

XTP极速交易系统QuoteAPI

制作者 中泰证券股份有限公司

Contents

1	XTP 极速行情交易系统 Quote API 2.2.30.6	1
2	继承关系索引	3
2.1	类继承关系	3
3	结构体索引	5
3.1	结构体	5
4	文件索引	9
4.1	文件列表	9
5	结构体说明	11
5.1	DemoTestMdSpi类 参考	11
5.1.1	详细描述	12
5.2	OrderBookStruct结构体 参考	13
5.2.1	详细描述	13
5.3	QuoteApi类 参考	13
5.3.1	详细描述	14
5.3.2	成员函数说明	15
5.3.2.1	CreateQuoteApi()	15
5.3.2.2	GetApiLastError()	15
5.3.2.3	GetApiVersion()	16
5.3.2.4	GetTradingDay()	16
5.3.2.5	Login()	16
5.3.2.6	Logout()	17

5.3.2.7	QueryAllTickers()	17
5.3.2.8	QueryAllTickersFullInfo()	17
5.3.2.9	QueryAllTickersPriceInfo()	18
5.3.2.10	QueryTickersPriceInfo()	18
5.3.2.11	RegisterSpi()	19
5.3.2.12	Release()	19
5.3.2.13	SetHeartBeatInterval()	19
5.3.2.14	SetUDPBufferSize()	19
5.3.2.15	SetUDPParseThreadAffinity()	20
5.3.2.16	SetUDPParseThreadAffinityArray()	20
5.3.2.17	SetUDPRecvThreadAffinity()	21
5.3.2.18	SetUDPRecvThreadAffinityArray()	21
5.3.2.19	SetUDPSeqLogOutPutFlag()	21
5.3.2.20	SubscribeAllMarketData()	22
5.3.2.21	SubscribeAllOptionMarketData()	22
5.3.2.22	SubscribeAllOptionOrderBook()	23
5.3.2.23	SubscribeAllOptionTickByTick()	23
5.3.2.24	SubscribeAllOrderBook()	24
5.3.2.25	SubscribeAllTickByTick()	24
5.3.2.26	SubscribeMarketData()	25
5.3.2.27	SubscribeOrderBook()	25
5.3.2.28	SubscribeTickByTick()	26
5.3.2.29	UnSubscribeAllMarketData()	26
5.3.2.30	UnSubscribeAllOptionMarketData()	27
5.3.2.31	UnSubscribeAllOptionOrderBook()	27
5.3.2.32	UnSubscribeAllOptionTickByTick()	28
5.3.2.33	UnSubscribeAllOrderBook()	28
5.3.2.34	UnSubscribeAllTickByTick()	29
5.3.2.35	UnSubscribeMarketData()	29
5.3.2.36	UnSubscribeOrderBook()	30

5.3.2.37	UnSubscribeTickByTick()	30
5.4	QuoteSpi类 参考	31
5.4.1	详细描述	32
5.4.2	成员函数说明	32
5.4.2.1	OnDepthMarketData()	32
5.4.2.2	OnDisconnected()	33
5.4.2.3	OnError()	33
5.4.2.4	OnOrderBook()	34
5.4.2.5	OnQueryAllTickers()	34
5.4.2.6	OnQueryAllTickersFullInfo()	34
5.4.2.7	OnQueryTickersPriceInfo()	35
5.4.2.8	OnSubMarketData()	35
5.4.2.9	OnSubOrderBook()	36
5.4.2.10	OnSubscribeAllMarketData()	36
5.4.2.11	OnSubscribeAllOptionMarketData()	37
5.4.2.12	OnSubscribeAllOptionOrderBook()	37
5.4.2.13	OnSubscribeAllOptionTickByTick()	37
5.4.2.14	OnSubscribeAllOrderBook()	39
5.4.2.15	OnSubscribeAllTickByTick()	39
5.4.2.16	OnSubTickByTick()	40
5.4.2.17	OnTickByTick()	40
5.4.2.18	OnUnSubMarketData()	41
5.4.2.19	OnUnSubOrderBook()	41
5.4.2.20	OnUnSubscribeAllMarketData()	42
5.4.2.21	OnUnSubscribeAllOptionMarketData()	42
5.4.2.22	OnUnSubscribeAllOptionOrderBook()	43
5.4.2.23	OnUnSubscribeAllOptionTickByTick()	43
5.4.2.24	OnUnSubscribeAllOrderBook()	44
5.4.2.25	OnUnSubscribeAllTickByTick()	44
5.4.2.26	OnUnSubTickByTick()	45

5.5	XTPClientQueryCrdDebtStockReq结构体 参考	45
5.5.1	详细描述	45
5.6	XTPClientQueryCrdPositionStkInfo结构体 参考	46
5.6.1	详细描述	46
5.7	XTPClientQueryCrdPositionStockReq结构体 参考	46
5.7.1	详细描述	47
5.8	XTPClientQueryCrdSurplusStkReqInfo结构体 参考	47
5.8.1	详细描述	47
5.9	XTPClientQueryCrdSurplusStkRspInfo结构体 参考	47
5.9.1	详细描述	48
5.10	XTPCombLegStrategy结构体 参考	48
5.10.1	详细描述	48
5.11	XTPCrdCashRepayDebtInterestFeeRsp结构体 参考	48
5.11.1	详细描述	49
5.12	XTPCrdCashRepayInfo结构体 参考	49
5.12.1	详细描述	49
5.13	XTPCrdCashRepayRsp结构体 参考	50
5.13.1	详细描述	50
5.14	XTPCrdDebtInfo结构体 参考	50
5.14.1	详细描述	51
5.15	XTPCrdDebtStockInfo结构体 参考	51
5.15.1	详细描述	52
5.16	XTPCrdFundExtraInfo结构体 参考	52
5.16.1	详细描述	52
5.17	XTPCrdFundInfo结构体 参考	52
5.17.1	详细描述	53
5.18	XTPCrdPositionExtraInfo结构体 参考	53
5.18.1	详细描述	53
5.19	XTPCreditDebtExtendNotice结构体 参考	53
5.19.1	详细描述	54

5.20 XTPCreditDebtExtendReq结构体 参考	54
5.20.1 详细描述	54
5.21 XTPFundQueryReq结构体 参考	54
5.21.1 详细描述	55
5.22 XTPFundQueryRsp结构体 参考	55
5.22.1 详细描述	55
5.23 XTPFundTransferNotice结构体 参考	55
5.23.1 详细描述	56
5.24 XTPFundTransferReq结构体 参考	56
5.24.1 详细描述	56
5.25 XTPMarketDataOptionExData结构体 参考	57
5.25.1 详细描述	57
5.26 XTPMarketDataStockExData结构体 参考	57
5.26.1 详细描述	59
5.27 XTPMarketDataStruct结构体 参考	59
5.27.1 详细描述	60
5.28 XTPOptCombLegInfo结构体 参考	60
5.28.1 详细描述	61
5.29 XTPOptCombOrderInfo结构体 参考	61
5.29.1 详细描述	62
5.29.2 结构体成员变量说明	62
5.29.2.1 market	62
5.30 XTPOptCombOrderInsertInfo结构体 参考	62
5.30.1 详细描述	63
5.31 XTPOptCombPlugin结构体 参考	63
5.31.1 详细描述	63
5.32 XTPOptCombTradeReport结构体 参考	64
5.32.1 详细描述	64
5.33 XTPOrderCancelInfo结构体 参考	65
5.33.1 详细描述	65

5.34 XTPOrderInfo结构体 参考	65
5.34.1 详细描述	66
5.35 XTPOrderInsertInfo结构体 参考	66
5.35.1 详细描述	67
5.36 XTPQueryAssetRsp结构体 参考	67
5.36.1 详细描述	69
5.37 XTPQueryCombineStrategyInfoRsp结构体 参考	69
5.37.1 详细描述	69
5.38 XTPQueryETFBaseReq结构体 参考	70
5.38.1 详细描述	70
5.39 XTPQueryETFBaseRsp结构体 参考	70
5.39.1 详细描述	71
5.40 XTPQueryETFComponentReq结构体 参考	71
5.40.1 详细描述	71
5.41 XTPQueryETFComponentRsp结构体 参考	71
5.41.1 详细描述	72
5.42 XTPQueryETFComponentRspV1结构体 参考	72
5.42.1 详细描述	73
5.43 XTPQueryFundTransferLogReq结构体 参考	73
5.43.1 详细描述	73
5.44 XTPQueryIPOQuotaRsp结构体 参考	74
5.44.1 详细描述	74
5.45 XTPQueryIPOQuotaRspV1结构体 参考	74
5.45.1 详细描述	74
5.46 XTPQueryIPOTickerRsp结构体 参考	75
5.46.1 详细描述	75
5.47 XTPQueryOptCombExecPosReq结构体 参考	75
5.47.1 详细描述	76
5.48 XTPQueryOptCombExecPosRsp结构体 参考	76
5.48.1 详细描述	77

5.49 XTPQueryOptCombOrderByPageReq结构体 参考	77
5.49.1 详细描述	77
5.50 XTPQueryOptCombOrderReq结构体 参考	77
5.50.1 详细描述	78
5.51 XTPQueryOptCombPositionReq结构体 参考	78
5.51.1 详细描述	78
5.52 XTPQueryOptCombPositionRsp结构体 参考	78
5.52.1 详细描述	79
5.53 XTPQueryOptCombReportByExecIdReq结构体 参考	79
5.53.1 详细描述	79
5.54 XTPQueryOptCombTraderByPageReq结构体 参考	80
5.54.1 详细描述	80
5.55 XTPQueryOptCombTraderReq结构体 参考	80
5.55.1 详细描述	80
5.56 XTPQueryOptExecInfoRsp结构体 参考	81
5.56.1 详细描述	81
5.57 XTPQueryOptionAuctionInfoReq结构体 参考	82
5.57.1 详细描述	82
5.58 XTPQueryOptionAuctionInfoRsp结构体 参考	82
5.58.1 详细描述	84
5.59 XTPQueryOrderByPageReq结构体 参考	84
5.59.1 详细描述	84
5.60 XTPQueryOrderReq结构体 参考	84
5.60.1 详细描述	85
5.61 XTPQueryReportByExecIdReq结构体 参考	85
5.61.1 详细描述	85
5.62 XTPQueryStkPositionReq结构体 参考	85
5.62.1 详细描述	86
5.63 XTPQueryStkPositionRsp结构体 参考	86
5.63.1 详细描述	87

5.64 XTPQueryStructuredFundInfoReq结构体 参考	87
5.64.1 详细描述	87
5.65 XTPQueryTraderByPageReq结构体 参考	88
5.65.1 详细描述	88
5.66 XTPQueryTraderReq结构体 参考	88
5.66.1 详细描述	88
5.67 XTPQuoteFullInfo结构体 参考	89
5.67.1 详细描述	90
5.68 XTPQuoteStaticInfo结构体 参考	90
5.68.1 详细描述	91
5.69 XTPRspInfoStruct结构体 参考	91
5.69.1 详细描述	91
5.70 XTPSpecificTickerStruct结构体 参考	91
5.70.1 详细描述	92
5.71 XTPStructuredFundInfo结构体 参考	92
5.71.1 详细描述	92
5.72 XTPTickByTickEntrust结构体 参考	93
5.72.1 详细描述	93
5.72.2 结构体成员变量说明	93
5.72.2.1 ord_type	93
5.72.2.2 order_no	93
5.72.2.3 qty	93
5.72.2.4 seq	94
5.72.2.5 side	94
5.73 XTPTickByTickStruct结构体 参考	94
5.73.1 详细描述	94
5.73.2 结构体成员变量说明	95
5.73.2.1 seq	95
5.74 XTPTickByTickTrade结构体 参考	95
5.74.1 详细描述	95
5.74.2 结构体成员变量说明	95
5.74.2.1 seq	96
5.74.2.2 trade_flag	96
5.75 XTPTickerPricInfo结构体 参考	96
5.75.1 详细描述	96
5.76 XTPTradeReport结构体 参考	96
5.76.1 详细描述	98
5.77 XTPUserTerminalInfoReq结构体 参考	98
5.77.1 详细描述	98

