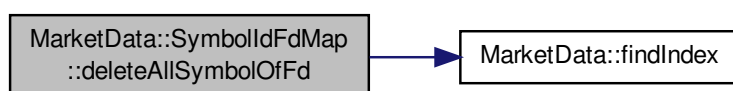
6.12.2.3 void MarketData::SymbolIdFdMap::deleteAllSymbolOfConsolidatedFd (int fd)

Here is the call graph for this function:

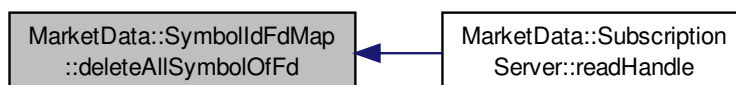


6.12.2.4 void MarketData::SymbolIdFdMap::deleteAllSymbolOfFd (int fd)

Here is the call graph for this function:



Here is the caller graph for this function:



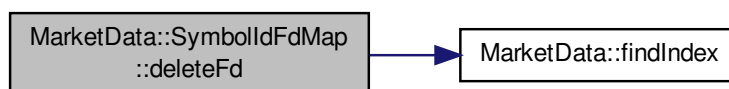
6.12.2.5 void MarketData::SymbolIdFdMap::deleteFd (int *symbolId*, int *fd*, int *feedType*)

Delete member from subscribed list.

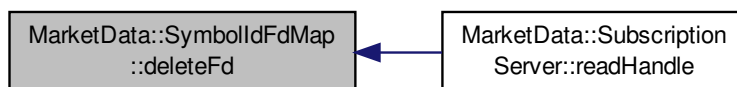
Parameters

<i>symbolId</i>	
<i>fd</i>	

Here is the call graph for this function:



Here is the caller graph for this function:



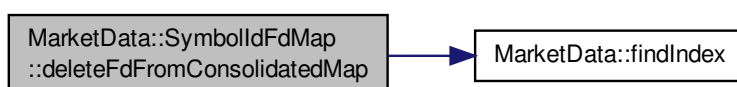
6.12.2.6 void MarketData::SymbolIdFdMap::deleteFdFromConsolidatedMap (int *symbolId*, int *fd*)

Delete member from subscribed list.

Parameters

<i>symbolId</i>	
<i>fd</i>	

Here is the call graph for this function:



6.12.2.7 `std::deque< std::pair< int, int > > & MarketData::SymbolIdFdMap::getFdForSymbol (int symbolId) throw (std::domain_error)`

Get Fds associated to symbol id.

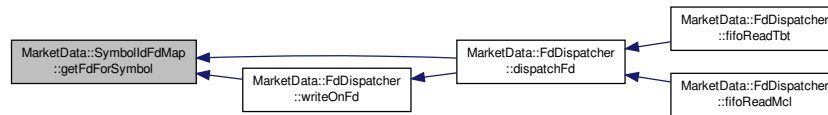
Parameters

<i>symbolId</i>

Returns

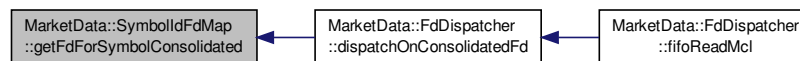
queue having fds associated to specific symbolId.

Here is the caller graph for this function:



6.12.2.8 `std::deque< int > & MarketData::SymbolIdFdMap::getFdForSymbolConsolidated (int symbolId) throw (std::domain_error)`

Here is the caller graph for this function:



6.12.2.9 `static SymbolIdFdMap* MarketData::SymbolIdFdMap::getInstance () [inline], [static]`

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

6.12.3.3 **FdIdMap** MarketData::SymbolIdFdMap::fdIdMap [private]

6.12.3.4 **IdFdMap** MarketData::SymbolIdFdMap::idFdMap [private]

6.12.3.5 **IdFdMapItr** MarketData::SymbolIdFdMap::iter [private]

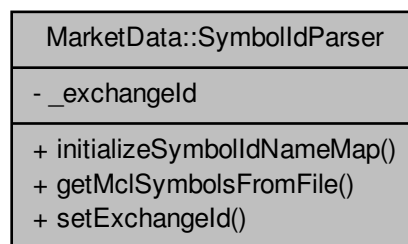
The documentation for this class was generated from the following files:

- [symbolIdFdMap.h](#)
- [symbolIdFdMap.cpp](#)

6.13 MarketData::SymbolIdParser Class Reference

```
#include <symbolIdParser.h>
```

Collaboration diagram for MarketData::SymbolIdParser:



Public Member Functions

- void [initializeSymbolIdNameMap](#) ()
- void [getMcISymbolsFromFile](#) (std::string file)

Static Public Member Functions

- static void [setExchangeId](#) (int val)

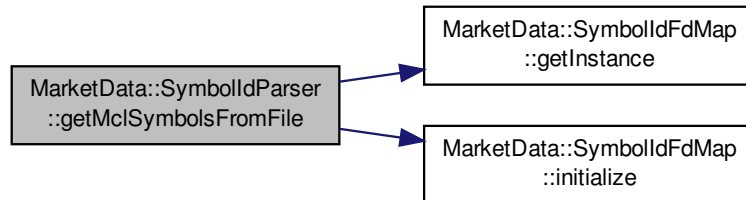
Static Private Attributes

- static int [_exchangeId](#) = 0

6.13.1 Member Function Documentation

6.13.1.1 void MarketData::SymbolIdParser::getMcISymbolsFromFile (std::string file)

Here is the call graph for this function:

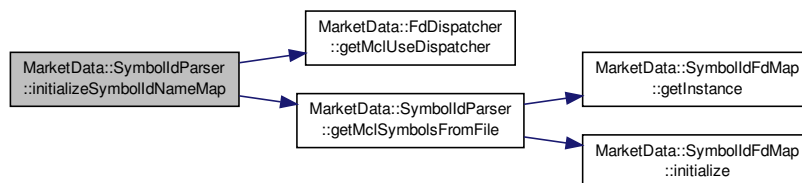


Here is the caller graph for this function:

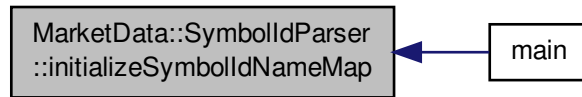


6.13.1.2 void MarketData::SymbolIdParser::initializeSymbolIdNameMap ()

Here is the call graph for this function:

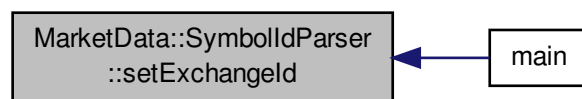


Here is the caller graph for this function:



6.13.1.3 `static void MarketData::SymbolIdParser::setExchangeId (int val)` `[inline],[static]`

Here is the caller graph for this function:



6.13.2 Member Data Documentation

6.13.2.1 `int MarketData::SymbolIdParser::_exchangeId = 0` `[static],[private]`

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

- [symbolIdParser.h](#)
- [symbolIdParser.cpp](#)

6.14 MarketData::SymbolNameIdMap Class Reference

```
#include <symbolIdFdMap.h>
```

Collaboration diagram for MarketData::SymbolNameIdMap:

MarketData::SymbolNameIdMap
- _symbolNameIdMap
+ setId() + getId() + getInstance() - SymbolNameIdMap()

Public Member Functions

- void [setId](#) (std::string &name, int &id)
- int [getId](#) (std::string &name)

Static Public Member Functions

- static [SymbolNameIdMap](#) * [getInstance](#) ()

Private Member Functions

- [SymbolNameIdMap](#) ()

Private Attributes

- std::map< std::string, int > [_symbolNameIdMap](#)

6.14.1 Constructor & Destructor Documentation

6.14.1.1 `MarketData::SymbolNameIdMap::SymbolNameIdMap ()` `[inline]`, `[private]`

6.14.2 Member Function Documentation

6.14.2.1 `int MarketData::SymbolNameIdMap::getId (std::string & name)`

6.14.2.2 `static SymbolNameIdMap* MarketData::SymbolNameIdMap::getInstance ()` `[inline]`, `[static]`

6.14.2.3 `void MarketData::SymbolNameIdMap::setId (std::string & name, int & id)`

6.14.3 Member Data Documentation

6.14.3.1 `std::map<std::string, int> MarketData::SymbolNameIdMap::_symbolNameIdMap` `[private]`

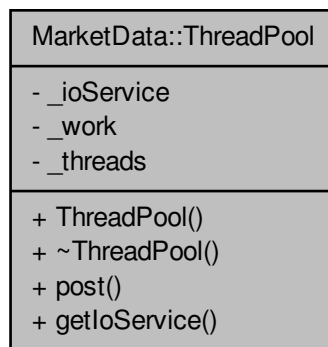
The documentation for this class was generated from the following files:

- [symbolIdFdMap.h](#)
- [symbolIdFdMap.cpp](#)

6.15 MarketData::ThreadPool Class Reference

```
#include <threadPool.h>
```

Collaboration diagram for MarketData::ThreadPool:



Public Member Functions

- [ThreadPool](#) (int noOfThreads=1)
- [~ThreadPool](#) ()
- template<class func >
void [post](#) (func f)
- boost::asio::io_service & [getIoService](#) ()

Private Attributes

- boost::asio::io_service [_ioService](#)
- boost::asio::io_service::work [_work](#)
- boost::thread_group [_threads](#)

6.15.1 Constructor & Destructor Documentation

6.15.1.1 [MarketData::ThreadPool::ThreadPool \(int noOfThreads = 1 \)](#) [inline]

6.15.1.2 [MarketData::ThreadPool::~~ThreadPool \(\)](#) [inline]

6.15.2 Member Function Documentation

6.15.2.1 [boost::asio::io_service & MarketData::ThreadPool::getIoService \(\)](#) [inline]

6.15.2.2 `template<class func > void MarketData::ThreadPool::post (func f)`

Here is the caller graph for this function:



6.15.3 Member Data Documentation

6.15.3.1 `boost::asio::io_service MarketData::ThreadPool::_ioService` [private]

6.15.3.2 `boost::thread_group MarketData::ThreadPool::_threads` [private]

6.15.3.3 `boost::asio::io_service::work MarketData::ThreadPool::_work` [private]

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

- [threadPool.h](#)

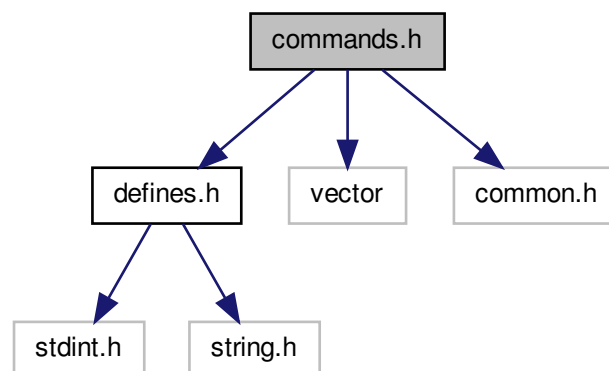
Chapter 7

File Documentation

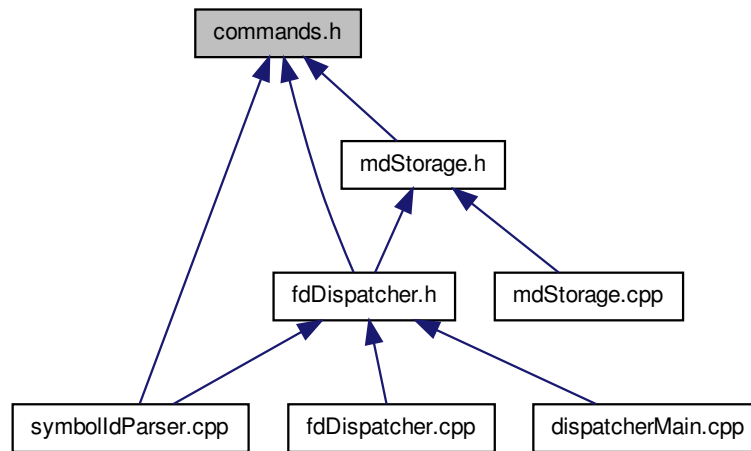
7.1 commands.h File Reference

```
#include "defines.h"  
#include <vector>  
#include "common.h"
```

Include dependency graph for commands.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CMD::ScripMasterDataRequest](#)
- class [CMD::InstrumentRequest](#)
Instrument search request.
- class [CMD::MDSubscribeRequest](#)
MarketData Subscribe/Unsubscribe Request.
- class [CMD::MDQuote](#)
MarketData Quote.
- class [CMD::MDQuoteConsolidated](#)
MarketData Quote Consolidated.

Namespaces

- namespace [CMD](#)

Macros

- `#define EXCHANGE_ID_BASE 10000000`
- `#define GET_SYMBOL_ID(scripCode, exchangeId) ((exchangeId)*EXCHANGE_ID_BASE + scripCode)`
- `#define GET_SCRIP_CODE(symbolId) (symbolId % EXCHANGE_ID_BASE)`
- `#define GET_EXCHANGE_ID(symbolId) ((CMD::ExchangeId)(symbolId/EXCHANGE_ID_BASE))`

Enumerations

- enum [CMD::ExchangeId](#) {
[CMD::ExchangeId_BSE](#) = 1, [CMD::ExchangeId_NSE](#), [CMD::ExchangeId_ESMNSE](#), [CMD::ExchangeId_SGX](#),
[CMD::ExchangeId_NSECDS](#), [CMD::ExchangeId_BSEETI](#), [CMD::ExchangeId_CFH](#), [CMD::ExchangeId_MAX](#) }

- enum `CMD::CommandCategory` {
`CMD::CommandCategory_SINGLE_ORDER`, `CMD::CommandCategory_DUMMY_TWO_LEG_STRATEGY`, `CMD::CommandCategory_TWO_LEG_ARBITRAGE_STRATEGY`, `CMD::CommandCategory_MARKET_MAKING_STRATEGY`,
`CMD::CommandCategory_STRATEGY_COMMAND`, `CMD::CommandCategory_PAUSE_STRATEGY`, `CMD::CommandCategory_RUN_STRATEGY`, `CMD::CommandCategory_TERMINATE_STRATEGY`,
`CMD::CommandCategory_TERMINATE_SQUAREOFF`, `CMD::CommandCategory_LOGIN`, `CMD::CommandCategory_LOGOUT`, `CMD::CommandCategory_RMS_UPDATE`,
`CMD::CommandCategory_REQUEST_RMS_CONFIGURATION`, `CMD::CommandCategory_REQUEST_OFFLINE_DATA`, `CMD::CommandCategory_HEART_BEAT_MESSAGE`, `CMD::CommandCategory_THREE_LEG_ARBITRAGE_STRATEGY`,
`CMD::CommandCategory_DUMMY_THREE_LEG_ARBITRAGE_STRATEGY`, `CMD::CommandCategory_BSE_DISCONNECTED`, `CMD::CommandCategory_NSECM_DISCONNECTED`, `CMD::CommandCategory_NSEFO_DISCONNECTED`,
`CMD::CommandCategory_TBTCM_DISCONNECTED`, `CMD::CommandCategory_TBTFO_DISCONNECTED`, `CMD::CommandCategory_BSE_SNAPSHOTFEED_DISCONNECTED`, `CMD::CommandCategory_NSEFO_SNAPSHOTFEED_DISCONNECTED`,
`CMD::CommandCategory_NSECM_SNAPSHOTFEED_DISCONNECTED`, `CMD::CommandCategory_MARKET_MAKING_TURNOVER_STRATEGY`, `CMD::CommandCategory_TBT_MARKET_DATA`, `CMD::CommandCategory_FOUR_LEG_BUTTERFLY_STRATEGY`,
`CMD::CommandCategory_BOX_STRATEGY`, `CMD::CommandCategory_TWO_LEG_THREE_LEG_ARBITRAGE_STRATEGY`, `CMD::CommandCategory_MODIFY_STRATEGY`, `CMD::CommandCategory_RMS_LOGIN_PARAMETERS`,
`CMD::CommandCategory_RMS_ORDER_LIMITS`, `CMD::CommandCategory_RMS_GLOBAL_PARAMETERS`, `CMD::CommandCategory_RMS_RISK_PERMISSIONS`, `CMD::CommandCategory_GET_RMS_PARAMETERS`,
`CMD::CommandCategory_CHANGE_PASSWORD`, `CMD::CommandCategory_SEND_SCRIP_MASTER_DATA`, `CMD::CommandCategory_GET_TOTAL_RECORDS_SCRIP_MASTER`, `CMD::CommandCategory_GET_TOTAL_ORDER_CONFIRMATIONS`,
`CMD::CommandCategory_GET_CONNECTIVITY_STATUS`, `CMD::CommandCategory_RMS_SECURITY_WISE_LIMITS`, `CMD::CommandCategory_IMPLIED_VOLATILITY_STRATEGY`, `CMD::CommandCategory_CASH_CASH_STRATEGY`,
`CMD::CommandCategory_GET_CLIENT_CONNECTIVITY_STATUS`, `CMD::CommandCategory_FORCE_LOG_OFF_CLIENT`, `CMD::CommandCategory_PAIRS_STRATEGY`, `CMD::CommandCategory_PAIRS_CLOSE_POSITIONS_NO_TERMINATE`,
`CMD::CommandCategory_TBTCDS_DISCONNECTED`, `CMD::CommandCategory_NSECDS_DISCONNECTED`, `CMD::CommandCategory_BSEETI_DISCONNECTED`, `CMD::CommandCategory_SGX_DISCONNECTED`,
`CMD::CommandCategory_INSTRUMENT_SEARCH_REQUEST`, `CMD::CommandCategory_MARKET_DATA_SUBSCRIPTION`, `CMD::CommandCategory_API_TRADED_SYMBOLS_COUNT_REQUEST`, `CMD::CommandCategory_API_REPLAY_TRADED_SYMBOLS_REQUEST`,
`CMD::CommandCategory_MAX` }

7.1.1 Macro Definition Documentation

7.1.1.1 `#define EXCHANGE_ID_BASE 10000000`

7.1.1.2 `#define GET_EXCHANGE_ID(symbolId) ((CMD::Exchangeld)(symbolId/EXCHANGE_ID_BASE))`

7.1.1.3 `#define GET_SCRIP_CODE(symbolId) (symbolId % EXCHANGE_ID_BASE)`

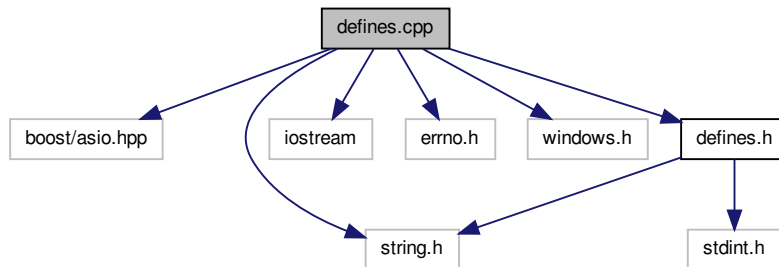
7.1.1.4 `#define GET_SYMBOL_ID(scripCode, exchangeld) ((exchangeld)*EXCHANGE_ID_BASE + scripCode)`

7.2 defines.cpp File Reference

```
#include <boost/asio.hpp>
```

```
#include <string.h>
#include <iostream>
#include <errno.h>
#include <windows.h>
#include "defines.h"
```

Include dependency graph for defines.cpp:



Macros

- `#define _WINSOCKAPI_`

Functions

- `UNSIGNED_SHORT htons_16 (uint16_t x)`
- `UNSIGNED_SHORT ntohs_16 (uint16_t x)`
- `UNSIGNED_INTEGER htonl_32 (uint32_t x)`
- `UNSIGNED_INTEGER ntohl_32 (uint32_t x)`
- `UNSIGNED_LONG htonl_64 (uint64_t x)`
- `UNSIGNED_LONG ntohl_64 (uint64_t x)`
- `SIGNED_LONG signed_htonl_64 (int64_t x)`
- `SIGNED_LONG signed_ntohl_64 (int64_t x)`
- `long generateUniqueOrderId (int strategyId)`

Variables

- `static pthread_mutex_t uniqueOrderIdMutex = PTHREAD_MUTEX_INITIALIZER`

7.2.1 Macro Definition Documentation

7.2.1.1 `#define _WINSOCKAPI_`

7.2.2 Function Documentation

7.2.2.1 `long generateUniqueOrderId (int strategyId)`

7.2.2.2 `UNSIGNED_INTEGER htonl_32 (uint32_t x)`

7.2.2.3 `UNSIGNED_LONG htonl_64 (uint64_t x)`

7.2.2.4 **UNSIGNED_SHORT** htons_16 (uint16_t x)

7.2.2.5 **UNSIGNED_INTEGER** ntohl_32 (uint32_t x)

7.2.2.6 **UNSIGNED_LONG** ntohl_64 (uint64_t x)

7.2.2.7 **UNSIGNED_SHORT** ntohs_16 (uint16_t x)

7.2.2.8 **SIGNED_LONG** signed_htonl_64 (int64_t x)

7.2.2.9 **SIGNED_LONG** signed_ntohl_64 (int64_t x)

7.2.3 Variable Documentation

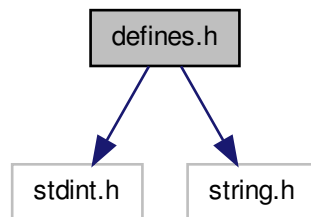
7.2.3.1 pthread_mutex_t uniqueOrderIdMutex = PTHREAD_MUTEX_INITIALIZER [static]

7.3 defines.h File Reference

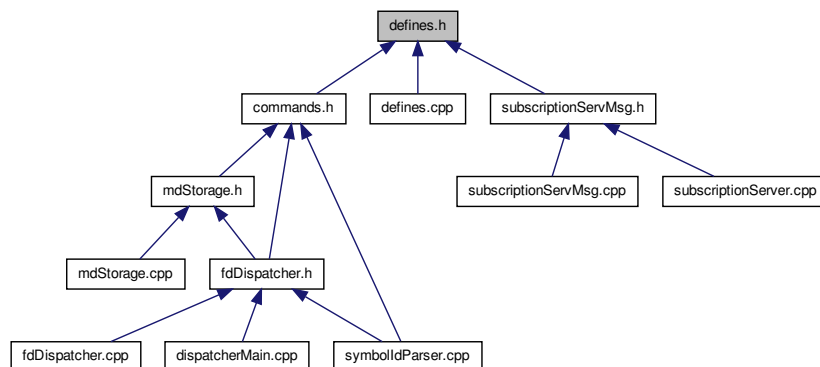
```
#include <stdint.h>
```

```
#include <string.h>
```

Include dependency graph for defines.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define SYMBOL_SIZE 24`
- `#define SCRIP_NAME_SIZE 100`
- `#define SYMBOL_ALIAS_SIZE 50`
- `#define ISIN_SIZE 50`
- `#define MARKET_PICTURE_BROADCAST_FLAG_SIZE 10`
- `#define PRODUCT_CODE_SIZE 20`
- `#define MARKET_NAME_SIZE 24`
- `#define SERIES_SIZE 8`
- `#define PASSWORD_SIZE 100`
- `#define PASSWORD_SALT_SIZE 100`
- `#define CHARACTER_FIELD_SIZE 100`
- `#define ACCOUNT_FIELD_SIZE 12`
- `#define IS_BIG_ENDIAN (*(uint16_t *)"0\xff" < 0x100)`
- `#define SWAP16(a)`
- `#define SWAP32(a)`
- `#define SWAP64(a)`
- `#define SERIALIZE_8(tmp, getMethod, buf, bytes)`
- `#define SERIALIZE_16(tmp, getMethod, buf, bytes)`
- `#define SERIALIZE_32(tmp, getMethod, buf, bytes)`
- `#define SERIALIZE_64(tmp, getMethod, buf, bytes)`
- `#define SERIALIZE_SIGNED_64(tmp, getMethod, buf, bytes)`
- `#define DESERIALIZE_8(tmp, setMethod, buf, offset)`
- `#define DESERIALIZE_16(tmp, tmpData, setMethod, buf, offset)`
- `#define DESERIALIZE_32(tmp, tmpData, setMethod, buf, offset)`
- `#define DESERIALIZE_64(tmp, tmpData, setMethod, buf, offset)`
- `#define DESERIALIZE_SIGNED_64(tmp, tmpData, setMethod, buf, offset)`