6 文件说明	99
6.1 demo_test_quote_api.cpp 文件参考	99
6.1.1 详细描述	100
6.1.2 函数说明	100
6.1.2.1 main()	100
6.1.3 变量说明	100
6.1.3.1 ticker_count	101
6.2 demo_test_quote_spi.h 文件参考	101
6.2.1 详细描述	101
6.3 xoms_api_fund_struct.h 文件参考	101
6.3.1 详细描述	102
6.4 xoms_api_struct.h 文件参考	102
6.4.1 详细描述	106
6.5 xquote_api_struct.h 文件参考	106
6.5.1 详细描述	107
6.6 xtp_api_data_type.h 文件参考	107
6.6.1 详细描述	114
6.6.2 宏定义说明	114
6.6.2.1 XTP_SIDE_STOCK_REPAY_STOCK	114
6.6.3 枚举类型说明	115
6.6.3.1 ETF_REPLACE_TYPE	115
6.6.3.2 XTP_ACCOUNT_TYPE	115
6.6.3.3 XTP_AUTO_SPLIT_TYPE	115
6.6.3.4 XTP_BUSINESS_TYPE	116
6.6.3.5 XTP_CRD_CR_STATUS	116
6.6.3.6 XTP_DEBT_EXTEND_OPER_STATUS	118
6.6.3.7 XTP_EXCHANGE_TYPE	118
6.6.3.8 XTP_EXPIRE_DATE_TYPE	118
6.6.3.9 XTP_FUND_OPER_STATUS	119
6.6.3.10 XTP_FUND_QUERY_TYPE	119

6.6.3.11	XTP_FUND_TRANSFER_TYPE	119
6.6.3.12	XTP_LOG_LEVEL	120
6.6.3.13	XTP_MARKET_TYPE	120
6.6.3.14	XTP_OPT_CALL_OR_PUT_TYPE	121
6.6.3.15	XTP_OPT_COVERED_OR_UNCOVERED	121
6.6.3.16	XTP_OPT_EXERCISE_TYPE_TYPE	121
6.6.3.17	XTP_OPT_POSITION_TYPE	121
6.6.3.18	XTP_ORDER_ACTION_STATUS_TYPE	122
6.6.3.19	XTP_ORDER_DETAIL_TYPE	122
6.6.3.20	XTP_ORDER_STATUS_TYPE	122
6.6.3.21	XTP_ORDER_SUBMIT_STATUS_TYPE	123
6.6.3.22	XTP_POSITION_DIRECTION_TYPE	123
6.6.3.23	XTP_POSITION_SECURITY_TYPE	124
6.6.3.24	XTP_PRICE_TYPE	124
6.6.3.25	XTP_PROTOCOL_TYPE	124
6.6.3.26	XTP_QUALIFICATION_TYPE	125
6.6.3.27	XTP_SECURITY_STATUS	125
6.6.3.28	XTP_SECURITY_TYPE	125
6.6.3.29	XTP_SPLIT_MERGE_STATUS	126
6.6.3.30	XTP_TBT_TYPE	126
6.6.3.31	XTP_TE_RESUME_TYPE	127
6.6.3.32	XTP_TICKER_TYPE	127
6.6.3.33	XTP_UNDERLYING_TYPE	127
6.6.3.34	XPTerminalType	128
6.7	xtp_api_struct.h 文件参考	128
6.7.1	详细描述	128
6.8	xtp_api_struct_common.h 文件参考	129
6.8.1	详细描述	129
6.9	xtp_quote_api.h 文件参考	129
6.9.1	详细描述	129

Chapter 1

XTP 极速行情交易系统 Quote API 2.2.30.6

本项目是XTP项目中的行情类接口

(1) XTP的行情订阅接口和响应类 [xtp_quote_api.h](#)

(2) 行情订阅测试Demo [demo_test_quote_api.cpp](#)

Chapter 2

继承关系索引

2.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

OrderBookStruct	13
QuoteApi	13
QuoteSpi	31
DemoTestMdSpi	11
XTPClientQueryCrdDebtStockReq	45
XTPClientQueryCrdPositionStkInfo	46
XTPClientQueryCrdPositionStockReq	46
XTPClientQueryCrdSurplusStkReqInfo	47
XTPClientQueryCrdSurplusStkRspInfo	47
XTPCombLegStrategy	48
XTPCrdCashRepayDebtInterestFeeRsp	48
XTPCrdCashRepayInfo	49
XTPCrdCashRepayRsp	50
XTPCrdDebtInfo	50
XTPCrdDebtStockInfo	51
XTPCrdFundExtraInfo	52
XTPCrdFundInfo	52
XTPCrdPositionExtraInfo	53
XTPCreditDebtExtendNotice	53
XTPCreditDebtExtendReq	54
XTPFundQueryReq	54
XTPFundQueryRsp	55
XTPFundTransferNotice	55
XTPFundTransferReq	56
XTPMarketDataOptionExData	57
XTPMarketDataStockExData	57
XTPMarketDataStruct	59
XTPOptCombLegInfo	60
XTPOptCombOrderInfo	61
XTPOptCombOrderInsertInfo	62
XTPOptCombPlugin	63
XTPOptCombTradeReport	64
XTPOrderCancelInfo	65
XTPOrderInfo	65
XTPOrderInsertInfo	66

XTPQueryAssetRsp	67
XTPQueryCombineStrategyInfoRsp	69
XTPQueryETFBaseReq	70
XTPQueryETFBaseRsp	70
XTPQueryETFComponentReq	71
XTPQueryETFComponentRsp	71
XTPQueryETFComponentRspV1	72
XTPQueryFundTransferLogReq	73
XTPQueryIPOQuotaRsp	74
XTPQueryIPOQuotaRspV1	74
XTPQueryIPOTickerRsp	75
XTPQueryOptCombExecPosReq	75
XTPQueryOptCombExecPosRsp	76
XTPQueryOptCombOrderByPageReq	77
XTPQueryOptCombOrderReq	77
XTPQueryOptCombPositionReq	78
XTPQueryOptCombPositionRsp	78
XTPQueryOptCombReportByExeclIdReq	79
XTPQueryOptCombTraderByPageReq	80
XTPQueryOptCombTraderReq	80
XTPQueryOptExecInfoRsp	81
XTPQueryOptionAuctionInfoReq	82
XTPQueryOptionAuctionInfoRsp	82
XTPQueryOrderByPageReq	84
XTPQueryOrderReq	84
XTPQueryReportByExeclIdReq	85
XTPQueryStkPositionReq	85
XTPQueryStkPositionRsp	86
XTPQueryStructuredFundInfoReq	87
XTPQueryTraderByPageReq	88
XTPQueryTraderReq	88
XTPQuoteFullInfo	89
XTPQuoteStaticInfo	90
XTPRspInfoStruct	91
XTPSpecificTickerStruct	91
XTPStructuredFundInfo	92
XTPTickByTickEntrust	93
XTPTickByTickStruct	94
XTPTickByTickTrade	95
XTPTickerPriceInfo	96
XTPTradeReport	96
XTPUserTerminalInfoReq	98

Chapter 3

结构体索引

3.1 结构体

这里列出了所有结构体，并附带简要说明：

DemoTestMdSpi	
Demo自定义行情订阅接口响应类	11
OrderBookStruct	
定单簿	13
QuoteApi	
行情订阅接口类	13
QuoteSpi	
行情回调类	31
XTPClientQueryCrdDebtStockReq	
融资融券指定证券上的负债未还数量请求结构体	45
XTPClientQueryCrdPositionStkInfo	
融券头寸证券信息	46
XTPClientQueryCrdPositionStockReq	
融券头寸证券查询请求结构体	46
XTPClientQueryCrdSurplusStkReqInfo	
信用业务余券查询请求结构体	47
XTPClientQueryCrdSurplusStkRsplInfo	
信用业务余券信息	47
XTPCombLegStrategy	
期权组合策略的成分合约信息	48
XTPCrdCashRepayDebtInterestFeeRsp	
融资融券现金还息费响应信息	48
XTPCrdCashRepayInfo	
单条融资融券直接还款记录信息	49
XTPCrdCashRepayRsp	
融资融券直接还款响应信息	50
XTPCrdDebtInfo	
单条融资融券负债记录信息	50
XTPCrdDebtStockInfo	
融资融券指定证券的融券负债相关信息	51
XTPCrdFundExtralInfo	
融资融券帐户附加信息	52
XTPCrdFundInfo	
融资融券特有帐户数据	52
XTPCrdPositionExtralInfo	
融资融券帐户持仓附加信息	53

XTPCreditDebtExtendNotice	53
用户展期请求的通知	
XTPCreditDebtExtendReq	54
用户展期请求	
XTPFundQueryReq	54
用户资金查询请求结构体	
XTPFundQueryRsp	55
用户资金查询响应结构体	
XTPFundTransferNotice	55
资金内转流水通知	
XTPFundTransferReq	56
用户资金请求	
XTPMarketDataOptionExData	57
期权额外数据	
XTPMarketDataStockExData	57
股票、基金、债券等额外数据	
XTPMarketDataStruct	59
行情	
XTPOptCombLegInfo	60
组合策略腿合约信息结构体	
XTPOptCombOrderInfo	61
期权组合策略报单响应结构体	
XTPOptCombOrderInsertInfo	62
期权组合策略新订单请求	
XTPOptCombPlugin	63
期权组合策略报单附加信息结构体	
XTPOptCombTradeReport	64
期权组合策略报单成交结构体	
XTPOrderCancelInfo	65
撤单失败响应消息	
XTPOrderInfo	65
报单响应结构体	
XTPOrderInsertInfo	66
新订单请求	
XTPQueryAssetRsp	67
账户资金查询响应结构体	
XTPQueryCombineStrategyInfoRsp	69
查询期权组合策略信息的响应	
XTPQueryETFBaseReq	70
XTPQueryETFBaseRsp	70
查询股票ETF合约基本情况-响应结构体	
XTPQueryETFComponentReq	71
查询股票ETF合约成分股信息-请求结构体,请求参数为:交易市场+ETF买卖代码	
XTPQueryETFComponentRsp	71
查询股票ETF成分股信息-响应结构体	
XTPQueryETFComponentRspV1	72
查询股票ETF成分股信息-响应结构体, 旧版本。	
XTPQueryFundTransferLogReq	73
资金内转流水查询请求与响应	
XTPQueryIPOQuotaRsp	74
查询用户申购额度-包含创业板额度	
XTPQueryIPOQuotaRspV1	74
查询用户申购额度-旧版	
XTPQueryIPOTickerRsp	75
查询当日可申购新股信息	
XTPQueryOptCombExecPosReq	75
查询期权行权合并头寸请求结构体	

XTPQueryOptCombExecPosRsp	76
查询期权行权合并头寸的响应	
XTPQueryOptCombOrderByPageReq	77
查询期权组合策略订单请求-分页查询	
XTPQueryOptCombOrderReq	77
期权组合策略报单查询 ////////////////////////////////// 期权组合策略报单查 询请求-条件查询	
XTPQueryOptCombPositionReq	78
查询期权组合策略持仓情况请求结构体	
XTPQueryOptCombPositionRsp	78
查询期权组合策略持仓信息的响应	
XTPQueryOptCombReportByExecIdReq	79
期权组合策略成交回报查询 ////////////////////////////////// 查询期权组合策 略成交报告请求-根据执行编号查询（保留字段）	
XTPQueryOptCombTraderByPageReq	80
查询期权组合策略成交回报请求-分页查询	
XTPQueryOptCombTraderReq	80
查询期权组合策略成交回报请求-查询条件	
XTPQueryOptExecInfoRsp	81
查询期权合约行权信息的响应	
XTPQueryOptionAuctionInfoReq	82
查询期权竞价交易业务参考信息-请求结构体,请求参数为:交易市场+8位期权代码	
XTPQueryOptionAuctionInfoRsp	82
查询期权竞价交易业务参考信息	
XTPQueryOrderByPageReq	84
查询订单请求-分页查询	
XTPQueryOrderReq	84
报单查询 ////////////////////////////////// 报单查询请求-条件查询	
XTPQueryReportByExecIdReq	85
成交回报查询 ////////////////////////////////// 查询成交报告请求-根据执行编 号查询（保留字段）	
XTPQueryStkPositionReq	85
查询股票持仓情况请求结构体	
XTPQueryStkPositionRsp	86
查询股票持仓情况	
XTPQueryStructuredFundInfoReq	87
查询分级基金信息结构体	
XTPQueryTraderByPageReq	88
查询成交回报请求-分页查询	
XTPQueryTraderReq	88
查询成交回报请求-查询条件	
XTPQuoteFullInfo	89
股票行情全量静态信息	
XTPQuoteStaticInfo	90
股票行情静态信息	
XTPRspInfoStruct	91
响应信息	
XTPSpecificTickerStruct	91
指定的合约	
XTPStructuredFundInfo	92
查询分级基金信息响应结构体	
XTPTickByTickEntrust	93
逐笔委托	
XTPTickByTickStruct	94
逐笔数据信息	
XTPTickByTickTrade	95
逐笔成交	