Typedefs

- `typedef uint8_t UNSIGNED_CHARACTER`
- `typedef uint16_t UNSIGNED_SHORT`
- `typedef uint32_t UNSIGNED_INTEGER`
- `typedef uint64_t UNSIGNED_LONG`
- `typedef int64_t SIGNED_LONG`

Functions

- `UNSIGNED_SHORT htons_16 (uint16_t x)`
- `UNSIGNED_SHORT ntohs_16 (uint16_t x)`
- `UNSIGNED_INTEGER htonl_32 (uint32_t x)`
- `UNSIGNED_INTEGER ntohl_32 (uint32_t x)`
- `UNSIGNED_LONG htonl_64 (uint64_t x)`
- `SIGNED_LONG signed_htonl_64 (int64_t x)`
- `UNSIGNED_LONG ntohl_64 (uint64_t x)`
- `SIGNED_LONG signed_ntohl_64 (int64_t x)`
- `char * getPacket (int socketFd, char recvbuf[], int bufSize, int &bytesAvailable, int &bytesProcessed, bool &noMoreBytes, bool &clientDisconnected)`

7.3.1 Macro Definition Documentation

7.3.1.1 #define ACCOUNT_FIELD_SIZE 12

7.3.1.2 #define CHARACTER_FIELD_SIZE 100

7.3.1.3 #define DESERIALIZE_16(tmp, tmpData, setMethod, buf, offset)

Value:

```
memcpy(&tmp, buf + offset, sizeof(tmp));\
tmp = ntohs_16(tmp);\
memcpy(&tmpData, &tmp, sizeof(tmpData));\
setMethod;\
offset += sizeof(tmp);
```

7.3.1.4 #define DESERIALIZE_32(tmp, tmpData, setMethod, buf, offset)

Value:

```
memcpy(&tmp, buf + offset, sizeof(tmp));\
tmp = ntohl_32(tmp);\
memcpy(&tmpData, &tmp, sizeof(tmpData));\
setMethod;\
offset += sizeof(tmp);
```

7.3.1.5 #define DESERIALIZE_64(tmp, tmpData, setMethod, buf, offset)

Value:

```
memcpy(&tmp, buf + offset, sizeof(tmp));\
tmp = ntohl_64(tmp);\
memcpy(&tmpData, &tmp, sizeof(tmpData));\
setMethod;\
offset += sizeof(tmp);
```

7.3.1.6 #define DESERIALIZE_8(tmp, setMethod, buf, offset)

Value:

```
memcpy(&tmp, buf + offset, sizeof(tmp));\
setMethod;\
offset += sizeof(tmp);
```

7.3.1.7 #define DESERIALIZE_SIGNED_64(tmp, tmpData, setMethod, buf, offset)

Value:

```
memcpy(&tmp, buf + offset, sizeof(tmp));\
tmp = signed_ntohl_64(tmp);\
memcpy(&tmpData, &tmp, sizeof(tmpData));\
setMethod;\
offset += sizeof(tmp);
```

7.3.1.8 `#define IS_BIG_ENDIAN (*(uint16_t *)"\0\xff" < 0x100)`

7.3.1.9 `#define ISIN_SIZE 50`

7.3.1.10 `#define MARKET_NAME_SIZE 24`

7.3.1.11 `#define MARKET_PICTURE_BROADCAST_FLAG_SIZE 10`

7.3.1.12 `#define PASSWORD_SALT_SIZE 100`

7.3.1.13 `#define PASSWORD_SIZE 100`

7.3.1.14 `#define PRODUCT_CODE_SIZE 20`

7.3.1.15 `#define SCRIP_NAME_SIZE 100`

7.3.1.16 `#define SERIALIZE_16(tmp, getMethod, buf, bytes)`

Value:

```
tmp = htons_16(getMethod);\
memcpy(buf+bytes, &tmp, sizeof(tmp));\
bytes += sizeof(tmp);
```

7.3.1.17 `#define SERIALIZE_32(tmp, getMethod, buf, bytes)`

Value:

```
tmp = htonl_32(getMethod);\
memcpy(buf+bytes, &tmp, sizeof(tmp));\
bytes += sizeof(tmp);
```

7.3.1.18 `#define SERIALIZE_64(tmp, getMethod, buf, bytes)`

Value:

```
tmp = htonl_64(getMethod);\
memcpy(buf+bytes, &tmp, sizeof(tmp));\
bytes += sizeof(tmp);
```

7.3.1.19 `#define SERIALIZE_8(tmp, getMethod, buf, bytes)`

Value:

```
tmp = getMethod;\
memcpy(buf+bytes, &tmp, sizeof(tmp));\
bytes += sizeof(tmp);
```

7.3.1.20 `#define SERIALIZE_SIGNED_64(tmp, getMethod, buf, bytes)`

Value:

```
tmp = signed_htonl_64(getMethod);\
memcpy(buf+bytes, &tmp, sizeof(tmp));\
bytes += sizeof(tmp);
```

7.3.1.21 #define SERIES_SIZE 8**7.3.1.22 #define SWAP16(a)****Value:**

```
( \
    ((a & 0x00FF) << 8) | ((a & 0xFF00) >> 8) )
```

7.3.1.23 #define SWAP32(a)**Value:**

```
( \
    ((a & 0x000000FF) << 24) | \
    ((a & 0x0000FF00) << 8) | \
    ((a & 0x00FF0000) >> 8) | \
    ((a & 0xFF000000) >> 24) )
```

7.3.1.24 #define SWAP64(a)**Value:**

```
( \
    ((a & 0x0000000000000000FFULL) << 56) | \
    ((a & 0x0000000000000000FF00ULL) << 40) | \
    ((a & 0x0000000000000000FF0000ULL) << 24) | \
    ((a & 0x0000000000000000FF000000ULL) << 8) | \
    ((a & 0x000000FF0000000000ULL) >> 8) | \
    ((a & 0x0000FF000000000000ULL) >> 24) | \
    ((a & 0x00FF00000000000000ULL) >> 40) | \
    ((a & 0xFF0000000000000000ULL) >> 56) )
```

7.3.1.25 #define SYMBOL_ALIAS_SIZE 50**7.3.1.26 #define SYMBOL_SIZE 24****7.3.2 Typedef Documentation****7.3.2.1 typedef int64_t SIGNED_LONG****7.3.2.2 typedef uint8_t UNSIGNED_CHARACTER****7.3.2.3 typedef uint32_t UNSIGNED_INTEGER****7.3.2.4 typedef uint64_t UNSIGNED_LONG****7.3.2.5 typedef uint16_t UNSIGNED_SHORT****7.3.3 Function Documentation****7.3.3.1 char* getPacket (int socketFd, char recvbuff[], int bufSize, int & bytesAvailable, int & bytesProcessed, bool & noMoreBytes, bool & clientDisconnected)****7.3.3.2 UNSIGNED_INTEGER htonl_32 (uint32_t x)****7.3.3.3 UNSIGNED_LONG htonl_64 (uint64_t x)**

7.3.3.4 **UNSIGNED_SHORT** htons_16 (uint16_t x)

7.3.3.5 **UNSIGNED_INTEGER** ntohl_32 (uint32_t x)

7.3.3.6 **UNSIGNED_LONG** ntohl_64 (uint64_t x)

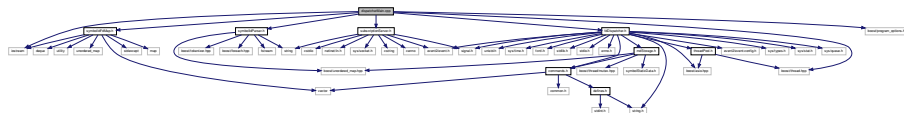
7.3.3.7 **UNSIGNED_SHORT** ntohs_16 (uint16_t x)

7.3.3.8 **SIGNED_LONG** signed_htonl_64 (int64_t x)

7.3.3.9 **SIGNED_LONG** signed_ntohl_64 (int64_t x)

7.4 dispatcherMain.cpp File Reference

```
#include <iostream>
#include "symbolIdParser.h"
#include "symbolIdFdMap.h"
#include <boost/thread.hpp>
#include <boost/program_options.hpp>
#include "subscriptionServer.h"
#include "fdDispatcher.h"
Include dependency graph for dispatcherMain.cpp:
```

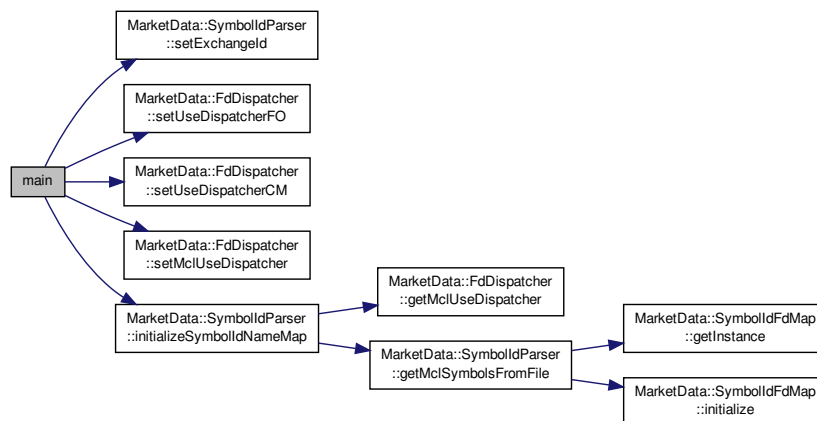


Functions

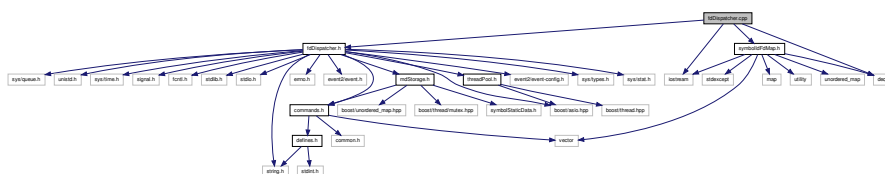
- int [main](#) ()

7.4.1 Function Documentation

Here is the call graph for this function:



```
#include <iostream>
#include "fdDispatcher.h"
#include "symbolIdFdMap.h"
#include <deque>
Include dependency graph for fdDispatcher.cpp:
```

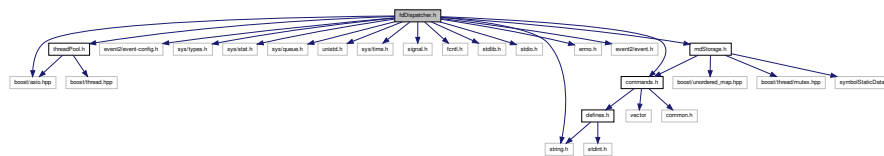


- namespace **MarketData**

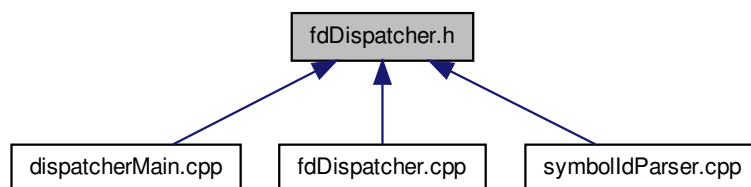
7.6 fdDispatcher.h File Reference

```
#include <boost/asio.hpp>
#include "threadPool.h"
#include <event2/event-config.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/queue.h>
#include <unistd.h>
#include <sys/time.h>
#include <signal.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <event2/event.h>
#include <commands.h>
#include "mdStorage.h"
```

Include dependency graph for fdDispatcher.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [MarketData::FdDispatcher](#)
FdDispatcher Class.

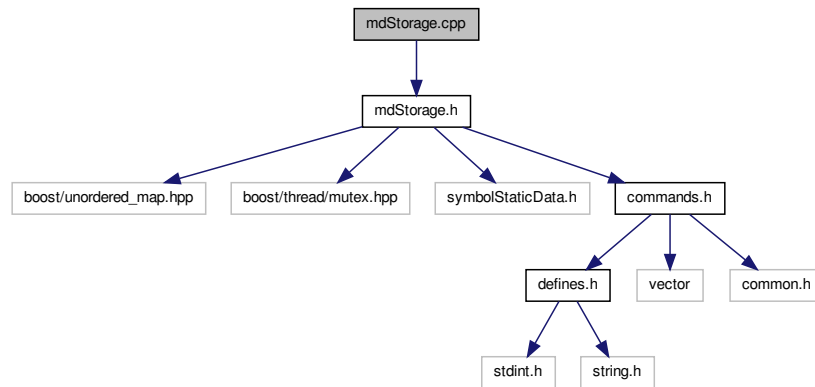
Namespaces

- namespace [MarketData](#)

7.7 mdStorage.cpp File Reference

```
#include "mdStorage.h"
```

Include dependency graph for mdStorage.cpp:



Namespaces

- namespace [MarketData](#)

7.8 mdStorage.h File Reference

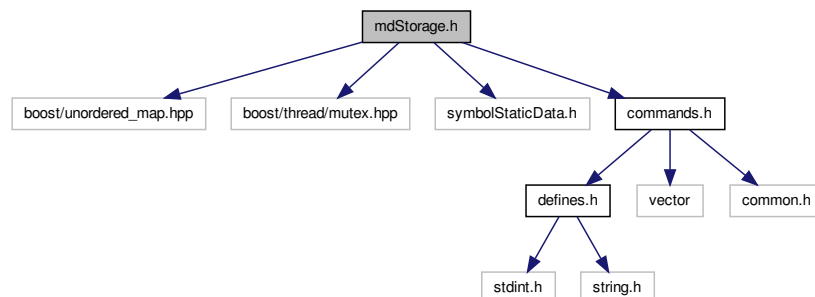
```
#include <boost/unordered_map.hpp>
```

```
#include <boost/thread/mutex.hpp>
```

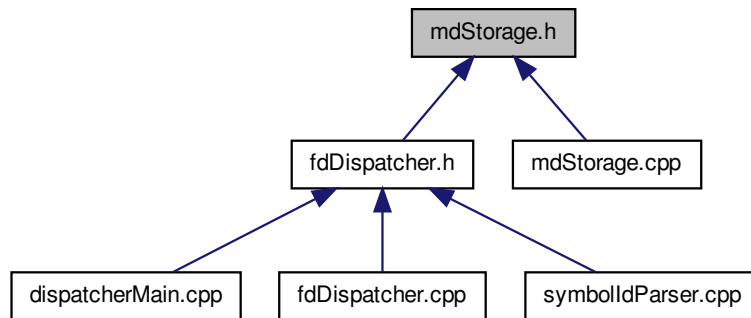
```
#include <symbolStaticData.h>
```

```
#include <commands.h>
```

Include dependency graph for mdStorage.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [MarketData::MDStorage](#)
MDStorage Class.

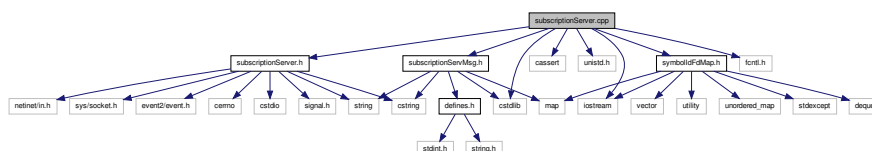
Namespaces

- namespace [MarketData](#)

7.9 subscriptionServer.cpp File Reference

```
#include "subscriptionServer.h"
#include <iostream>
#include <cstdlib>
#include <cassert>
#include <unistd.h>
#include "subscriptionServMsg.h"
#include "symbolIdFdMap.h"
#include <fcntl.h>
```

Include dependency graph for subscriptionServer.cpp:



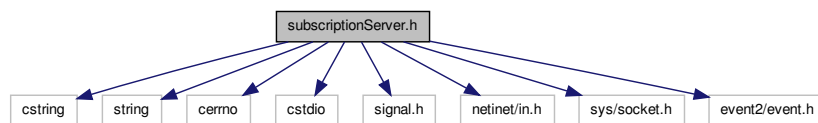
Namespaces

- namespace [MarketData](#)

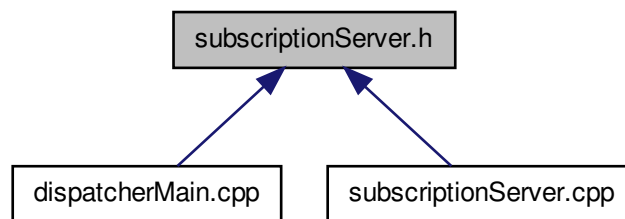
7.10 subscriptionServer.h File Reference

```
#include <cstring>
#include <string>
#include <cerrno>
#include <cstdio>
#include <signal.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <event2/event.h>
```

Include dependency graph for subscriptionServer.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [MarketData::FdState](#)
- class [MarketData::SubscriptionServer](#)

Namespaces

- namespace [MarketData](#)

Macros

- `#define` [MAXLINE](#) 1024
- `#define` [MAXDATABUFFER](#) 16000

7.10.1 Macro Definition Documentation

7.10.1.1 `#define MAXDATABUFFER 16000`

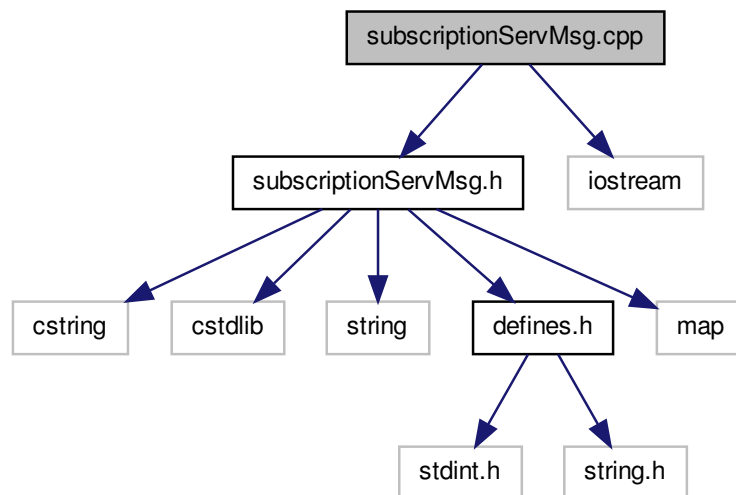
7.10.1.2 `#define MAXLINE 1024`

7.11 subscriptionServMsg.cpp File Reference

```
#include "subscriptionServMsg.h"
```

```
#include <iostream>
```

Include dependency graph for subscriptionServMsg.cpp:



Namespaces

- namespace [MarketData](#)

7.12 subscriptionServMsg.h File Reference

```
#include <cstring>
```

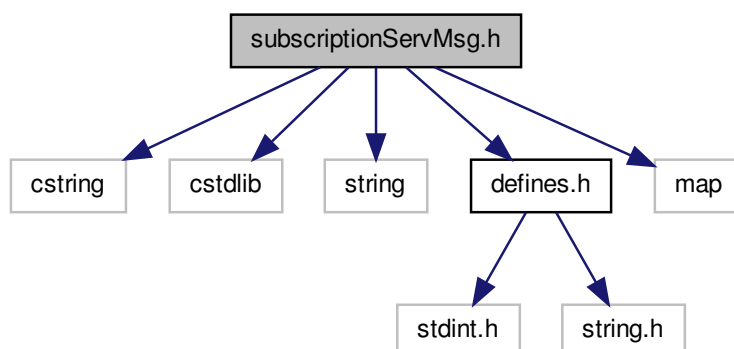
```
#include <cstdlib>
```

```
#include <string>
```

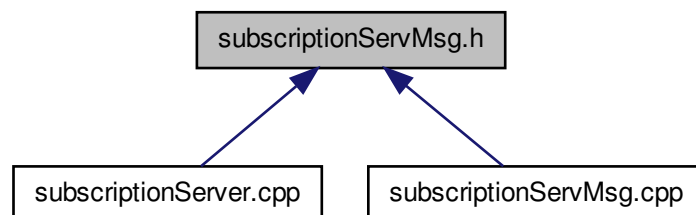
```
#include <defines.h>
```

```
#include <map>
```

Include dependency graph for subscriptionServMsg.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [MarketData::SubscriptionServMsg](#)

This class is responsible for serialization of request data to the server.

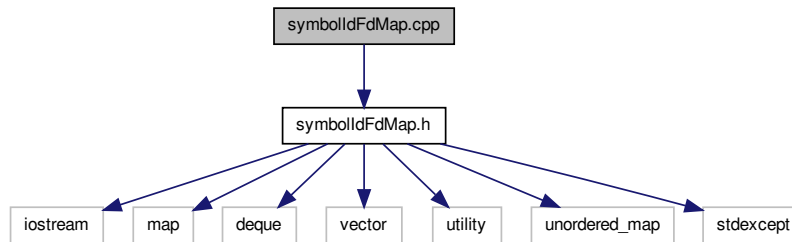
Namespaces

- namespace [MarketData](#)

7.13 symbolIdFdMap.cpp File Reference

```
#include "symbolIdFdMap.h"
```

Include dependency graph for symbolIdFdMap.cpp:



Namespaces

- namespace [MarketData](#)

Functions

- `template<class T >`
`int MarketData::findIndex (T &vec, int id)`

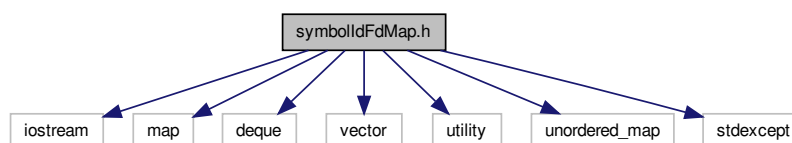
7.14 symbolIdFdMap.h File Reference

```

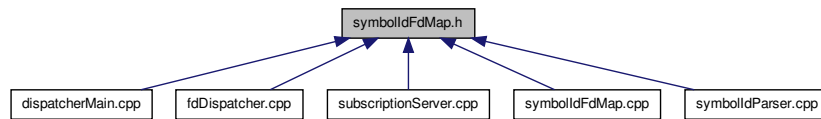
#include <iostream>
#include <map>
#include <deque>
#include <vector>
#include <utility>
#include <unordered_map>
#include <stdexcept>

```

Include dependency graph for symbolIdFdMap.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [MarketData::SymbolIdFdMap](#)
[SymbolIdFdMap](#) class This is the data structure which is used by tbt to send event on each subscriber for each symbol id.
- class [MarketData::SymbolNameIdMap](#)

Namespaces

- namespace [MarketData](#)

Typedefs

- typedef std::unordered_map< int, std::deque< std::pair< int, int > > > [MarketData::IdFdMap](#)
- typedef std::unordered_map< int, std::deque< std::pair< int, int > > >::iterator [MarketData::IdFdMapItr](#)
- typedef std::unordered_map< int, std::vector< int > > [MarketData::FdIdMap](#)
- typedef std::unordered_map< int, std::deque< int > > [MarketData::ConsolidatedIdFdMap](#)

Enumerations

- enum [MarketData::FeedType](#) { [MarketData::TBT](#) = 0, [MarketData::MCL](#) = 1 }

7.15 symbolIdParser.cpp File Reference

```

#include <boost/unordered_map.hpp>
#include <boost/tokenizer.hpp>
#include <boost/foreach.hpp>
#include <string>
#include <fstream>
#include "symbolIdParser.h"
#include "symbolIdFdMap.h"
#include <commands.h>
#include <boost/lexical_cast.hpp>
#include "fdDispatcher.h"

```


Classes

- class [MarketData::SymbolIdParser](#)