XTPTickerPriceInfo	
供查询的最新信息	96
XTPTradeReport	
报单成交结构体	96
XTPUserTerminalInfoReq	
申报用户的ip和mac等信息，仅限授权用户使用	98

Chapter 4

文件索引

4.1 文件列表

这里列出了所有文档化的文件，并附带简要说明：

demo_test_quote_api.cpp	定义控制台测试应用程序的入口点	99
demo_test_quote_spi.h	Demo自定义客户端行情订阅响应接口类	101
xoms_api_fund_struct.h	定义资金划拨相关结构体类型	101
xoms_api_struct.h	定义交易类相关数据结构	102
xquote_api_struct.h	定义行情类相关数据结构	106
xtp_api_data_type.h	定义兼容数据基本类型	107
xtp_api_struct.h	定义业务数据结构	128
xtp_api_struct_common.h	定义业务公共数据结构	129
xtp_quote_api.h	定义行情订阅客户端接口	129

Chapter 5

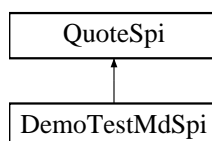
结构体说明

5.1 DemoTestMdSpi类 参考

Demo自定义行情订阅接口响应类

```
#include <demo_test_quote_spi.h>
```

类 DemoTestMdSpi 继承关系图:



Public 成员函数

- virtual void `OnDisconnected` (int reason)
当客户端与行情后台通信连接断开
- virtual void `OnError` (XTPRI *error_info)
错误应答
- virtual void `OnSubMarketData` (XTPST *ticker, XTPRI *error_info, bool is_last)
重写订阅行情应答
- virtual void `OnUnSubMarketData` (XTPST *ticker, XTPRI *error_info, bool is_last)
重写取消订阅行情应答
- virtual void `OnDepthMarketData` (XTPMD *market_data, int64_t bid1_qty[], int32_t bid1_count, int32_t max_bid1_count, int64_t ask1_qty[], int32_t ask1_count, int32_t max_ask1_count)
重写行情通知
- virtual void `OnSubOrderBook` (XTPST *ticker, XTPRI *error_info, bool is_last)
重写订阅行情订单簿应答
- virtual void `OnUnSubOrderBook` (XTPST *ticker, XTPRI *error_info, bool is_last)
重写退订行情订单簿应答
- virtual void `OnSubTickByTick` (XTPST *ticker, XTPRI *error_info, bool is_last)
订阅逐笔行情应答
- virtual void `OnUnSubTickByTick` (XTPST *ticker, XTPRI *error_info, bool is_last)
退订逐笔行情应答

- virtual void [OnOrderBook](#) (XTPOB *order_book)
行情订单簿通知
- virtual void [OnTickByTick](#) (XTPBT *tbt_data)
逐笔行情通知, 包括股票、指数和期权
- virtual void [OnQueryAllTickers](#) (XTPQSI *ticker_info, XTPRI *error_info, bool is_last)
查询可交易合约的应答
- virtual void [OnQueryTickersPriceInfo](#) (XTPPTI *ticker_info, XTPRI *error_info, bool is_last)
查询合约的最新价格信息应答
- virtual void [OnSubscribeAllMarketData](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
订阅全市场的股票行情应答
- virtual void [OnUnSubscribeAllMarketData](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
退订全市场的股票行情应答
- virtual void [OnSubscribeAllOrderBook](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
订阅全市场的股票行情订单簿应答
- virtual void [OnUnSubscribeAllOrderBook](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
退订全市场的股票行情订单簿应答
- virtual void [OnSubscribeAllTickByTick](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
订阅全市场的股票逐笔行情应答
- virtual void [OnUnSubscribeAllTickByTick](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
退订全市场的股票逐笔行情应答
- virtual void [OnSubscribeAllOptionMarketData](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
订阅全市场的期权行情应答
- virtual void [OnUnSubscribeAllOptionMarketData](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
退订全市场的期权行情应答
- virtual void [OnSubscribeAllOptionOrderBook](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
订阅全市场的期权行情订单簿应答
- virtual void [OnUnSubscribeAllOptionOrderBook](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
退订全市场的期权行情订单簿应答
- virtual void [OnSubscribeAllOptionTickByTick](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
订阅全市场的期权逐笔行情应答
- virtual void [OnUnSubscribeAllOptionTickByTick](#) (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
退订全市场的期权逐笔行情应答

5.1.1 详细描述

Demo自定义行情订阅接口响应类

作者

中泰证券股份有限公司

日期

十月 2015

该类的文档由以下文件生成:

- [demo_test_quote_spi.h](#)

5.2 OrderBookStruct结构体 参考

定单簿

```
#include <xquote_api_struct.h>
```

成员变量

- [XTP_EXCHANGE_TYPE](#) `exchange_id`
交易所代码
- `char` [ticker](#) [[XTP_TICKER_LEN](#)]
合约代码（不包含交易所信息），不带空格，以'\0'结尾
- `double` [last_price](#)
最新价
- `int64_t` [qty](#)
数量，为总成交量
- `double` [turnover](#)
成交金额，为总成交金额
- `int64_t` [trades_count](#)
成交笔数
- `double` [bid](#) [10]
十档申买价
- `double` [ask](#) [10]
十档申卖价
- `int64_t` [bid_qty](#) [10]
十档申买量
- `int64_t` [ask_qty](#) [10]
十档申卖量
- `int64_t` [data_time](#)
时间类

5.2.1 详细描述

定单簿

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.3 QuoteApi类 参考

行情订阅接口类

```
#include <xtp_quote_api.h>
```

Public 成员函数

- virtual void **Release** ()=0
- virtual const char * **GetTradingDay** ()=0
- virtual const char * **GetApiVersion** ()=0
- virtual **XTPRI** * **GetApiLastError** ()=0
- virtual void **SetUDPBufferSize** (uint32_t buff_size)=0
- virtual void **RegisterSpi** (QuoteSpi *spi)=0
- virtual void **SetHeartBeatInterval** (uint32_t interval)=0
- virtual void **SetUDPRecvThreadAffinity** (int32_t cpu_no)=0
- virtual void **SetUDPRecvThreadAffinityArray** (int32_t cpu_no_array[], int32_t count)=0
- virtual void **SetUDPParseThreadAffinity** (int32_t cpu_no)=0
- virtual void **SetUDPParseThreadAffinityArray** (int32_t cpu_no_array[], int32_t count)=0
- virtual void **SetUDPSeqLogOutPutFlag** (bool flag=true)=0
- virtual int **SubscribeMarketData** (char *ticker[], int count, **XTP_EXCHANGE_TYPE** exchange_id)=0
- virtual int **UnSubscribeMarketData** (char *ticker[], int count, **XTP_EXCHANGE_TYPE** exchange_id)=0
- virtual int **SubscribeOrderBook** (char *ticker[], int count, **XTP_EXCHANGE_TYPE** exchange_id)=0
- virtual int **UnSubscribeOrderBook** (char *ticker[], int count, **XTP_EXCHANGE_TYPE** exchange_id)=0
- virtual int **SubscribeTickByTick** (char *ticker[], int count, **XTP_EXCHANGE_TYPE** exchange_id)=0
- virtual int **UnSubscribeTickByTick** (char *ticker[], int count, **XTP_EXCHANGE_TYPE** exchange_id)=0
- virtual int **SubscribeAllMarketData** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **UnSubscribeAllMarketData** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **SubscribeAllOrderBook** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **UnSubscribeAllOrderBook** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **SubscribeAllTickByTick** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **UnSubscribeAllTickByTick** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **Login** (const char *ip, int port, const char *user, const char *password, **XTP_PROTOCOL_TYPE** sock_type, const char *local_ip=NULL)=0
- virtual int **Logout** ()=0
- virtual int **QueryAllTickers** (**XTP_EXCHANGE_TYPE** exchange_id)=0
- virtual int **QueryTickersPriceInfo** (char *ticker[], int count, **XTP_EXCHANGE_TYPE** exchange_id)=0
- virtual int **QueryAllTickersPriceInfo** ()=0
- virtual int **SubscribeAllOptionMarketData** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **UnSubscribeAllOptionMarketData** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **SubscribeAllOptionOrderBook** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **UnSubscribeAllOptionOrderBook** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **SubscribeAllOptionTickByTick** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **UnSubscribeAllOptionTickByTick** (**XTP_EXCHANGE_TYPE** exchange_id=**XTP_EXCHANGE_UNKNOWN**)=0
- virtual int **QueryAllTickersFullInfo** (**XTP_EXCHANGE_TYPE** exchange_id)=0

静态 Public 成员函数

- static QuoteApi * **CreateQuoteApi** (uint8_t client_id, const char *save_file_path, **XTP_LOG_LEVEL** log_level=**XTP_LOG_LEVEL_DEBUG**)

5.3.1 详细描述

行情订阅接口类

作者

中泰证券股份有限公司

日期

十月 2015

5.3.2 成员函数说明

5.3.2.1 CreateQuoteApi()

```
static QuoteApi* CreateQuoteApi (
    uint8_t client_id,
    const char * save_file_path,
    XTP_LOG_LEVEL log_level = XTP_LOG_LEVEL_DEBUG ) [static]
```

创建QuoteApi

参数

<i>client_id</i>	(必须输入) 用于区分同一用户的不同客户端，由用户自定义
<i>save_file_path</i>	(必须输入) 存贮订阅信息文件的目录，请设定一个有可写权限的真实存在的路径，如果路径不存在的话，可能会因为写冲突而造成断线
<i>log_level</i>	日志输出级别

返回

创建出的UserApi

备注

如果一个账户需要在多个客户端登录，请使用不同的client_id，系统允许一个账户同时登录多个客户端，但是对于同一账户，相同的client_id只能保持一个session连接，后面的登录在前一个session存续期间，无法连接

5.3.2.2 GetApiLastError()

```
virtual XTPRI* GetApiLastError ( ) [pure virtual]
```

获取API的系统错误

返回

返回的错误信息，可以在Login、Logout、订阅、取消订阅失败时调用，获取失败的原因

备注

可以在调用api接口失败时调用，例如login失败时

5.3.2.3 GetApiVersion()

```
virtual const char* GetApiVersion ( ) [pure virtual]
```

获取API的发行版本号

返回
返回api发行版本号

5.3.2.4 GetTradingDay()

```
virtual const char* GetTradingDay ( ) [pure virtual]
```

获取当前交易日

返回
获取到的交易日

备注
只有登录成功后,才能得到正确的交易日

5.3.2.5 Login()

```
virtual int Login (
    const char * ip,
    int port,
    const char * user,
    const char * password,
    XTP_PROTOCOL_TYPE sock_type,
    const char * local_ip = NULL ) [pure virtual]
```

用户登录请求

返回
登录是否成功，“0”表示登录成功，“-1”表示连接服务器出错，此时用户可以调用GetApiLastError()来获取错误代码，“-2”表示已存在连接，不允许重复登录，如果需要重连，请先logout，“-3”表示输入有错误

参数

<i>ip</i>	服务器ip地址，类似“127.0.0.1”
<i>port</i>	服务器端口号
<i>user</i>	登陆用户名
<i>password</i>	登陆密码
<i>sock_type</i>	“1”代表TCP，“2”代表UDP

备注

此函数为同步阻塞式，不需要异步等待登录成功，当函数返回即可进行后续操作，此api只能有一个连接

5.3.2.6 Logout()

```
virtual int Logout ( ) [pure virtual]
```

登出请求

返回

登出是否成功，“0”表示登出成功，非“0”表示登出出错，此时用户可以调用GetApiLastError()来获取错误代码

备注

此函数为同步阻塞式，不需要异步等待登出，当函数返回即可进行后续操作

5.3.2.7 QueryAllTickers()

```
virtual int QueryAllTickers (
    XTP_EXCHANGE_TYPE exchange_id ) [pure virtual]
```

获取当前交易日合约部分静态信息

返回

发送查询请求是否成功，“0”表示发送查询请求成功，非“0”表示发送查询请求不成功

参数

<code>exchange_id</code>	交易所代码，必须提供 1-上海 2-深圳
--------------------------	----------------------

5.3.2.8 QueryAllTickersFullInfo()

```
virtual int QueryAllTickersFullInfo (
    XTP_EXCHANGE_TYPE exchange_id ) [pure virtual]
```

获取所有合约的详细静态信息，包括指数等非可交易的

返回

发送查询请求是否成功，“0”表示发送查询请求成功，非“0”表示发送查询请求不成功

参数

<i>exchange↵ _id</i>	交易所代码，必须提供 1-上海 2-深圳
--------------------------	----------------------

5.3.2.9 QueryAllTickersPriceInfo()

```
virtual int QueryAllTickersPriceInfo ( ) [pure virtual]
```

获取所有合约的最新价格信息

返回

发送查询请求是否成功，“0”表示发送查询请求成功，非“0”表示发送查询请求不成功

5.3.2.10 QueryTickersPriceInfo()

```
virtual int QueryTickersPriceInfo (
    char * ticker[],
    int count,
    XTP_EXCHANGE_TYPE exchange_id ) [pure virtual]
```

获取合约的最新价格信息

返回

发送查询请求是否成功，“0”表示发送查询请求成功，非“0”表示发送查询请求不成功

参数

<i>ticker</i>	合约ID数组，注意合约代码必须以'\0'结尾，不包含空格
<i>count</i>	要查询的合约个数
<i>exchange↵ _id</i>	交易所代码

5.3.2.11 RegisterSpi()

```
virtual void RegisterSpi (
    QuoteSpi * spi ) [pure virtual]
```

注册回调接口

参数

<i>spi</i>	派生自回调接口类的实例，请在登录之前设定
------------	----------------------

5.3.2.12 Release()

```
virtual void Release ( ) [pure virtual]
```

删除接口对象本身

备注

不再使用本接口对象时,调用该函数删除接口对象

5.3.2.13 SetHeartBeatInterval()

```
virtual void SetHeartBeatInterval (
    uint32_t interval ) [pure virtual]
```

设置心跳检测时间间隔，单位为秒

参数

<i>interval</i>	心跳检测时间间隔，单位为秒
-----------------	---------------

备注

此函数必须在Login之前调用

5.3.2.14 SetUDPBufferSize()

```
virtual void SetUDPBufferSize (
    uint32_t buff_size ) [pure virtual]
```

设置采用UDP方式连接时的单个队列接收缓冲区大小，目前可能最大使用4个缓冲区队列

备注

需要在Login之前调用，默认大小和最小设置均为64MB。此缓存大小单位为MB，请输入2的次方数，例如128MB请输入128。

5.3.2.15 SetUDPParseThreadAffinity()

```
virtual void SetUDPParseThreadAffinity (
    int32_t cpu_no ) [pure virtual]
```

使用UDP接收行情时，设置解析行情线程绑定的cpu，此版本不建议使用，只为跟之前的版本兼容，请替换使用SetUDPParseThreadAffinityArray函数

参数

<i>cpu_no</i>	设置绑定的cpu，例如绑定cpu 0，可以设置0，绑定cpu 2，可以设置2，建议绑定后面的cpu
---------------	---

备注

此版本不建议使用，请替换使用SetUDPParseThreadAffinityArray函数，如果调用则必须在Login之前调用，否则不会生效，与SetUDPParseThreadAffinityArray一起使用时，仅第一个被调用的生效

5.3.2.16 SetUDPParseThreadAffinityArray()

```
virtual void SetUDPParseThreadAffinityArray (
    int32_t cpu_no_array[],
    int32_t count ) [pure virtual]
```

使用UDP接收行情时，设置解析行情线程绑定的cpu集合

参数

<i>cpu_no_array</i>	设置绑定的cpu集合数组
<i>count</i>	cpu集合数组长度

备注

此函数可不调用，如果调用则必须在Login之前调用，否则不会生效。绑核时，将从数组前面的核开始使用

5.3.2.17 SetUDPRecvThreadAffinity()

```
virtual void SetUDPRecvThreadAffinity (
    int32_t cpu_no ) [pure virtual]
```

使用UDP接收行情时，设置接收行情线程绑定的cpu，此版本不建议使用，只为跟之前的版本兼容，请替换使用SetUDPRecvThreadAffinityArray函数

参数

<i>cpu_no</i>	设置绑定的cpu，例如绑定cpu 0，可以设置0，绑定cpu 2，可以设置2，建议绑定后面的cpu
---------------	---

备注

此版本不建议使用,请替换使用SetUDPRecvThreadAffinityArray函数，如果调用则必须在Login之前调用，否则不会生效，与SetUDPRecvThreadAffinityArray一起使用时，仅第一个被调用的生效

5.3.2.18 SetUDPRecvThreadAffinityArray()

```
virtual void SetUDPRecvThreadAffinityArray (
    int32_t cpu_no_array[],
    int32_t count ) [pure virtual]
```

使用UDP接收行情时，设置接收行情线程绑定的cpu集合

参数

<i>cpu_no_array</i>	设置绑定的cpu集合数组
<i>count</i>	cpu集合数组长度

备注

此函数可不调用，如果调用则必须在Login之前调用，否则不会生效。绑核时，将从数组前面的核开始使用

5.3.2.19 SetUDPSeqLogOutPutFlag()

```
virtual void SetUDPSeqLogOutPutFlag (
    bool flag = true ) [pure virtual]
```

设定UDP收行情时是否输出异步日志

参数

<i>flag</i>	是否输出标识，默认为true，如果不想输出“udpseq”开头的异步日志，请设置此参数为false
-------------	---

备注

此函数可不调用，如果调用则必须在Login之前调用，否则不会生效

5.3.2.20 SubscribeAllMarketData()

```
virtual int SubscribeAllMarketData (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

订阅全市场的股票行情

返回

订阅全市场行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<i>exchange_id</i>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------	--

备注

需要与全市场退订行情接口配套使用

5.3.2.21 SubscribeAllOptionMarketData()

```
virtual int SubscribeAllOptionMarketData (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

订阅全市场的期权行情

返回

订阅全市场期权场行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<i>exchange_id</i>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------	--

备注

需要与全市场退订期权行情接口配套使用

5.3.2.22 SubscribeAllOptionOrderBook()

```
virtual int SubscribeAllOptionOrderBook (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

订阅全市场的期权行情订单簿

返回

订阅全市场期权行情订单簿接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

exchange↔ _id	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
------------------	--

备注

需要与全市场退订期权行情订单簿接口配套使用

5.3.2.23 SubscribeAllOptionTickByTick()

```
virtual int SubscribeAllOptionTickByTick (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

订阅全市场的期权逐笔行情

返回

订阅全市场期权逐笔行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

exchange↔ _id	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
------------------	--

备注

需要与全市场退订期权逐笔行情接口配套使用

5.3.2.24 SubscribeAllOrderBook()

```
virtual int SubscribeAllOrderBook (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

订阅全市场的股票行情订单簿

返回

订阅全市场行情订单簿接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<code>exchange_id</code>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------------	--

备注

需要与全市场退订行情订单簿接口配套使用

5.3.2.25 SubscribeAllTickByTick()

```
virtual int SubscribeAllTickByTick (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

订阅全市场的股票逐笔行情

返回

订阅全市场逐笔行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<code>exchange_id</code>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------------	--

备注

需要与全市场退订逐笔行情接口配套使用

5.3.2.26 SubscribeMarketData()

```
virtual int SubscribeMarketData (
    char * ticker[],
    int count,
    XTP_EXCHANGE_TYPE exchange_id ) [pure virtual]
```

订阅行情，包括股票、指数和期权。

返回

订阅接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<i>ticker</i>	合约ID数组，注意合约代码必须以'\0'结尾，不包含空格
<i>count</i>	要订阅/退订行情的合约个数
<i>exchange_id</i>	交易所代码

备注

可以一次性订阅同一证券交易所的多个合约，无论用户因为何种问题需要重新登录行情服务器，都需要重新订阅行情

5.3.2.27 SubscribeOrderBook()

```
virtual int SubscribeOrderBook (
    char * ticker[],
    int count,
    XTP_EXCHANGE_TYPE exchange_id ) [pure virtual]
```

订阅行情订单簿，包括股票、指数和期权。

返回

订阅行情订单簿接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<i>ticker</i>	合约ID数组，注意合约代码必须以'\0'结尾，不包含空格
<i>count</i>	要订阅/退订行情订单簿的合约个数
<i>exchange_id</i>	交易所代码

备注

可以一次性订阅同一证券交易所的多个合约，无论用户因为何种问题需要重新登录行情服务器，都需要重新订阅行情(仅支持深交所)

5.3.2.28 SubscribeTickByTick()

```
virtual int SubscribeTickByTick (
    char * ticker[],
    int count,
    XTP_EXCHANGE_TYPE exchange_id ) [pure virtual]
```

订阅逐笔行情，包括股票、指数和期权。

返回

订阅逐笔行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<i>ticker</i>	合约ID数组，注意合约代码必须以'\0'结尾，不包含空格
<i>count</i>	要订阅/退订行情订单簿的合约个数
<i>exchange_id</i>	交易所代码

备注

可以一次性订阅同一证券交易所的多个合约，无论用户因为何种问题需要重新登录行情服务器，都需要重新订阅行情

5.3.2.29 UnSubscribeAllMarketData()

```
virtual int UnSubscribeAllMarketData (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

退订全市场的股票行情

返回

退订全市场行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<code>exchange_id</code>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------------	---

备注

需要与订阅全市场行情接口配套使用

5.3.2.30 UnSubscribeAllOptionMarketData()

```
virtual int UnSubscribeAllOptionMarketData (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

退订全市场的期权行情

返回

退订全市场期权行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<code>exchange_id</code>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------------	---

备注

需要与订阅全市场期权行情接口配套使用

5.3.2.31 UnSubscribeAllOptionOrderBook()

```
virtual int UnSubscribeAllOptionOrderBook (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

退订全市场的期权行情订单簿

返回

退订全市场期权行情订单簿接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<code>exchange_id</code>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------------	---

备注

需要与订阅全市场期权行情订单簿接口配套使用

5.3.2.32 UnSubscribeAllOptionTickByTick()

```
virtual int UnSubscribeAllOptionTickByTick (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

退订全市场的期权逐笔行情

返回

退订全市场期权逐笔行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<code>exchange_id</code>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------------	---

备注

需要与订阅全市场期权逐笔行情接口配套使用

5.3.2.33 UnSubscribeAllOrderBook()

```
virtual int UnSubscribeAllOrderBook (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

退订全市场的股票行情订单簿

返回

退订全市场行情订单簿接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<code>exchange_id</code>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------------	---

备注

需要与订阅全市场行情订单簿接口配套使用

5.3.2.34 UnSubscribeAllTickByTick()

```
virtual int UnSubscribeAllTickByTick (
    XTP_EXCHANGE_TYPE exchange_id = XTP_EXCHANGE_UNKNOWN ) [pure virtual]
```

退订全市场的股票逐笔行情

返回

退订全市场逐笔行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<code>exchange_id</code>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
--------------------------	---

备注

需要与订阅全市场逐笔行情接口配套使用

5.3.2.35 UnSubscribeMarketData()