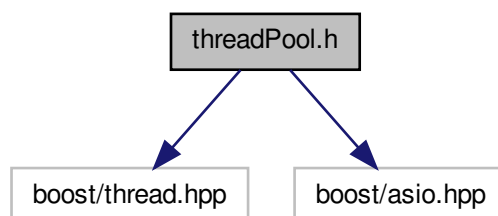
Namespaces

- namespace [MarketData](#)

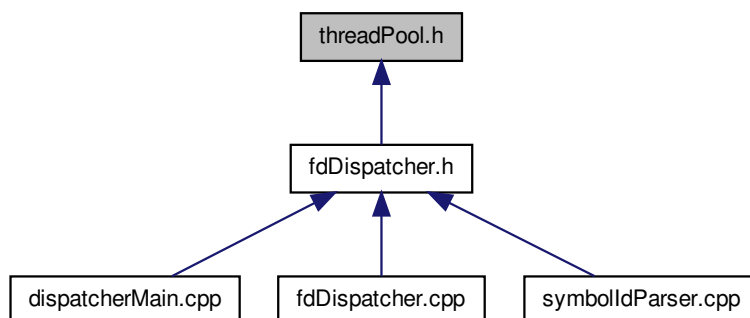
7.17 threadPool.h File Reference

```
#include <boost/thread.hpp>
#include <boost/asio.hpp>
```

Include dependency graph for threadPool.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [MarketData::ThreadPool](#)

Namespaces

- namespace [MarketData](#)

Index

- ~FdDispatcher
 - MarketData::FdDispatcher, [18](#)
- ~ThreadPool
 - MarketData::ThreadPool, [85](#)
- _WINSOCKAPI_
 - defines.cpp, [90](#)
- _askPrice
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _askQty
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _bidPrice
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _bidQty
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _clientId
 - CMD::ScripMasterDataRequest, [57](#)
- _closePrice
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _consolidatedFdIdMap
 - MarketData::SymbolIdFdMap, [80](#)
- _consolidatedIdFdMap
 - MarketData::SymbolIdFdMap, [80](#)
- _depth
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
 - MarketData::MDStorage, [51](#)
- _exchangeId
 - CMD::ScripMasterDataRequest, [57](#)
 - MarketData::SymbolIdParser, [83](#)
- _expiryDate
 - CMD::ScripMasterDataRequest, [57](#)
- _expiryYearMon
 - CMD::ScripMasterDataRequest, [57](#)
- _fdIdMap
 - MarketData::SymbolIdFdMap, [80](#)
- _fdName
 - MarketData::SubscriptionServMsg, [73](#)
- _feedType
 - MarketData::SubscriptionServMsg, [73](#)
- _highPrice
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _idFdMap
 - MarketData::SymbolIdFdMap, [81](#)
- _ioService
 - MarketData::ThreadPool, [86](#)
- _isConsolidatedFeed
 - MarketData::SubscriptionServMsg, [73](#)
- _isRunningLocally
 - MarketData::SubscriptionServMsg, [73](#)
- _isSubscription
 - CMD::MDSubscribeRequest, [53](#)
 - MarketData::SubscriptionServMsg, [74](#)
- _isUnlinkFifo
 - MarketData::SubscriptionServMsg, [74](#)
- _iter
 - MarketData::SymbolIdFdMap, [81](#)
- _lastTradePrice
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _lastTradeQty
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _lowPrice
 - CMD::MDQuote, [39](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _lowerCktLimit
 - CMD::MDQuote, [39](#)
- _marketName
 - CMD::ScripMasterDataRequest, [57](#)
- _mclUseDispatcher
 - MarketData::FdDispatcher, [31](#)
- _mnemonic
 - CMD::InstrumentRequest, [34](#)
- _numSymbols
 - CMD::MDQuoteConsolidated, [47](#)
- _numberOfDispatcherThread
 - MarketData::FdDispatcher, [31](#)
- _numberOfRecords
 - CMD::ScripMasterDataRequest, [57](#)
- _numberOfTrades
 - CMD::MDQuote, [39](#)
- _openPrice
 - CMD::MDQuote, [40](#)
 - CMD::MDQuoteConsolidated, [47](#)
- _optionMode
 - CMD::ScripMasterDataRequest, [57](#)
- _optionType
 - CMD::ScripMasterDataRequest, [57](#)
- _pendingBuffer
 - MarketData::SubscriptionServer, [63](#)
- _pendingBufferSize
 - MarketData::SubscriptionServer, [63](#)

- `_port`
 - `MarketData::SubscriptionServer`, 63
- `_recordNumber`
 - `CMD::ScripMasterDataRequest`, 57
- `_scripMasterDataRequestType`
 - `CMD::ScripMasterDataRequest`, 57
- `_securityId`
 - `CMD::ScripMasterDataRequest`, 57
- `_securityType`
 - `CMD::ScripMasterDataRequest`, 57
- `_series`
 - `CMD::ScripMasterDataRequest`, 57
- `_strikePrice`
 - `CMD::ScripMasterDataRequest`, 57
- `_symbol`
 - `CMD::ScripMasterDataRequest`, 57
- `_symbolAlias`
 - `CMD::ScripMasterDataRequest`, 57
- `_symbolCount`
 - `MarketData::SubscriptionServMsg`, 74
- `_symbolId`
 - `CMD::MDQuote`, 40
 - `CMD::MDQuoteConsolidated`, 47
 - `CMD::MDSubscribeRequest`, 53
 - `CMD::ScripMasterDataRequest`, 57
 - `MarketData::SubscriptionServMsg`, 74
- `_symbolNameIdMap`
 - `MarketData::SymbolNameIdMap`, 84
- `_symbolQuoteMap`
 - `MarketData::MDStorage`, 51
- `_threads`
 - `MarketData::FdDispatcher`, 31
 - `MarketData::ThreadPool`, 86
- `_totalAskQty`
 - `CMD::MDQuote`, 40
- `_totalBidQty`
 - `CMD::MDQuote`, 40
- `_upperCktLimit`
 - `CMD::MDQuote`, 40
- `_useDispatcherCM`
 - `MarketData::FdDispatcher`, 31
- `_useDispatcherFO`
 - `MarketData::FdDispatcher`, 31
- `_userId`
 - `CMD::MDSubscribeRequest`, 53
- `_value`
 - `CMD::MDQuote`, 40
- `_volume`
 - `CMD::MDQuote`, 40
- `_work`
 - `MarketData::ThreadPool`, 86
- `ACCOUNT_FIELD_SIZE`
 - `defines.h`, 93
- `acceptHandle`
 - `MarketData::SubscriptionServer`, 60
- `addFd`
 - `MarketData::SymbolIdFdMap`, 76
- `addFdInConsolidatedMap`
 - `MarketData::SymbolIdFdMap`, 76
- `allocFdState`
 - `MarketData::SubscriptionServer`, 60
- `buffer`
 - `MarketData::FdState`, 32
- `buffer_used`
 - `MarketData::FdState`, 32
- `CMD`
 - `CommandCategory_API_REPLAY_TRADED_SYMBOLS_REQUEST`, 11
 - `CommandCategory_API_TRADED_SYMBOLS_COUNT_REQUEST`, 11
 - `CommandCategory_BOX_STRATEGY`, 11
 - `CommandCategory_BSE_DISCONNECTED`, 10
 - `CommandCategory_BSE_SNAPSHOTFEED_DISCONNECTED`, 10
 - `CommandCategory_BSEETI_DISCONNECTED`, 11
 - `CommandCategory_CASH_CASH_STRATEGY`, 11
 - `CommandCategory_CHANGE_PASSWORD`, 11
 - `CommandCategory_DUMMY_THREE_LEG_ARBITRAGE_STRATEGY`, 10
 - `CommandCategory_DUMMY_TWO_LEG_STRATEGY`, 10
 - `CommandCategory_FORCE_LOG_OFF_CLIENT`, 11
 - `CommandCategory_FOUR_LEG_BUTTERFLY_STRATEGY`, 11
 - `CommandCategory_GET_CLIENT_CONNECTIVITY_STATUS`, 11
 - `CommandCategory_GET_CONNECTIVITY_STATUS`, 11
 - `CommandCategory_GET_RMS_PARAMETERS`, 11
 - `CommandCategory_GET_TOTAL_ORDER_CONFIRMATIONS`, 11
 - `CommandCategory_GET_TOTAL_RECORDS_SCRIP_MASTER`, 11
 - `CommandCategory_HEART_BEAT_MESSAGE`, 10
 - `CommandCategory_IMPLIED_VOLATILITY_STRATEGY`, 11
 - `CommandCategory_INSTRUMENT_SEARCH_REQUEST`, 11
 - `CommandCategory_LOGIN`, 10
 - `CommandCategory_LOGOUT`, 10
 - `CommandCategory_MARKET_DATA_SUBSCRIPTION`, 11
 - `CommandCategory_MARKET_MAKING_STRATEGY`, 10
 - `CommandCategory_MARKET_MAKING_TURNOVER_STRATEGY`, 10
 - `CommandCategory_MAX`, 11
 - `CommandCategory_MODIFY_STRATEGY`, 11
 - `CommandCategory_NSECDS_DISCONNECTED`, 11

- CommandCategory_NSECM_DISCONNECTED, 10
- CommandCategory_NSECM_SNAPSHOTFEED_DISCONNECTED, 10
- CommandCategory_NSEFO_DISCONNECTED, 10
- CommandCategory_NSEFO_SNAPSHOTFEED_DISCONNECTED, 10
- CommandCategory_PAIRS_CLOSE_POSITION_S_NO_TERMINATE, 11
- CommandCategory_PAIRS_STRATEGY, 11
- CommandCategory_PAUSE_STRATEGY, 10
- CommandCategory_REQUEST_OFFLINE_DATA, 10
- CommandCategory_REQUEST_RMS_CONFIGURATION, 10
- CommandCategory_RMS_GLOBAL_PARAMETERS, 11
- CommandCategory_RMS_LOGIN_PARAMETERS, 11
- CommandCategory_RMS_ORDER_LIMITS, 11
- CommandCategory_RMS_RISK_PERMISSIONS, 11
- CommandCategory_RMS_SECURITY_WISE_LIMITS, 11
- CommandCategory_RMS_UPDATE, 10
- CommandCategory_RUN_STRATEGY, 10
- CommandCategory_SEND_SCRIP_MASTER_DATA, 11
- CommandCategory_SGX_DISCONNECTED, 11
- CommandCategory_SINGLE_ORDER, 10
- CommandCategory_STRATEGY_COMMAND, 10
- CommandCategory_TBT_MARKET_DATA, 11
- CommandCategory_TBTCDS_DISCONNECTED, 11
- CommandCategory_TBTCM_DISCONNECTED, 10
- CommandCategory_TBTFO_DISCONNECTED, 10
- CommandCategory_TERMINATE_SQUAREOFF, 10
- CommandCategory_TERMINATE_STRATEGY, 10
- CommandCategory_THREE_LEG_ARBITRAGE_STRATEGY, 10
- CommandCategory_TWO_LEG_ARBITRAGE_STRATEGY, 10
- CommandCategory_TWO_LEG_THREE_LEG_ARBITRAGE_STRATEGY, 11
- Exchangeld_BSE, 11
- Exchangeld_BSEETI, 11
- Exchangeld_CFH, 11
- Exchangeld_ESMNSE, 11
- Exchangeld_MAX, 11
- Exchangeld_NSE, 11
- Exchangeld_NSECDS, 11
- Exchangeld_SGX, 11
- CMD, 9
- CommandCategory, 10
- Exchangeld, 11
- CMD::InstrumentRequest, 32
 - _mnemonic, 34
 - getMnemonic, 33
 - initialize, 34
 - InstrumentRequest, 33
 - MNEMONIC_LENGTH, 34
 - serialize, 34
 - setMnemonic, 34
- CMD::MDQuote, 34
 - _askPrice, 39
 - _askQty, 39
 - _bidPrice, 39
 - _bidQty, 39
 - _closePrice, 39
 - _depth, 39
 - _highPrice, 39
 - _lastTradePrice, 39
 - _lastTradeQty, 39
 - _lowPrice, 39
 - _lowerCktLimit, 39
 - _numberOfTrades, 39
 - _openPrice, 40
 - _symbolId, 40
 - _totalAskQty, 40
 - _totalBidQty, 40
 - _upperCktLimit, 40
 - _value, 40
 - _volume, 40
- dump, 37
- dumpDepth, 37
- getAskPrice, 37
- getAskQty, 37
- getBidPrice, 37
- getBidQty, 37
- getClosePrice, 37
- getDepth, 37
- getHighPrice, 37
- getLastTradePrice, 37
- getLastTradeQty, 37
- getLowPrice, 37
- getLowerCktLimit, 37
- getNumberOfTrades, 38
- getOpenPrice, 38
- getSymbolId, 38
- getTotalAskQty, 38
- getTotalBidQty, 38
- getUpperCktLimit, 38
- getValue, 38
- getVolume, 38
- initialize, 38
- MDQuote, 37
- serialize, 38
- setAskPrice, 38
- setAskQty, 38
- setBidPrice, 38
- setBidQty, 38