```
virtual int UnSubscribeMarketData (
    char * ticker[],
    int count,
    XTP_EXCHANGE_TYPE exchange_id ) [pure virtual]
```

退订行情，包括股票、指数和期权。

返回

取消订阅接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<i>ticker</i>	合约ID数组，注意合约代码必须以'\0'结尾，不包含空格
<i>count</i>	要订阅/退订行情的合约个数
<i>exchange_id</i>	交易所代码

备注

可以一次性取消订阅同一证券交易所的多个合约，需要与订阅行情接口配套使用

5.3.2.36 UnSubscribeOrderBook()

```
virtual int UnSubscribeOrderBook (
    char * ticker[],
    int count,
    XTP_EXCHANGE_TYPE exchange_id ) [pure virtual]
```

退订行情订单簿，包括股票、指数和期权。

返回

取消订阅行情订单簿接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<i>ticker</i>	合约ID数组，注意合约代码必须以'\0'结尾，不包含空格
<i>count</i>	要订阅/退订行情订单簿的合约个数
<i>exchange_id</i>	交易所代码

备注

可以一次性取消订阅同一证券交易所的多个合约，需要与订阅行情订单簿接口配套使用

5.3.2.37 UnSubscribeTickByTick()

```
virtual int UnSubscribeTickByTick (
    char * ticker[],
    int count,
    XTP_EXCHANGE_TYPE exchange_id ) [pure virtual]
```

退订逐笔行情，包括股票、指数和期权。

返回

取消订阅逐笔行情接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

<i>ticker</i>	合约ID数组，注意合约代码必须以'\0'结尾，不包含空格
<i>count</i>	要订阅/退订行情订单簿的合约个数
<i>exchange_id</i>	交易所代码

备注

可以一次性取消订阅同一证券交易所的多个合约，需要与订阅逐笔行情接口配套使用

该类的文档由以下文件生成:

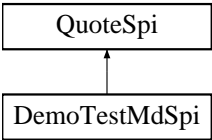
- [xtp_quote_api.h](#)

5.4 QuoteSpi类 参考

行情回调类

```
#include <xtp_quote_api.h>
```

类 QuoteSpi 继承关系图:



Public 成员函数

- virtual void OnDisconnected (int reason)
- virtual void OnError (XTPRI *error_info)
- virtual void OnSubMarketData (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void OnUnSubMarketData (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void OnDepthMarketData (XTPMD *market_data, int64_t bid1_qty[], int32_t bid1_count, int32_t max_bid1_count, int64_t ask1_qty[], int32_t ask1_count, int32_t max_ask1_count)
- virtual void OnSubOrderBook (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void OnUnSubOrderBook (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void OnOrderBook (XTPOB *order_book)
- virtual void OnSubTickByTick (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void OnUnSubTickByTick (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void OnTickByTick (XTPTBT *tbt_data)
- virtual void OnSubscribeAllMarketData (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)

- virtual void OnUnSubscribeAllMarketData (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnSubscribeAllOrderBook (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnUnSubscribeAllOrderBook (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnSubscribeAllTickByTick (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnUnSubscribeAllTickByTick (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnQueryAllTickers (XTPQSI *ticker_info, XTPRI *error_info, bool is_last)
- virtual void OnQueryTickersPriceInfo (XTPTPI *ticker_info, XTPRI *error_info, bool is_last)
- virtual void OnSubscribeAllOptionMarketData (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnUnSubscribeAllOptionMarketData (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnSubscribeAllOptionOrderBook (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnUnSubscribeAllOptionOrderBook (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnSubscribeAllOptionTickByTick (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnUnSubscribeAllOptionTickByTick (XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)
- virtual void OnQueryAllTickersFullInfo (XTPQFI *ticker_info, XTPRI *error_info, bool is_last)

5.4.1 详细描述

行情回调类

作者

中泰证券股份有限公司

日期

十月 2015

5.4.2 成员函数说明

5.4.2.1 OnDepthMarketData()

```
virtual void OnDepthMarketData (  
    XTPMD * market_data,  
    int64_t bid1_qty[],  
    int32_t bid1_count,  
    int32_t max_bid1_count,  
    int64_t ask1_qty[],  
    int32_t ask1_count,  
    int32_t max_ask1_count ) [inline], [virtual]
```

深度行情通知，包含买一卖一队列

参数

market_data	行情数据
bid1_qty	买一队列数据
bid1_count	买一队列的有效委托笔数
max_bid1_count	买一队列总委托笔数
ask1_qty	卖一队列数据
ask1_count	卖一队列的有效委托笔数
max_ask1_count	卖一队列总委托笔数

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestMdSpi](#) 重载.

5.4.2.2 OnDisconnected()

```
virtual void OnDisconnected (
    int reason ) [inline], [virtual]
```

当客户端与行情后台通信连接断开时，该方法被调用。

参数

<i>reason</i>	错误原因，请与错误代码表对应
---------------	----------------

备注

[api](#)不会自动重连，当断线发生时，请用户自行选择后续操作。可以在此函数中调用Login重新登录。注意用户重新登录后，需要重新订阅行情

被 [DemoTestMdSpi](#) 重载.

5.4.2.3 OnError()

```
virtual void OnError (
    XTPRI * error_info ) [inline], [virtual]
```

错误应答

参数

<i>error_info</i>	当服务器响应发生错误时的具体的错误代码和错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
-------------------	--

备注

此函数只有在服务器发生错误时才会调用，一般无需用户处理

被 [DemoTestMdSpi](#) 重载.

5.4.2.4 OnOrderBook()

```
virtual void OnOrderBook (
    XTPOB * order_book ) [inline], [virtual]
```

行情订单簿通知，包括股票、指数和期权

参数

order_book	行情订单簿数据，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线
------------	---------------------------------------

被 [DemoTestMdSpi](#) 重载.

5.4.2.5 OnQueryAllTickers()

```
virtual void OnQueryAllTickers (
    XTPQSI * ticker_info,
    XTPRI * error_info,
    bool is_last ) [inline], [virtual]
```

查询合约部分静态信息的应答

参数

ticker_info	合约部分静态信息
error_info	查询合约部分静态信息时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
is_last	是否此次查询合约部分静态信息的最后一个应答，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应

被 [DemoTestMdSpi](#) 重载.

5.4.2.6 OnQueryAllTickersFullInfo()

```
virtual void OnQueryAllTickersFullInfo (
    XTPQFI * ticker_info,
    XTPRI * error_info,
    bool is_last ) [inline], [virtual]
```

查询合约完整静态信息的应答

参数

ticker_info	合约完整静态信息
error_info	查询合约完整静态信息时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
is_last	是否此次查询合约完整静态信息的最后一个应答，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应

5.4.2.7 OnQueryTickersPriceInfo()

```
virtual void OnQueryTickersPriceInfo (
    XTPPTPI * ticker_info,
    XTPPRI * error_info,
    bool is_last ) [inline], [virtual]
```

查询合约的最新价格信息应答

参数

<i>ticker_info</i>	合约的最新价格信息
<i>error_info</i>	查询合约的最新价格信息时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
<i>is_last</i>	是否此次查询的最后一个应答，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应

被 [DemoTestMdSpi](#) 重载.

5.4.2.8 OnSubMarketData()

```
virtual void OnSubMarketData (
    XTPST * ticker,
    XTPPRI * error_info,
    bool is_last ) [inline], [virtual]
```

订阅行情应答，包括股票、指数和期权

参数

<i>ticker</i>	详细的合约订阅情况
<i>error_info</i>	订阅合约发生错误时的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
<i>is_last</i>	是否此次订阅的最后一个应答，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应

备注

每条订阅的合约均对应一条订阅应答，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestMdSpi](#) 重载.

5.4.2.9 OnSubOrderBook()

```
virtual void OnSubOrderBook (
    XTPST * ticker,
    XTPRI * error_info,
    bool is_last ) [inline], [virtual]
```

订阅行情订单簿应答，包括股票、指数和期权

参数

<i>ticker</i>	详细的合约订阅情况
<i>error_info</i>	订阅合约发生错误时的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
<i>is_last</i>	是否此次订阅的最后一个应答，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应

备注

每条订阅的合约均对应一条订阅应答，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestMdSpi](#) 重载.

5.4.2.10 OnSubscribeAllMarketData()

```
virtual void OnSubscribeAllMarketData (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

订阅全市场的股票行情应答

参数

<i>exchange_id</i>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.11 OnSubscribeAllOptionMarketData()

```
virtual void OnSubscribeAllOptionMarketData (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

订阅全市场的期权行情应答

参数

<i>exchange_id</i>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.12 OnSubscribeAllOptionOrderBook()

```
virtual void OnSubscribeAllOptionOrderBook (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

订阅全市场的期权行情订单簿应答

参数

<i>exchange_id</i>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.13 OnSubscribeAllOptionTickByTick()

```
virtual void OnSubscribeAllOptionTickByTick (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

订阅全市场的期权逐笔行情应答

参数

<i>exchange_id</i>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.14 OnSubscribeAllOrderBook()

```
virtual void OnSubscribeAllOrderBook (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

订阅全市场的股票行情订单簿应答

参数

<i>exchange_id</i>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.15 OnSubscribeAllTickByTick()

```
virtual void OnSubscribeAllTickByTick (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

订阅全市场的股票逐笔行情应答

参数

<i>exchange_id</i>	表示当前全订阅的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.16 OnSubTickByTick()

```
virtual void OnSubTickByTick (
    XTPST * ticker,
    XTPRI * error_info,
    bool is_last ) [inline], [virtual]
```

订阅逐笔行情应答，包括股票、指数和期权

参数

<i>ticker</i>	详细的合约订阅情况
<i>error_info</i>	订阅合约发生错误时的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
<i>is_last</i>	是否此次订阅的最后一个应答，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应

备注

每条订阅的合约均对应一条订阅应答，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestMdSpi](#) 重载.

5.4.2.17 OnTickByTick()

```
virtual void OnTickByTick (
    XTPTBT * tbt_data ) [inline], [virtual]
```

逐笔行情通知，包括股票、指数和期权

参数

<i>tb_t_data</i>	逐笔行情数据，包括逐笔委托和逐笔成交，此为共用结构体，需要根据type来区分是逐笔委托还是逐笔成交，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线
------------------	---

被 [DemoTestMdSpi](#) 重载.

5.4.2.18 OnUnSubMarketData()

```
virtual void OnUnSubMarketData (
    XTPST * ticker,
    XTPRI * error_info,
    bool is_last ) [inline], [virtual]
```

退订行情应答，包括股票、指数和期权

参数

<i>ticker</i>	详细的合约取消订阅情况
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
<i>is_last</i>	是否此次取消订阅的最后一个应答，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应

备注

每条取消订阅的合约均对应一条取消订阅应答，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestMdSpi](#) 重载.

5.4.2.19 OnUnSubOrderBook()

```
virtual void OnUnSubOrderBook (
    XTPST * ticker,
    XTPRI * error_info,
    bool is_last ) [inline], [virtual]
```

退订行情订单簿应答，包括股票、指数和期权

参数

<i>ticker</i>	详细的合约取消订阅情况
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
<i>is_last</i>	是否此次取消订阅的最后一个应答，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
制作者 Doxygen	

备注

每条取消订阅的合约均对应一条取消订阅应答，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestMdSpi](#) 重载.

5.4.2.20 OnUnSubscribeAllMarketData()

```
virtual void OnUnSubscribeAllMarketData (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

退订全市场的股票行情应答

参数

<i>exchange_id</i>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.21 OnUnSubscribeAllOptionMarketData()

```
virtual void OnUnSubscribeAllOptionMarketData (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

退订全市场的期权行情应答

参数

<i>exchange_id</i>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.22 OnUnSubscribeAllOptionOrderBook()

```
virtual void OnUnSubscribeAllOptionOrderBook (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

退订全市场的期权行情订单簿应答

参数

<i>exchange_id</i>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.23 OnUnSubscribeAllOptionTickByTick()

```
virtual void OnUnSubscribeAllOptionTickByTick (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

退订全市场的期权逐笔行情应答

参数

<i>exchange_id</i>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.24 OnUnSubscribeAllOrderBook()

```
virtual void OnUnSubscribeAllOrderBook (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

退订全市场的股票行情订单簿应答

参数

<i>exchange_id</i>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.25 OnUnSubscribeAllTickByTick()

```
virtual void OnUnSubscribeAllTickByTick (
    XTP_EXCHANGE_TYPE exchange_id,
    XTPRI * error_info ) [inline], [virtual]
```

退订全市场的股票逐笔行情应答

参数

<i>exchange_id</i>	表示当前退订的市场，如果为XTP_EXCHANGE_UNKNOWN，表示沪深全市场，XTP_EXCHANGE_SH表示为上海全市场，XTP_EXCHANGE_SZ表示为深圳全市场
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.26 OnUnSubTickByTick()

```
virtual void OnUnSubTickByTick (
    XTPST * ticker,
    XTPRI * error_info,
    bool is_last ) [inline], [virtual]
```

退订逐笔行情应答，包括股票、指数和期权

参数

<i>ticker</i>	详细的合约取消订阅情况
<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
<i>is_last</i>	是否此次取消订阅的最后一个应答，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应

备注

每条取消订阅的合约均对应一条取消订阅应答，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestMdSpi](#) 重载.

该类的文档由以下文件生成:

- [xtp_quote_api.h](#)

5.5 XTPClientQueryCrdDebtStockReq结构体 参考

融资融券指定证券上的负债未还数量请求结构体

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP_MARKET_TYPE](#) market
市场
- [char](#) ticker [[XTP_TICKER_LEN](#)]
证券代码

5.5.1 详细描述

融资融券指定证券上的负债未还数量请求结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.6 XTPClientQueryCrdPositionStkInfo结构体 参考

融券头寸证券信息

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP_MARKET_TYPE market](#)
证券市场
- [char ticker \[XTP_TICKER_LEN\]](#)
证券代码
- [int64_t limit_qty](#)
融券限量(保留字段)
- [int64_t yesterday_qty](#)
昨日日融券数量(保留字段)
- [int64_t left_qty](#)
剩余可融券数量
- [int64_t frozen_qty](#)
冻结融券数量(保留字段)

5.6.1 详细描述

融券头寸证券信息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.7 XTPClientQueryCrdPositionStockReq结构体 参考

融券头寸证券查询请求结构体

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP_MARKET_TYPE market](#)
证券市场
- [char ticker \[XTP_TICKER_LEN\]](#)
证券代码

5.7.1 详细描述

融券头寸证券查询请求结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.8 XTPClientQueryCrdSurplusStkReqInfo结构体 参考

信用业务余券查询请求结构体

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP_MARKET_TYPE market](#)
证券市场
- [char ticker \[XTP_TICKER_LEN\]](#)
证券代码

5.8.1 详细描述

信用业务余券查询请求结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.9 XTPClientQueryCrdSurplusStkRspInfo结构体 参考

信用业务余券信息

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP_MARKET_TYPE market](#)
证券市场
- [char ticker \[XTP_TICKER_LEN\]](#)
证券代码
- [int64_t transferable_quantity](#)
可划转数量
- [int64_t transferred_quantity](#)
已划转数量

5.9.1 详细描述

信用业务余券信息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.10 XTPCombLegStrategy结构体 参考

期权组合策略的成分合约信息

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP_OPT_CALL_OR_PUT_TYPE call_or_put](#)
合约类型，认沽或认购
- [XTP_POSITION_DIRECTION_TYPE position_side](#)
权利仓或者义务仓或备兑义务仓
- [TXTPExerciseSeqType exercise_price_seq](#)
行权价顺序
- [int32_t expire_date_seq](#)
到期日顺序
- [int64_t leg_qty](#)
单份组合策略中包含的此合约张数

5.10.1 详细描述

期权组合策略的成分合约信息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.11 XTPCrdCashRepayDebtInterestFeeRsp结构体 参考

融资融券现金还息费响应信息

```
#include <xoms_api_struct.h>
```

成员变量

- `int64_t xtp_id`
直接还款操作的XTPID
- `double request_amount`
直接还款的申请金额
- `double cash_repay_amount`
实际还款使用金额
- `char debt_compact_id [XTP_CREDIT_DEBT_ID_LEN]`
指定的负债合约编号
- `char unknow [32]`
保留字段

5.11.1 详细描述

融资融券现金还息费响应信息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.12 XTPCrdCashRepayInfo结构体 参考

单条融资融券直接还款记录信息

```
#include <xoms_api_struct.h>
```

成员变量

- `int64_t xtp_id`
直接还款操作的XTPID
- `XTP_CRD_CR_STATUS status`
直接还款处理状态
- `double request_amount`
直接还款的申请金额
- `double cash_repay_amount`
实际还款使用金额
- `XTP_POSITION_EFFECT_TYPE position_effect`
强平标志
- `XTPRI error_info`
直接还款发生错误时的错误信息

5.12.1 详细描述

单条融资融券直接还款记录信息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.13 XTPCrdCashRepayRsp结构体 参考

融资融券直接还款响应信息

```
#include <xoms_api_struct.h>
```

成员变量

- `int64_t xtp_id`
直接还款操作的`XTPID`
- `double request_amount`
直接还款的申请金额
- `double cash_repay_amount`
实际还款使用金额

5.13.1 详细描述

融资融券直接还款响应信息

该结构体的文档由以下文件生成:

- `xoms_api_struct.h`

5.14 XTPCrdDebtInfo结构体 参考

单条融资融券负债记录信息

```
#include <xoms_api_struct.h>
```

成员变量

- `int32_t debt_type`
负债合约类型: `0`为融资, `1`为融券, `2`未知
- `char debt_id [33]`
负债合约编号
- `int64_t position_id`
负债对应两融头寸编号
- `uint64_t order_xtp_id`
生成负债的订单编号, 非当日负债无此项
- `int32_t debt_status`
负债合约状态: `0`为未偿还或部分偿还, `1`为已偿还, `2`为过期未平仓, `3`未知
- `XTP_MARKET_TYPE market`
市场
- `char ticker [XTP_TICKER_LEN]`
证券代码
- `uint64_t order_date`

- 委托日期
- `uint64_t end_date`
负债截止日期
- `uint64_t orig_end_date`
负债原始截止日期
- `bool is_extended`
当日是否接收到展期请求: *false*为没收到, *true*为收到
- `double remain_amt`
未偿还金额
- `int64_t remain_qty`
未偿还融券数量
- `double remain_principal`
未偿还本金金额
- `int64_t due_right_qty`
应偿还权益数量
- `int64_t unknown` [2]
保留字段

5.14.1 详细描述

单条融资融券负债记录信息

该结构体的文档由以下文件生成:

- `xoms_api_struct.h`

5.15 XTPCrdDebtStockInfo结构体 参考

融资融券指定证券的融券负债相关信息

```
#include <xoms_api_struct.h>
```

成员变量

- `XTP_MARKET_TYPE market`
市场
- `char ticker[XTP_TICKER_LEN]`
证券代码
- `int64_t stock_repay_quantity`
融券负债可还券数量
- `int64_t stock_total_quantity`
融券负债未还总数量

5.15.1 详细描述

融资融券指定证券的融券负债相关信息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.16 XTPCrdFundExtraInfo结构体 参考

融资融券帐户附加信息

```
#include <xoms_api_struct.h>
```

成员变量

- double [mf_rs_avl_used](#)
当前资金账户购买货币基金使用的融券卖出所得资金占用
- char [reserve](#) [64]
预留空间

5.16.1 详细描述

融资融券帐户附加信息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.17 XTPCrdFundInfo结构体 参考

融资融券特有帐户数据

```
#include <xoms_api_struct.h>
```

成员变量

- double [maintenance_ratio](#)
维持担保品比例
- double [all_asset](#)
总资产(包含证券资产)
- double [all_debt](#)
总负债
- double [line_of_credit](#)
两融授信额度
- double [guaranty](#)
两融保证金可用数
- double [reserved](#)
保留字段

5.17.1 详细描述

融资融券特有帐户数据

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.18 XTPCrdPositionExtralInfo结构体 参考

融资融券帐户持仓附加信息

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP_MARKET_TYPE](#) market
证券市场
- [char ticker](#) [[XTP_TICKER_LEN](#)]
证券代码
- [double mf_rs_avl_used](#)
购买货币基金使用的融券卖出所得资金占用
- [char reserve](#) [64]
预留空间

5.18.1 详细描述

融资融券帐户持仓附加信息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.19 XTPCreditDebtExtendNotice结构体 参考

用户展期请求的通知

```
#include <xoms_api_struct.h>
```

成员变量

- [uint64_t xtpid](#)
XTP系统订单ID, 无需用户填写, 在XTP系统中唯一
- [char debt_id](#) [[XTP_CREDIT_DEBT_ID_LEN](#)]
负债合约编号
- [XTP_DEBT_EXTEND_OPER_STATUS](#) oper_status
展期请求操作状态
- [uint64_t oper_time](#)
操作时间

5.19.1 详细描述

用户展期请求的通知

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.20 XTPCreditDebtExtendReq结构体 参考

用户展期请求

```
#include <xoms_api_struct.h>
```

成员变量

- `uint64_t xtpid`
xtpid
- `char debt_id [XTP_CREDIT_DEBT_ID_LEN]`
负债合约编号
- `uint32_t defer_days`
展期天数
- `char fund_account [XTP_ACCOUNT_NAME_LEN]`
资金账号
- `char password [XTP_ACCOUNT_PASSWORD_LEN]`
资金账号密码

5.20.1 详细描述

用户展期请求

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.21 XTPFundQueryReq结构体 参考

用户资金查询请求结构体

```
#include <xoms_api_fund_struct.h>
```

成员变量

- char [fund_account](#) [XTP_ACCOUNT_NAME_LEN]
资金账户代码
- char [password](#) [XTP_ACCOUNT_PASSWORD_LEN]
资金账户密码
- [XTP_FUND_QUERY_TYPE](#) [query_type](#)
查询类型
- [uint64_t](#) [unknown](#) [4]
预留字段，用户无需填写

5.21.1 详细描述

用户资金查询请求结构体

该结构体的文档由以下文件生成:

- [xoms_api_fund_struct.h](#)

5.22 XTPFundQueryRsp结构体 参考

用户资金查询响应结构体

```
#include <xoms_api_fund_struct.h>
```

成员变量

- double [amount](#)
金额
- [XTP_FUND_QUERY_TYPE](#) [query_type](#)
查询类型
- [uint64_t](#) [unknown](#) [4]
预留字段，用户无需填写

5.22.1 详细描述

用户资金查询响应结构体

该结构体的文档由以下文件生成:

- [xoms_api_fund_struct.h](#)

5.23 XTPFundTransferNotice结构体 参考

资金内转流水通知

```
#include <xoms_api_struct.h>
```

成员变量

- `uint64_t serial_id`
资金内转编号
- `XTP_FUND_TRANSFER_TYPE transfer_type`
内转类型
- `double amount`
金额
- `XTP_FUND_OPER_STATUS oper_status`
操作结果
- `uint64_t transfer_time`
操作时间

5.23.1 详细描述

资金内转流水通知

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.24 XTPFundTransferReq结构体 参考

用户资金请求

```
#include <xoms_api_fund_struct.h>
```

成员变量

- `uint64_t serial_id`
资金内转编号，无需用户填写，类似于 *xtp_id*
- `char fund_account [XTP_ACCOUNT_NAME_LEN]`
资金账户代码
- `char password [XTP_ACCOUNT_PASSWORD_LEN]`
资金账户密码
- `double amount`
金额
- `XTP_FUND_TRANSFER_TYPE transfer_type`
内转类型

5.24.1 详细描述

用户资金请求

该结构体的文档由以下文件生成:

- [xoms_api_fund_struct.h](#)