- setClosePrice, [39](#)
- setDepth, [39](#)
- setHighPrice, [39](#)
- setLastTradePrice, [39](#)
- setLastTradeQty, [39](#)
- setLowPrice, [39](#)
- setLowerCktLimit, [39](#)
- setNummberOfTrades, [39](#)
- setOpenPrice, [39](#)
- setSymbolId, [39](#)
- setTotalAskQty, [39](#)
- setTotalBidQty, [39](#)
- setUpperCktLimit, [39](#)
- setValue, [39](#)
- setVolume, [39](#)
- CMD::MDQuoteConsolidated, [40](#)
 - _askPrice, [47](#)
 - _askQty, [47](#)
 - _bidPrice, [47](#)
 - _bidQty, [47](#)
 - _closePrice, [47](#)
 - _depth, [47](#)
 - _highPrice, [47](#)
 - _lastTradePrice, [47](#)
 - _lastTradeQty, [47](#)
 - _lowPrice, [47](#)
 - _numSymbols, [47](#)
 - _openPrice, [47](#)
 - _symbolId, [47](#)
 - dump, [43](#)
 - dumpDepth, [43](#)
 - getAskPrice, [43](#)
 - getAskQty, [43](#)
 - getBidPrice, [43](#)
 - getBidQty, [43](#)
 - getClosePrice, [43](#)
 - getDepth, [43](#)
 - getHighPrice, [43](#)
 - getLastTradePrice, [43](#)
 - getLastTradeQty, [43](#)
 - getLowPrice, [43](#)
 - getNumberOfSymbols, [43](#)
 - getOpenPrice, [43](#)
 - getSymbolId, [43](#)
 - initialize, [43](#)
 - MDQuoteConsolidated, [43](#)
 - serialize, [43](#)
 - setAskPrice, [44](#)
 - setAskQty, [44](#)
 - setBidPrice, [44](#)
 - setBidQty, [44](#)
 - setClosePrice, [45](#)
 - setDepth, [45](#)
 - setHighPrice, [45](#)
 - setLastTradePrice, [45](#)
 - setLastTradeQty, [46](#)
 - setLowPrice, [46](#)
 - setNumberOfSymbols, [46](#)
 - setOpenPrice, [46](#)
 - setSymbolId, [47](#)
 - CMD::MDSubscribeRequest, [51](#)
 - _isSubscription, [53](#)
 - _symbolId, [53](#)
 - _userId, [53](#)
 - getSymbolId, [53](#)
 - getUserId, [53](#)
 - initialize, [53](#)
 - isSubscriptionReq, [53](#)
 - MDSubscribeRequest, [53](#)
 - serialize, [53](#)
 - setSubscriptionReq, [53](#)
 - setSymbolId, [53](#)
 - setUserId, [53](#)
 - TOKEN_LENGTH, [53](#)
 - CMD::ScripMasterDataRequest, [53](#)
 - _clientId, [57](#)
 - _exchangeId, [57](#)
 - _expiryDate, [57](#)
 - _expiryYearMon, [57](#)
 - _marketName, [57](#)
 - _numberOfRecords, [57](#)
 - _optionMode, [57](#)
 - _optionType, [57](#)
 - _recordNumber, [57](#)
 - _scripMasterDataRequestType, [57](#)
 - _securityId, [57](#)
 - _securityType, [57](#)
 - _series, [57](#)
 - _strikePrice, [57](#)
 - _symbol, [57](#)
 - _symbolAlias, [57](#)
 - _symbolId, [57](#)
 - dump, [55](#)
 - getClientId, [55](#)
 - getExchangeId, [55](#)
 - getExpiryDate, [56](#)
 - getExpiryYearMon, [56](#)
 - getMarketName, [56](#)
 - getNumberOfRecords, [56](#)
 - getOptionMode, [56](#)
 - getOptionType, [56](#)
 - getRecordNumber, [56](#)
 - getScripMasterDataRequestType, [56](#)
 - getSecurityId, [56](#)
 - getSecurityType, [56](#)
 - getSeries, [56](#)
 - getStrikePrice, [56](#)
 - getSymbol, [56](#)
 - getSymbolAlias, [56](#)
 - getSymbolId, [56](#)
 - ScripMasterDataRequest, [55](#)
 - serialize, [56](#)
 - setClientId, [56](#)
 - setExchangeId, [56](#)
 - setExpiryDate, [56](#)
 - setExpiryYearMon, [56](#)

- setMarketName, [56](#)
- setNumberOfRecords, [56](#)
- setOptionMode, [56](#)
- setOptionType, [56](#)
- setRecordNumber, [56](#)
- setScripMasterDataRequestType, [56](#)
- setSecurityId, [56](#)
- setSecurityType, [57](#)
- setSeries, [57](#)
- setStrikePrice, [57](#)
- setSymbol, [57](#)
- setSymbolAlias, [57](#)
- setSymbolId, [57](#)
- CommandCategory_API_REPLAY_TRADED_SYMBOLS_REQUEST
 - CMD, [11](#)
- CommandCategory_API_TRADED_SYMBOLS_COUNT_REQUEST
 - CMD, [11](#)
- CommandCategory_BOX_STRATEGY
 - CMD, [11](#)
- CommandCategory_BSE_DISCONNECTED
 - CMD, [10](#)
- CommandCategory_BSE_SNAPSHOTFEED_DISCONNECTED
 - CMD, [10](#)
- CommandCategory_BSEETI_DISCONNECTED
 - CMD, [11](#)
- CommandCategory_CASH_CASH_STRATEGY
 - CMD, [11](#)
- CommandCategory_CHANGE_PASSWORD
 - CMD, [11](#)
- CommandCategory_DUMMY_THREE_LEG_ARBITRAGE_STRATEGY
 - CMD, [10](#)
- CommandCategory_DUMMY_TWO_LEG_STRATEGY
 - CMD, [10](#)
- CommandCategory_FORCE_LOG_OFF_CLIENT
 - CMD, [11](#)
- CommandCategory_FOUR_LEG_BUTTERFLY_STRATEGY
 - CMD, [11](#)
- CommandCategory_GET_CLIENT_CONNECTIVITY_STATUS
 - CMD, [11](#)
- CommandCategory_GET_CONNECTIVITY_STATUS
 - CMD, [11](#)
- CommandCategory_GET_RMS_PARAMETERS
 - CMD, [11](#)
- CommandCategory_GET_TOTAL_ORDER_CONFIRMATIONS
 - CMD, [11](#)
- CommandCategory_GET_TOTAL_RECORDS_SCRIPT_MASTER
 - CMD, [11](#)
- CommandCategory_HEART_BEAT_MESSAGE
 - CMD, [10](#)
- CommandCategory_IMPLIED_VOLATILITY_STRATEGY
 - CMD, [11](#)
- CommandCategory_INSTRUMENT_SEARCH_REQUEST
 - CMD, [11](#)
- CommandCategory_LOGIN
 - CMD, [10](#)
- CommandCategory_LOGOUT
 - CMD, [10](#)
- CommandCategory_MARKET_DATA_SUBSCRIPTION
 - CMD, [11](#)
- CommandCategory_MARKET_MAKING_STRATEGY
 - CMD, [10](#)
- CommandCategory_MARKET_MAKING_TURNOVER_STRATEGY
 - CMD, [10](#)
- CommandCategory_MAX
 - CMD, [11](#)
- CommandCategory_MODIFY_STRATEGY
 - CMD, [11](#)
- CommandCategory_NSECDS_DISCONNECTED
 - CMD, [11](#)
- CommandCategory_NSECM_DISCONNECTED
 - CMD, [10](#)
- CommandCategory_NSECM_SNAPSHOTFEED_DISCONNECTED
 - CMD, [10](#)
- CommandCategory_NSEFO_DISCONNECTED
 - CMD, [10](#)
- CommandCategory_NSEFO_SNAPSHOTFEED_DISCONNECTED
 - CMD, [10](#)
- CommandCategory_PAIRS_CLOSE_POSITIONS_NO_TERMINATE
 - CMD, [11](#)
- CommandCategory_PAIRS_STRATEGY
 - CMD, [11](#)
- CommandCategory_PAUSE_STRATEGY
 - CMD, [10](#)
- CommandCategory_REQUEST_OFFLINE_DATA
 - CMD, [10](#)
- CommandCategory_REQUEST_RMS_CONFIGURATION
 - CMD, [10](#)
- CommandCategory_RMS_GLOBAL_PARAMETERS
 - CMD, [11](#)
- CommandCategory_RMS_LOGIN_PARAMETERS
 - CMD, [11](#)
- CommandCategory_RMS_ORDER_LIMITS
 - CMD, [11](#)
- CommandCategory_RMS_RISK_PERMISSIONS
 - CMD, [11](#)
- CommandCategory_RMS_SECURITY_WISE_LIMITS
 - CMD, [11](#)
- CommandCategory_RMS_UPDATE
 - CMD, [10](#)
- CommandCategory_RUN_STRATEGY