5.25 XTPMarketDataOptionExData结构体 参考

期权额外数据

```
#include <xquote_api_struct.h>
```

成员变量

- double [auction_price](#)
波段性中断参考价(SH)
- int64_t [auction_qty](#)
波段性中断集合竞价虚拟匹配量(SH)
- int64_t [last_enquiry_time](#)
最近询价时间(SH)

5.25.1 详细描述

期权额外数据

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.26 XTPMarketDataStockExData结构体 参考

股票、基金、债券等额外数据

```
#include <xquote_api_struct.h>
```

成员变量

- int64_t [total_bid_qty](#)
委托买入总量(SH,SZ)
- int64_t [total_ask_qty](#)
委托卖出总量(SH,SZ)
- double [ma_bid_price](#)
加权平均委买价格(SH,SZ)
- double [ma_ask_price](#)
加权平均委卖价格(SH,SZ)
- double [ma_bond_bid_price](#)
债券加权平均委买价格(SH)
- double [ma_bond_ask_price](#)
债券加权平均委卖价格(SH)
- double [yield_to_maturity](#)
债券到期收益率(SH)
- double [iopv](#)

- 基金实时参考净值(*SH,SZ*)
- `int32_t etf_buy_count`
ETF申购笔数(*SH*)
- `int32_t etf_sell_count`
ETF赎回笔数(*SH*)
- `double etf_buy_qty`
ETF申购数量(*SH*)
- `double etf_buy_money`
ETF申购金额(*SH*)
- `double etf_sell_qty`
ETF赎回数量(*SH*)
- `double etf_sell_money`
ETF赎回金额(*SH*)
- `double total_warrant_exec_qty`
权证执行的总数量(*SH*)
- `double warrant_lower_price`
权证跌停价格（元）(*SH*)
- `double warrant_upper_price`
权证涨停价格（元）(*SH*)
- `int32_t cancel_buy_count`
买入撤单笔数(*SH*)
- `int32_t cancel_sell_count`
卖出撤单笔数(*SH*)
- `double cancel_buy_qty`
买入撤单数量(*SH*)
- `double cancel_sell_qty`
卖出撤单数量(*SH*)
- `double cancel_buy_money`
买入撤单金额(*SH*)
- `double cancel_sell_money`
卖出撤单金额(*SH*)
- `int64_t total_buy_count`
买入总笔数(*SH*)
- `int64_t total_sell_count`
卖出总笔数(*SH*)
- `int32_t duration_after_buy`
买入委托成交最大等待时间(*SH*)
- `int32_t duration_after_sell`
卖出委托成交最大等待时间(*SH*)
- `int32_t num_bid_orders`
买方委托价位数(*SH*)
- `int32_t num_ask_orders`
卖方委托价位数(*SH*)
- `double pre_iopv`
基金*T-1*日净值(*SZ*)
- `int64_t r1`
预留
- `int64_t r2`
预留

5.26.1 详细描述

股票、基金、债券等额外数据

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.27 XTPMarketDataStruct结构体 参考

行情

```
#include <xquote_api_struct.h>
```

成员变量

- [XTP_EXCHANGE_TYPE exchange_id](#)
交易所代码
- [char ticker \[XTP_TICKER_LEN\]](#)
合约代码（不包含交易所信息），不带空格，以'\0'结尾
- [double last_price](#)
最新价
- [double pre_close_price](#)
昨收盘
- [double open_price](#)
今开盘
- [double high_price](#)
最高价
- [double low_price](#)
最低价
- [double close_price](#)
今收盘
- [int64_t pre_total_long_positon](#)
昨日持仓量(张)(目前未填写)
- [int64_t total_long_positon](#)
持仓量(张)
- [double pre_settl_price](#)
昨日结算价
- [double settl_price](#)
今日结算价
- [double upper_limit_price](#)
涨停价
- [double lower_limit_price](#)
跌停价
- [double pre_delta](#)
预留
- [double curr_delta](#)
预留

- `int64_t data_time`
时间类，格式为 `YYYYMMDDHHMMSSsss`
- `int64_t qty`
数量，为总成交量（单位股，与交易所一致）
- `double turnover`
成交金额，为总成交金额（单位元，与交易所一致）
- `double avg_price`
当日均价= $(turnover/qty)$
- `double bid [10]`
十档申买价
- `double ask [10]`
十档申卖价
- `int64_t bid_qty [10]`
十档申买量
- `int64_t ask_qty [10]`
十档申卖量
- `int64_t trades_count`
成交笔数
- `char ticker_status [8]`
当前交易状态说明，参阅《*XTP API常见问题.doc*》文档
- ```
union {
 XTPMarketDataStockExData stk
 XTPMarketDataOptionExData opt
};
```

数据
- `XTP_MARKETDATA_TYPE data_type`  
决定了 `union` 是哪种数据类型
- `int32_t r4`  
预留

### 5.27.1 详细描述

行情

该结构体的文档由以下文件生成:

- `xquote_api_struct.h`

## 5.28 XTPOptCombLegInfo结构体 参考

组合策略腿合约信息结构体

```
#include <xoms_api_struct.h>
```



## 成员变量

- `char leg_security_id [XTP_TICKER_LEN]`  
成分合约代码
- `XTP_OPT_CALL_OR_PUT_TYPE leg_cntr_type`  
合约类型，认沽或认购。
- `XTP_POSITION_DIRECTION_TYPE leg_side`  
持仓方向，权利方或义务方。
- `XTP_OPT_COVERED_OR_UNCOVERED leg_covered`  
备兑标签
- `int32_t leg_qty`  
成分合约数量（张）

## 5.28.1 详细描述

组合策略腿合约信息结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.29 XTPOptCombOrderInfo结构体 参考

期权组合策略报单响应结构体

```
#include <xoms_api_struct.h>
```

## 成员变量

- `uint64_t order_xtp_id`  
*XTP系统订单ID，在XTP系统中唯一*
- `uint32_t order_client_id`  
报单引用，用户自定义
- `uint32_t order_cancel_client_id`  
报单操作引用，用户自定义（暂未使用）
- `uint64_t order_cancel_xtp_id`  
撤单在XTP系统中的id，在XTP系统中唯一
- `XTP_MARKET_TYPE market`
- `int64_t quantity`  
数量，此订单的报单数量
- `XTP_SIDE_TYPE side`  
组合方向
- `XTP_BUSINESS_TYPE business_type`  
业务类型
- `int64_t qty_traded`  
今成交数量，为此订单累计成交数量
- `int64_t qty_left`  
剩余数量，当撤单成功时，表示撤单数量

- `int64_t insert_time`  
委托时间, 格式为 `YYYYMMDDHHMMSSsss`
- `int64_t update_time`  
最后修改时间, 格式为 `YYYYMMDDHHMMSSsss`
- `int64_t cancel_time`  
撤销时间, 格式为 `YYYYMMDDHHMMSSsss`
- `double trade_amount`  
成交金额, 组合拆分涉及的保证金(保留字段)
- `char order_local_id [XTP_LOCAL_ORDER_LEN]`  
本地报单编号 *OMS*生成的单号, 不等同于 *order\_xtp\_id*, 为服务器传到报盘的单号
- `XTP_ORDER_STATUS_TYPE order_status`  
报单状态, 订单响应中没有部分成交状态的推送, 在查询订单结果中, 会有部分成交状态
- `XTP_ORDER_SUBMIT_STATUS_TYPE order_submit_status`  
报单提交状态, 用户可用此字段来区分撤单和报单
- `TXTPOrderTypeType order_type`  
报单类型
- `XTPOptCombPlugin opt_comb_info`  
期权组合策略信息

### 5.29.1 详细描述

期权组合策略报单响应结构体

### 5.29.2 结构体成员变量说明

#### 5.29.2.1 market

`XTP_MARKET_TYPE` market

证券代码 `char ticker[XTP_TICKER_LEN]`; 交易市场

该结构体的文档由以下文件生成:

- `xoms_api_struct.h`

## 5.30 XTPOptCombOrderInsertInfo结构体 参考

期权组合策略新订单请求

```
#include <xoms_api_struct.h>
```

## 成员变量

- `uint64_t order_xtp_id`  
XTP系统订单ID, 无需用户填写, 在XTP系统中唯一
- `uint32_t order_client_id`  
报单引用, 由客户自定义
- `XTP_MARKET_TYPE market`  
交易市场
- `int64_t quantity`  
数量(单位为份)
- `XTP_SIDE_TYPE side`  
组合方向
- `XTP_BUSINESS_TYPE business_type`  
业务类型
- `XTPOptCombPlugin opt_comb_info`  
期权组合策略信息

## 5.30.1 详细描述

期权组合策略新订单请求

该结构体的文档由以下文件生成:

- `xoms_api_struct.h`

## 5.31 XTPOptCombPlugin结构体 参考

期权组合策略报单附加信息结构体

```
#include <xoms_api_struct.h>
```

## 成员变量

- `char strategy_id [XTP_STRATEGY_ID_LEN]`  
组合策略代码, 比如CNSJC认购牛市价差策略等。合并行权时, 此字段可为空
- `char comb_num [XTP_SECONDARY_ORDER_ID_LEN]`  
组合编码, 组合申报时, 该字段为空; 拆分申报时, 填写拟拆分组合的组合编码。
- `int32_t num_legs`  
成分合约数
- `XTPOptCombLegInfo leg_detail [XTP_STRATEGEG_LEG_NUM]`  
成分合约数组, 最多四条腿。

## 5.31.1 详细描述

期权组合策略报单附加信息结构体

该结构体的文档由以下文件生成:

- `xoms_api_struct.h`

## 5.32 XTPOptCombTradeReport结构体 参考

期权组合策略报单成交结构体

```
#include <xoms_api_struct.h>
```

成员变量

- `uint64_t order_xtp_id`  
XTP系统订单ID, 此成交回报相关的订单ID, 在XTP系统中唯一
- `uint32_t order_client_id`  
报单引用
- `XTP_MARKET_TYPE market`  
交易市场
- `uint64_t local_order_id`  
订单号, 引入XTPID后, 该字段实际和order\_xtp\_id重复。接口中暂时保留。
- `char exec_id [XTP_EXEC_ID_LEN]`  
成交编号, 深交所唯一, 上交所每笔交易唯一, 当发现2笔成交回报拥有相同的exec\_id, 则可以认为此笔交易自成交
- `int64_t quantity`  
数量, 此次成交的数量, 不是累计数量
- `int64_t trade_time`  
成交时间, 格式为YYYYMMDDHHMMSSsss
- `double trade_amount`  
成交金额, 组合拆分涉及的保证金
- `uint64_t report_index`  
成交序号 - 回报记录号, 每个交易所唯一, report\_index+market字段可以组成唯一标识表示成交回报
- `char order_exch_id [XTP_ORDER_EXCH_LEN]`  
报单编号 - 交易所单号(保留字段)
- `TXTPTradeTypeType trade_type`  
成交类型 - 成交回报中的执行类型
- `XTP_SIDE_TYPE side`  
组合方向
- `XTP_BUSINESS_TYPE business_type`  
业务类型
- `char branch_pbu [XTP_BRANCH_PBU_LEN]`  
交易所交易员代码
- `XTPOptCombPlugin opt_comb_info`  
期权组合策略信息

### 5.32.1 详细描述

期权组合策略报单成交结构体

该结构体的文档由以下文件生成:

- `xoms_api_struct.h`

## 5.33 XTPOrderCancelInfo结构体 参考

撤单失败响应消息

```
#include <xoms_api_struct.h>
```

成员变量

- `uint64_t order_cancel_xtp_id`  
撤单XTPID
- `uint64_t order_xtp_id`  
原始订单XTPID

### 5.33.1 详细描述

撤单失败响应消息

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.34 XTPOrderInfo结构体 参考

报单响应结构体

```
#include <xoms_api_struct.h>
```

成员变量

- `uint64_t order_xtp_id`  
XTP系统订单ID, 在XTP系统中唯一
- `uint32_t order_client_id`  
报单引用, 用户自定义
- `uint32_t order_cancel_client_id`  
报单操作引用, 用户自定义 (暂未使用)
- `uint64_t order_cancel_xtp_id`  
撤单在XTP系统中的id, 在XTP系统中唯一
- `char ticker[XTP_TICKER_LEN]`  
合约代码
- `XTP_MARKET_TYPE market`  
交易市场
- `double price`  
价格
- `int64_t quantity`  
数量, 此订单的报单数量
- `XTP_PRICE_TYPE price_type`