- CMD, [10](#)
- CommandCategory_SEND_SCRIP_MASTER_DATA
 - CMD, [11](#)
- CommandCategory_SGX_DISCONNECTED
 - CMD, [11](#)
- CommandCategory_SINGLE_ORDER
 - CMD, [10](#)
- CommandCategory_STRATEGY_COMMAND
 - CMD, [10](#)
- CommandCategory_TBT_MARKET_DATA
 - CMD, [11](#)
- CommandCategory_TBTCDS_DISCONNECTED
 - CMD, [11](#)
- CommandCategory_TBTCM_DISCONNECTED
 - CMD, [10](#)
- CommandCategory_TBTFO_DISCONNECTED
 - CMD, [10](#)
- CommandCategory_TERMINATE_SQUAREOFF
 - CMD, [10](#)
- CommandCategory_TERMINATE_STRATEGY
 - CMD, [10](#)
- CommandCategory_THREE_LEG_ARBITRAGE_STRATEGY
 - CMD, [10](#)
- CommandCategory_TWO_LEG_ARBITRAGE_STRATEGY
 - CMD, [10](#)
- CommandCategory_TWO_LEG_THREE_LEG_ARBITRAGE_STRATEGY
 - CMD, [11](#)
- CommandCategory
 - CMD, [10](#)
- commands.h, [87](#)
 - EXCHANGE_ID_BASE, [89](#)
 - GET_EXCHANGE_ID, [89](#)
 - GET_SCRIP_CODE, [89](#)
 - GET_SYMBOL_ID, [89](#)
- ConsolidatedIdFdMap
 - MarketData, [12](#)
- createFifo
 - MarketData::FdDispatcher, [18](#)
- DESERIALIZE_16
 - defines.h, [93](#)
- DESERIALIZE_32
 - defines.h, [93](#)
- DESERIALIZE_64
 - defines.h, [93](#)
- DESERIALIZE_8
 - defines.h, [93](#)
- DESERIALIZE_SIGNED_64
 - defines.h, [93](#)
- defines.cpp, [89](#)
 - _WINSOCKAPI_, [90](#)
 - generateUniqueOrderId, [90](#)
 - htonl_32, [90](#)
 - htonl_64, [90](#)
 - htons_16, [90](#)
 - ntohl_32, [91](#)
 - ntohl_64, [91](#)
 - ntohs_16, [91](#)
 - signed_htonl_64, [91](#)
 - signed_ntohl_64, [91](#)
 - uniqueOrderIdMutex, [91](#)
- defines.h, [91](#)
 - ACCOUNT_FIELD_SIZE, [93](#)
 - DESERIALIZE_16, [93](#)
 - DESERIALIZE_32, [93](#)
 - DESERIALIZE_64, [93](#)
 - DESERIALIZE_8, [93](#)
 - DESERIALIZE_SIGNED_64, [93](#)
 - getPacket, [95](#)
 - htonl_32, [95](#)
 - htonl_64, [95](#)
 - htons_16, [95](#)
 - IS_BIG_ENDIAN, [93](#)
 - ISIN_SIZE, [94](#)
 - MARKET_NAME_SIZE, [94](#)
 - ntohl_32, [96](#)
 - ntohl_64, [96](#)
 - ntohs_16, [96](#)
 - PASSWORD_SALT_SIZE, [94](#)
 - PASSWORD_SIZE, [94](#)
 - PRODUCT_CODE_SIZE, [94](#)
 - SCRIP_NAME_SIZE, [94](#)
 - SERIALIZE_16, [94](#)
 - SERIALIZE_32, [94](#)
 - SERIALIZE_64, [94](#)
 - SERIALIZE_8, [94](#)
 - SERIALIZE_SIGNED_64, [94](#)
 - SERIES_SIZE, [94](#)
 - SIGNED_LONG, [95](#)
 - SWAP16, [95](#)
 - SWAP32, [95](#)
 - SWAP64, [95](#)
 - SYMBOL_ALIAS_SIZE, [95](#)
 - SYMBOL_SIZE, [95](#)
 - signed_htonl_64, [96](#)
 - signed_ntohl_64, [96](#)
 - UNSIGNED_CHARACTER, [95](#)
 - UNSIGNED_INTEGER, [95](#)
 - UNSIGNED_LONG, [95](#)
 - UNSIGNED_SHORT, [95](#)
- deleteAllSymbolOfConsolidatedFd
 - MarketData::SymbolIdFdMap, [77](#)
- deleteAllSymbolOfFd
 - MarketData::SymbolIdFdMap, [77](#)
- deleteFd
 - MarketData::SymbolIdFdMap, [77](#)
- deleteFdFromConsolidatedMap
 - MarketData::SymbolIdFdMap, [78](#)
- dispatchFd
 - MarketData::FdDispatcher, [19](#)
- dispatchOnConsolidatedFd
 - MarketData::FdDispatcher, [20](#)
- dispatcherMain.cpp, [96](#)
 - main, [96](#)

- dump
 - CMD::MDQuote, [37](#)
 - CMD::MDQuoteConsolidated, [43](#)
 - CMD::ScripMasterDataRequest, [55](#)
 - MarketData::SubscriptionServMsg, [66](#)
- dumpDepth
 - CMD::MDQuote, [37](#)
 - CMD::MDQuoteConsolidated, [43](#)
- EXCHANGE_ID_BASE
 - commands.h, [89](#)
- Exchangeld_BSE
 - CMD, [11](#)
- Exchangeld_BSEETI
 - CMD, [11](#)
- Exchangeld_CFH
 - CMD, [11](#)
- Exchangeld_ESMNSE
 - CMD, [11](#)
- Exchangeld_MAX
 - CMD, [11](#)
- Exchangeld_NSE
 - CMD, [11](#)
- Exchangeld_NSECDS
 - CMD, [11](#)
- Exchangeld_SGX
 - CMD, [11](#)
- Exchangeld
 - CMD, [11](#)
- FD_LENGTH
 - MarketData::SubscriptionServMsg, [74](#)
- FdDispatcher
 - MarketData::FdDispatcher, [18](#)
- fdDispatcher.cpp, [97](#)
- fdDispatcher.h, [98](#)
- FdIdMap
 - MarketData, [12](#)
- FeedType
 - MarketData, [13](#)
- FeedTypeMap
 - MarketData::SubscriptionServMsg, [66](#)
- fifoReadMcl
 - MarketData::FdDispatcher, [22](#)
- fifoReadTbt
 - MarketData::FdDispatcher, [23](#)
- findIndex
 - MarketData, [13](#)
- freeFdState
 - MarketData::SubscriptionServer, [61](#)
- GET_EXCHANGE_ID
 - commands.h, [89](#)
- GET_SCRIP_CODE
 - commands.h, [89](#)
- GET_SYMBOL_ID
 - commands.h, [89](#)
- generateUniqueOrderId
 - defines.cpp, [90](#)
- get
 - MarketData::MDStorage, [49](#)
- getAskPrice
 - CMD::MDQuote, [37](#)
 - CMD::MDQuoteConsolidated, [43](#)
- getAskQty
 - CMD::MDQuote, [37](#)
 - CMD::MDQuoteConsolidated, [43](#)
- getBidPrice
 - CMD::MDQuote, [37](#)
 - CMD::MDQuoteConsolidated, [43](#)
- getBidQty
 - CMD::MDQuote, [37](#)
 - CMD::MDQuoteConsolidated, [43](#)
- getClientId
 - CMD::ScripMasterDataRequest, [55](#)
- getClosePrice
 - CMD::MDQuote, [37](#)
 - CMD::MDQuoteConsolidated, [43](#)
- getConsolidatedDepthAsk
 - MarketData::FdDispatcher, [24](#)
- getConsolidatedDepthBid
 - MarketData::FdDispatcher, [24](#)
- getCurrentLocalTimeStamp
 - MarketData, [13](#)
- getDepth
 - CMD::MDQuote, [37](#)
 - CMD::MDQuoteConsolidated, [43](#)
- getExchangeld
 - CMD::ScripMasterDataRequest, [55](#)
- getExpiryDate
 - CMD::ScripMasterDataRequest, [56](#)
- getExpiryYearMon
 - CMD::ScripMasterDataRequest, [56](#)
- getFdForSymbol
 - MarketData::SymbolIdFdMap, [78](#)
- getFdForSymbolConsolidated
 - MarketData::SymbolIdFdMap, [79](#)
- getFdName
 - MarketData::SubscriptionServMsg, [67](#)
- getFeedType
 - MarketData::SubscriptionServMsg, [68](#)
- getHighPrice
 - CMD::MDQuote, [37](#)
 - CMD::MDQuoteConsolidated, [43](#)
- getId
 - MarketData::SymbolNameIdMap, [84](#)
- getInstance
 - MarketData::MDStorage, [49](#)
 - MarketData::SymbolIdFdMap, [79](#)
 - MarketData::SymbolNameIdMap, [84](#)
- getIoService
 - MarketData::ThreadPool, [85](#)
- getIsConsolidatedFeed
 - MarketData::SubscriptionServMsg, [68](#)
- getIsRunningLocally
 - MarketData::SubscriptionServMsg, [68](#)
- getLastTradePrice

- CMD::MDQuote, 37
- CMD::MDQuoteConsolidated, 43
- getLastTradeQty
 - CMD::MDQuote, 37
 - CMD::MDQuoteConsolidated, 43
- getLowPrice
 - CMD::MDQuote, 37
 - CMD::MDQuoteConsolidated, 43
- getLowerCktLimit
 - CMD::MDQuote, 37
- getMarketName
 - CMD::ScripMasterDataRequest, 56
- getMclSymbolsFromFile
 - MarketData::SymbolIdParser, 81
- getMclUseDispatcher
 - MarketData::FdDispatcher, 25
- getMnemonic
 - CMD::InstrumentRequest, 33
- getNumberOfRecords
 - CMD::ScripMasterDataRequest, 56
- getNumberOfSymbols
 - CMD::MDQuoteConsolidated, 43
- getNumberOfTrades
 - CMD::MDQuote, 38
- getOpenPrice
 - CMD::MDQuote, 38
 - CMD::MDQuoteConsolidated, 43
- getOptionMode
 - CMD::ScripMasterDataRequest, 56
- getOptionType
 - CMD::ScripMasterDataRequest, 56
- getPacket
 - defines.h, 95
- getRecordNumber
 - CMD::ScripMasterDataRequest, 56
- getScripMasterDataRequestType
 - CMD::ScripMasterDataRequest, 56
- getSecurityId
 - CMD::ScripMasterDataRequest, 56
- getSecurityType
 - CMD::ScripMasterDataRequest, 56
- getSeries
 - CMD::ScripMasterDataRequest, 56
- getStrikePrice
 - CMD::ScripMasterDataRequest, 56
- getSymbol
 - CMD::ScripMasterDataRequest, 56
- getSymbolAlias
 - CMD::ScripMasterDataRequest, 56
- getSymbolCount
 - MarketData::SubscriptionServMsg, 69
- getSymbolForFdConsolidated
 - MarketData::SymbolIdFdMap, 80
- getSymbolId
 - CMD::MDQuote, 38
 - CMD::MDQuoteConsolidated, 43
 - CMD::MDSubscribeRequest, 53
 - CMD::ScripMasterDataRequest, 56
- MarketData::SubscriptionServMsg, 69
- getTotalAskQty
 - CMD::MDQuote, 38
- getTotalBidQty
 - CMD::MDQuote, 38
- getUpperCktLimit
 - CMD::MDQuote, 38
- getUseDispatcherCM
 - MarketData::FdDispatcher, 25
- getUseDispatcherFO
 - MarketData::FdDispatcher, 26
- getUserId
 - CMD::MDSubscribeRequest, 53
- getValue
 - CMD::MDQuote, 38
- getVolume
 - CMD::MDQuote, 38
- htonl_32
 - defines.cpp, 90
 - defines.h, 95
- htonl_64
 - defines.cpp, 90
 - defines.h, 95
- htons_16
 - defines.cpp, 90
 - defines.h, 95
- IS_BIG_ENDIAN
 - defines.h, 93
- ISIN_SIZE
 - defines.h, 94
- IdFdMap
 - MarketData, 12
- IdFdMapltr
 - MarketData, 12
- initialize
 - CMD::InstrumentRequest, 34
 - CMD::MDQuote, 38
 - CMD::MDQuoteConsolidated, 43
 - CMD::MDSubscribeRequest, 53
 - MarketData::SymbolIdFdMap, 80
- initializeSymbolIdNameMap
 - MarketData::SymbolIdParser, 82
- InstrumentRequest
 - CMD::InstrumentRequest, 33
- invokeAcceptHandler
 - MarketData::SubscriptionServer, 61
- invokeFifoReadMcl
 - MarketData::FdDispatcher, 26
- invokeFifoReadTbt
 - MarketData::FdDispatcher, 26
- invokeReadHandler
 - MarketData::SubscriptionServer, 62
- isSubscriptionReq
 - CMD::MDSubscribeRequest, 53
 - MarketData::SubscriptionServMsg, 69
- isUnlinkFifo
 - MarketData::SubscriptionServMsg, 70

- MCL
 - MarketData, 13
- MARKET_NAME_SIZE
 - defines.h, 94
- MAX_LOOKUP_LEVEL
 - CMD::MDQuote, 40
 - CMD::MDQuoteConsolidated, 47
- MAX_ORDER_DEPTH
 - MarketData::MDStorage, 51
- MAX_SYMBOL_COUNT
 - MarketData::SubscriptionServMsg, 74
- MAXDATABUFFER
 - subscriptionServer.h, 102
- MAXLINE
 - subscriptionServer.h, 102
- MDQuote
 - CMD::MDQuote, 37
- MDQuoteConsolidated
 - CMD::MDQuoteConsolidated, 43
- MDStorage
 - MarketData::MDStorage, 49
- MDSubscribeRequest
 - CMD::MDSubscribeRequest, 53
- MNEMONIC_LENGTH
 - CMD::InstrumentRequest, 34
- main
 - dispatcherMain.cpp, 96
- MarketData
 - MCL, 13
 - TBT, 13
- MarketData, 12
 - ConsolidatedIdFdMap, 12
 - FdIdMap, 12
 - FeedType, 13
 - findIndex, 13
 - getCurrentLocalTimeStamp, 13
 - IdFdMap, 12
 - IdFdMapItr, 12
- MarketData::FdDispatcher, 15
 - ~FdDispatcher, 18
 - _mclUseDispatcher, 31
 - _numberOfDispatcherThread, 31
 - _threads, 31
 - _useDispatcherCM, 31
 - _useDispatcherFO, 31
 - createFifo, 18
 - dispatchFd, 19
 - dispatchOnConsolidatedFd, 20
 - FdDispatcher, 18
 - fifoReadMcl, 22
 - fifoReadTbt, 23
 - getConsolidatedDepthAsk, 24
 - getConsolidatedDepthBid, 24
 - getMclUseDispatcher, 25
 - getUseDispatcherCM, 25
 - getUseDispatcherFO, 26
 - invokeFifoReadMcl, 26
 - invokeFifoReadTbt, 26
 - openFifo, 27
 - operator(), 27
 - setMclUseDispatcher, 28
 - setUseDispatcherCM, 28
 - setUseDispatcherFO, 29
 - writeOnFd, 29, 30
- MarketData::FdState, 31
 - buffer, 32
 - buffer_used, 32
 - readEvent, 32
- MarketData::MDStorage, 48
 - _depth, 51
 - _symbolQuoteMap, 51
 - get, 49
 - getInstance, 49
 - MAX_ORDER_DEPTH, 51
 - MDStorage, 49
 - quoteToMDQuote, 50
 - save, 50
 - SymbolQuoteMap, 49
- MarketData::SubscriptionServMsg, 64
 - _fdName, 73
 - _feedType, 73
 - _isConsolidatedFeed, 73
 - _isRunningLocally, 73
 - _isSubscription, 74
 - _isUnlinkFifo, 74
 - _symbolCount, 74
 - _symbolId, 74
 - dump, 66
 - FD_LENGTH, 74
 - FeedTypeMap, 66
 - getFdName, 67
 - getFeedType, 68
 - getIsConsolidatedFeed, 68
 - getIsRunningLocally, 68
 - getSymbolCount, 69
 - getSymbolId, 69
 - isSubscriptionReq, 69
 - isUnlinkFifo, 70
 - serialize, 70
 - setFdName, 71
 - setFeedType, 71
 - setIsConsolidatedFeed, 72
 - setIsRunningLocally, 72
 - setSubscriptionReq, 72
 - setSymbolCount, 72
 - setSymbolId, 73
 - setUnlinkFifo, 73
 - SubscriptionServMsg, 66
 - SymbolIdMap, 66
- MarketData::SubscriptionServer, 58
 - _pendingBuffer, 63
 - _pendingBufferSize, 63
 - _port, 63
 - acceptHandle, 60
 - allocFdState, 60
 - freeFdState, 61

- invokeAcceptHandler, 61
- invokeReadHandler, 62
- operator(), 62
- readHandle, 63
- SubscriptionServer, 60
- MarketData::SymbolIdFdMap, 74
 - _consolidatedFdIdMap, 80
 - _consolidatedIdFdMap, 80
 - _fdIdMap, 80
 - _idFdMap, 81
 - _iter, 81
 - addFd, 76
 - addFdInConsolidatedMap, 76
 - deleteAllSymbolOfConsolidatedFd, 77
 - deleteAllSymbolOfFd, 77
 - deleteFd, 77
 - deleteFdFromConsolidatedMap, 78
 - getFdForSymbol, 78
 - getFdForSymbolConsolidated, 79
 - getInstance, 79
 - getSymbolForFdConsolidated, 80
 - initialize, 80
- MarketData::SymbolIdParser, 81
 - _exchangeld, 83
 - getMcISymbolsFromFile, 81
 - initializeSymbolIdNameMap, 82
 - setExchangeld, 83
- MarketData::SymbolNameIdMap, 83
 - _symbolNameIdMap, 84
 - getId, 84
 - getInstance, 84
 - setId, 84
 - SymbolNameIdMap, 84
- MarketData::ThreadPool, 85
 - ~ThreadPool, 85
 - _ioService, 86
 - _threads, 86
 - _work, 86
 - getIoService, 85
 - post, 85
 - ThreadPool, 85
- mdStorage.cpp, 99
- mdStorage.h, 99
- ntohl_32
 - defines.cpp, 91
 - defines.h, 96
- ntohl_64
 - defines.cpp, 91
 - defines.h, 96
- ntohs_16
 - defines.cpp, 91
 - defines.h, 96
- openFifo
 - MarketData::FdDispatcher, 27
- operator()
 - MarketData::FdDispatcher, 27
 - MarketData::SubscriptionServer, 62
- PASSWORD_SALT_SIZE
 - defines.h, 94
- PASSWORD_SIZE
 - defines.h, 94
- PRODUCT_CODE_SIZE
 - defines.h, 94
- post
 - MarketData::ThreadPool, 85
- QUOTEDATA_LENGTH
 - CMD::MDQuote, 40
 - CMD::MDQuoteConsolidated, 48
- quoteToMDQuote
 - MarketData::MDStorage, 50
- readEvent
 - MarketData::FdState, 32
- readHandle
 - MarketData::SubscriptionServer, 63
- SCRIP_NAME_SIZE
 - defines.h, 94
- SERIALIZE_16
 - defines.h, 94
- SERIALIZE_32
 - defines.h, 94
- SERIALIZE_64
 - defines.h, 94
- SERIALIZE_8
 - defines.h, 94
- SERIALIZE_SIGNED_64
 - defines.h, 94
- SERIES_SIZE
 - defines.h, 94
- SIGNED_LONG
 - defines.h, 95
- SWAP16
 - defines.h, 95
- SWAP32
 - defines.h, 95
- SWAP64
 - defines.h, 95
- SYMBOL_ALIAS_SIZE
 - defines.h, 95
- SYMBOL_SIZE
 - defines.h, 95
- save
 - MarketData::MDStorage, 50
- ScripMasterDataRequest
 - CMD::ScripMasterDataRequest, 55
- serialize
 - CMD::InstrumentRequest, 34
 - CMD::MDQuote, 38
 - CMD::MDQuoteConsolidated, 43
 - CMD::MDSubscribeRequest, 53
 - CMD::ScripMasterDataRequest, 56
 - MarketData::SubscriptionServMsg, 70
- setAskPrice
 - CMD::MDQuote, 38

- CMD::MDQuoteConsolidated, 44
- setAskQty
 - CMD::MDQuote, 38
 - CMD::MDQuoteConsolidated, 44
- setBidPrice
 - CMD::MDQuote, 38
 - CMD::MDQuoteConsolidated, 44
- setBidQty
 - CMD::MDQuote, 38
 - CMD::MDQuoteConsolidated, 44
- setClientId
 - CMD::ScripMasterDataRequest, 56
- setClosePrice
 - CMD::MDQuote, 39
 - CMD::MDQuoteConsolidated, 45
- setDepth
 - CMD::MDQuote, 39
 - CMD::MDQuoteConsolidated, 45
- setExchanged
 - CMD::ScripMasterDataRequest, 56
 - MarketData::SymbolIdParser, 83
- setExpiryDate
 - CMD::ScripMasterDataRequest, 56
- setExpiryYearMon
 - CMD::ScripMasterDataRequest, 56
- setFdName
 - MarketData::SubscriptionServMsg, 71
- setFeedType
 - MarketData::SubscriptionServMsg, 71
- setHighPrice
 - CMD::MDQuote, 39
 - CMD::MDQuoteConsolidated, 45
- setId
 - MarketData::SymbolNameIdMap, 84
- setIsConsolidatedFeed
 - MarketData::SubscriptionServMsg, 72
- setIsRunningLocally
 - MarketData::SubscriptionServMsg, 72
- setLastTradePrice
 - CMD::MDQuote, 39
 - CMD::MDQuoteConsolidated, 45
- setLastTradeQty
 - CMD::MDQuote, 39
 - CMD::MDQuoteConsolidated, 46
- setLowPrice
 - CMD::MDQuote, 39
 - CMD::MDQuoteConsolidated, 46
- setLowerCktLimit
 - CMD::MDQuote, 39
- setMarketName
 - CMD::ScripMasterDataRequest, 56
- setMclUseDispatcher
 - MarketData::FdDispatcher, 28
- setMnemonic
 - CMD::InstrumentRequest, 34
- setNumberOfRecords
 - CMD::ScripMasterDataRequest, 56
- setNumberOfSymbols
 - CMD::MDQuoteConsolidated, 46
- setNumberOfTrades
 - CMD::MDQuote, 39
- setOpenPrice
 - CMD::MDQuote, 39
 - CMD::MDQuoteConsolidated, 46
- setOptionMode
 - CMD::ScripMasterDataRequest, 56
- setOptionType
 - CMD::ScripMasterDataRequest, 56
- setRecordNumber
 - CMD::ScripMasterDataRequest, 56
- setScripMasterDataRequestType
 - CMD::ScripMasterDataRequest, 56
- setSecurityId
 - CMD::ScripMasterDataRequest, 56
- setSecurityType
 - CMD::ScripMasterDataRequest, 57
- setSeries
 - CMD::ScripMasterDataRequest, 57
- setStrikePrice
 - CMD::ScripMasterDataRequest, 57
- setSubscriptionReq
 - CMD::MDSubscribeRequest, 53
 - MarketData::SubscriptionServMsg, 72
- setSymbol
 - CMD::ScripMasterDataRequest, 57
- setSymbolAlias
 - CMD::ScripMasterDataRequest, 57
- setSymbolCount
 - MarketData::SubscriptionServMsg, 72
- setSymbolId
 - CMD::MDQuote, 39
 - CMD::MDQuoteConsolidated, 47
 - CMD::MDSubscribeRequest, 53
 - CMD::ScripMasterDataRequest, 57
 - MarketData::SubscriptionServMsg, 73
- setTotalAskQty
 - CMD::MDQuote, 39
- setTotalBidQty
 - CMD::MDQuote, 39
- setUnlinkFifo
 - MarketData::SubscriptionServMsg, 73
- setUpperCktLimit
 - CMD::MDQuote, 39
- setUseDispatcherCM
 - MarketData::FdDispatcher, 28
- setUseDispatcherFO
 - MarketData::FdDispatcher, 29
- setUserId
 - CMD::MDSubscribeRequest, 53
- setValue
 - CMD::MDQuote, 39
- setVolume
 - CMD::MDQuote, 39
- signed_htonl_64
 - defines.cpp, 91
 - defines.h, 96

- signed_ntohl_64
 - defines.cpp, [91](#)
 - defines.h, [96](#)
- Subscription, [58](#)
- SubscriptionServMsg
 - MarketData::SubscriptionServMsg, [66](#)
- subscriptionServMsg.cpp, [102](#)
- subscriptionServMsg.h, [102](#)
- SubscriptionServer
 - MarketData::SubscriptionServer, [60](#)
- subscriptionServer.cpp, [100](#)
- subscriptionServer.h, [101](#)
 - MAXDATABUFFER, [102](#)
 - MAXLINE, [102](#)
- symbolIdFdMap.cpp, [103](#)
- symbolIdFdMap.h, [104](#)
- SymbolIdMap
 - MarketData::SubscriptionServMsg, [66](#)
- symbolIdParser.cpp, [105](#)
- symbolIdParser.h, [106](#)
- SymbolNameIdMap
 - MarketData::SymbolNameIdMap, [84](#)
- SymbolQuoteMap
 - MarketData::MDStorage, [49](#)
- TBT
 - MarketData, [13](#)
- TOKEN_LENGTH
 - CMD::MDSubscribeRequest, [53](#)
- ThreadPool
 - MarketData::ThreadPool, [85](#)
- threadPool.h, [107](#)
- UNSIGNED_CHARACTER
 - defines.h, [95](#)
- UNSIGNED_INTEGER
 - defines.h, [95](#)
- UNSIGNED_LONG
 - defines.h, [95](#)
- UNSIGNED_SHORT
 - defines.h, [95](#)
- uniqueOrderIdMutex
 - defines.cpp, [91](#)
- writeOnFd
 - MarketData::FdDispatcher, [29](#), [30](#)