- 报单价格条件
- - union {
    - uint32\_t [u32](#)
      - 32位字段，用来兼容老版本`api`，用户无需关心
    - struct {
      - [XTP\\_SIDE\\_TYPE](#) `side`
        - 买卖方向
      - [XTP\\_POSITION\\_EFFECT\\_TYPE](#) `position_effect`
        - 开平标志，期权用户关注字段，其余用户填0即可
      - uint8\_t [reserved1](#)
        - 预留字段1
      - uint8\_t [reserved2](#)
        - 预留字段2
- };
- [XTP\\_BUSINESS\\_TYPE](#) `business_type`
  - 业务类型
- [int64\\_t](#) `qty_traded`
  - 今成交数量，为此订单累计成交数量
- [int64\\_t](#) `qty_left`
  - 剩余数量，当撤单成功时，表示撤单数量
- [int64\\_t](#) `insert_time`
  - 委托时间，格式为YYYYMMDDHHMMSSsss
- [int64\\_t](#) `update_time`
  - 最后修改时间，格式为YYYYMMDDHHMMSSsss
- [int64\\_t](#) `cancel_time`
  - 撤销时间，格式为YYYYMMDDHHMMSSsss
- [double](#) `trade_amount`
  - 成交金额，为此订单的成交总金额
- [char](#) `order_local_id` [[XTP\\_LOCAL\\_ORDER\\_LEN](#)]
  - 本地报单编号 `OMS`生成的单号，不等同于`order_xtp_id`，为服务器传到报盘的单号
- [XTP\\_ORDER\\_STATUS\\_TYPE](#) `order_status`
  - 报单状态，订单响应中没有部分成交状态的推送，在查询订单结果中，会有部分成交状态
- [XTP\\_ORDER\\_SUBMIT\\_STATUS\\_TYPE](#) `order_submit_status`
  - 报单提交状态，`OMS`内部使用，用户可用此字段来区分撤单和报单
- [XTPOrderTypeType](#) `order_type`
  - 报单类型

### 5.34.1 详细描述

报单响应结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.35 XTPOrderInsertInfo结构体 参考

新订单请求

```
#include <xoms_api_struct.h>
```

## 成员变量

- uint64\_t [order\\_xtp\\_id](#)  
XTP系统订单ID, 无需用户填写, 在XTP系统中唯一
- uint32\_t [order\\_client\\_id](#)  
报单引用, 由客户自定义
- char [ticker](#) [XTP\_TICKER\_LEN]  
合约代码 客户端请求不带空格, 以'\0'结尾
- [XTP\\_MARKET\\_TYPE](#) [market](#)  
交易市场
- double [price](#)  
价格
- double [stop\\_price](#)  
止损价 (保留字段)
- int64\_t [quantity](#)  
数量(股票单位为股, 逆回购单位为张)
- [XTP\\_PRICE\\_TYPE](#) [price\\_type](#)  
报单价格
- union {  
  uint32\_t [u32](#)  
  32位字段, 用来兼容老版本api, 用户无需关心  
  struct {  
    [XTP\\_SIDE\\_TYPE](#) [side](#)  
    买卖方向  
    [XTP\\_POSITION\\_EFFECT\\_TYPE](#) [position\\_effect](#)  
    开平标志  
    uint8\_t [reserved1](#)  
    预留字段1  
    uint8\_t [reserved2](#)  
    预留字段2  
  }  
};
- [XTP\\_BUSINESS\\_TYPE](#) [business\\_type](#)  
业务类型

## 5.35.1 详细描述

## 新订单请求

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.36 XTPQueryAssetRsp结构体 参考

## 账户资金查询响应结构体

```
#include <xoms_api_struct.h>
```

## 成员变量

- double [total\\_asset](#)  
总资产（现货账户/期权账户参考公式：总资产 = 可用资金 + 证券资产（目前为0）+ 预扣的资金），（信用账户参考公式：总资产 = 可用资金 + 融券卖出所得资金余额 + 证券资产 + 预扣的资金）
- double [buying\\_power](#)  
可用资金
- double [security\\_asset](#)  
证券资产（保留字段，目前为0）
- double [fund\\_buy\\_amount](#)  
累计买入成交证券占用资金（仅限现货账户/期权账户，信用账户暂不可用）
- double [fund\\_buy\\_fee](#)  
累计买入成交交易费用（仅限现货账户/期权账户，信用账户暂不可用）
- double [fund\\_sell\\_amount](#)  
累计卖出成交证券所得资金（仅限现货账户/期权账户，信用账户暂不可用）
- double [fund\\_sell\\_fee](#)  
累计卖出成交交易费用（仅限现货账户/期权账户，信用账户暂不可用）
- double [withholding\\_amount](#)  
XTP系统预扣的资金（包括买卖股票时预扣的交易资金+预扣手续费）
- [XTP\\_ACCOUNT\\_TYPE](#) [account\\_type](#)  
账户类型
- double [frozen\\_margin](#)  
冻结的保证金（仅限期权账户）
- double [frozen\\_exec\\_cash](#)  
行权冻结资金（仅限期权账户）
- double [frozen\\_exec\\_fee](#)  
行权费用（仅限期权账户）
- double [pay\\_later](#)  
垫付资金（仅限期权账户）
- double [preadva\\_pay](#)  
预垫付资金（仅限期权账户）
- double [orig\\_banlance](#)  
昨日余额（仅限期权账户）
- double [banlance](#)  
当前余额（仅限期权账户）
- double [deposit\\_withdraw](#)  
当天出入金（仅限期权账户）
- double [trade\\_netting](#)  
当日交易资金轧差（仅限期权账户）
- double [captial\\_asset](#)  
资金资产（仅限期权账户）
- double [force\\_freeze\\_amount](#)  
强锁资金（仅限期权账户）
- double [preferred\\_amount](#)  
可取资金（仅限期权账户）
- double [repay\\_stock\\_aval\\_banlance](#)  
融券卖出所得资金余额（仅限信用账户，只能用于买券还券）
- double [fund\\_order\\_data\\_charges](#)  
累计订单流量费
- double [fund\\_cancel\\_data\\_charges](#)  
累计撤单流量费



- double [exchange\\_cur\\_risk\\_degree](#)  
交易所实时风险度（仅限期权账户,后续服务器版本支持，目前为0）
- double [company\\_cur\\_risk\\_degree](#)  
公司实时风险度（仅限期权账户,后续服务器版本支持，目前为0）
- uint64\_t [unknown](#) [43 - 12 - 1 - 2 - 2]  
(保留字段)

### 5.36.1 详细描述

账户资金查询响应结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.37 XTPQueryCombineStrategyInfoRsp结构体 参考

查询期权组合策略信息的响应

```
#include <xoms_api_struct.h>
```

成员变量

- char [strategy\\_id](#) [XTP\_STRATEGY\_ID\_LEN]  
组合策略代码, *CNSJC*、*PXSJC*、*PNSJC*、*CXSJC*、*KS*、*KKS*
- char [strategy\\_name](#) [XTP\_STRATEGY\_NAME\_LEN]  
组合策略名称, 认购牛市价差策略、认沽熊市价差策略、认沽牛市价差策略、认购熊市价差策略、跨式空头、宽跨式空头
- [XTP\\_MARKET\\_TYPE](#) [market](#)  
交易市场
- int32\_t [leg\\_num](#)  
成分合约个数, 1-4个, 即下面数组的实际大小
- [XTPCombLegStrategy](#) [leg\\_strategy](#) [XTP\_STRATEGIE\_LEG\_NUM]  
成分合约信息, 最多四条腿
- [XTP\\_EXPIRE\\_DATE\\_TYPE](#) [expire\\_date\\_type](#)  
到期日要求。枚举值为: 同到期日, 不同到期日, 无到期日要求
- [XTP\\_UNDERLYING\\_TYPE](#) [underlying\\_type](#)  
标的要求。枚举值为: 相同标的, 不同标的, 无标的要求
- [XTP\\_AUTO\\_SPLIT\\_TYPE](#) [auto\\_sep\\_type](#)  
自动解除类型。枚举值为: -1: 不适用; 0: 到期日自动解除; 1: *E-1*日自动解除, 依次类推
- uint64\_t [reserved](#) [10]  
预留的字段

### 5.37.1 详细描述

查询期权组合策略信息的响应

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.38 XTPQueryETFBaseReq结构体 参考

```
#include <xoms_api_struct.h>
```

### 成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
*ETF*买卖代码

### 5.38.1 详细描述

查询股票ETF合约基本情况-请求结构体, 请求参数为多条件参数:1,不填则返回所有市场的ETF合约信息。  
2,只填写market,返回该交易市场下结果 3,填写market及ticker参数,只返回该etf信息。

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.39 XTPQueryETFBaseRsp结构体 参考

查询股票ETF合约基本情况-响应结构体

```
#include <xoms_api_struct.h>
```

### 成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- [char etf \[XTP\\_TICKER\\_LEN\]](#)  
*etf*代码,买卖,申赎统一使用该代码
- [char subscribe\\_redemption\\_ticker \[XTP\\_TICKER\\_LEN\]](#)  
*etf*申购赎回代码
- [int32\\_t unit](#)  
最小申购赎回单位对应的*ETF*份数,例如上证"50*ETF*"就是900000
- [int32\\_t subscribe\\_status](#)  
是否允许申购,1-允许,0-禁止
- [int32\\_t redemption\\_status](#)  
是否允许赎回,1-允许,0-禁止
- [double max\\_cash\\_ratio](#)  
最大现金替代比例,小于1的数值 *TODO* 是否采用*double*
- [double estimate\\_amount](#)  
*T*日预估金额差额
- [double cash\\_component](#)  
*T-X*日现金差额
- [double net\\_value](#)  
基金单位净值
- [double total\\_amount](#)  
最小申赎单位净值总金额= $net\_value * unit$

### 5.39.1 详细描述

查询股票ETF合约基本情况-响应结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.40 XTPQueryETFComponentReq结构体 参考

查询股票ETF合约成分股信息-请求结构体,请求参数为:交易市场+ETF买卖代码

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
ETF买卖代码

### 5.40.1 详细描述

查询股票ETF合约成分股信息-请求结构体,请求参数为:交易市场+ETF买卖代码

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.41 XTPQueryETFComponentRsp结构体 参考

查询股票ETF成分股信息-响应结构体

```
#include <xoms_api_struct.h>
```

## 成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
ETF代码
- [char component\\_ticker \[XTP\\_TICKER\\_LEN\]](#)  
成份股代码
- [char component\\_name \[XTP\\_TICKER\\_NAME\\_LEN\]](#)  
成份股名称
- [int64\\_t quantity](#)  
成份股数量
- [XTP\\_MARKET\\_TYPE component\\_market](#)  
成份股交易市场
- [ETF\\_REPLACE\\_TYPE replace\\_type](#)  
成份股替代标识
- [double premium\\_ratio](#)  
溢价比例
- [double amount](#)  
成分股替代标识为必须现金替代时候的总金额
- [double creation\\_premium\\_ratio](#)  
申购溢价比例
- [double redemption\\_discount\\_ratio](#)  
赎回溢价比例
- [double creation\\_amount](#)  
申购时，成分股替代标识为必须现金替代时候的总金额
- [double redemption\\_amount](#)  
赎回时，成分股替代标识为必须现金替代时候的总金额

### 5.41.1 详细描述

查询股票ETF成分股信息-响应结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.42 XTPQueryETFComponentRspV1结构体 参考

查询股票ETF成分股信息-响应结构体，旧版本。

```
#include <xoms_api_struct.h>
```

## 成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
ETF代码
- [char component\\_ticker \[XTP\\_TICKER\\_LEN\]](#)  
成份股代码
- [char component\\_name \[XTP\\_TICKER\\_NAME\\_LEN\]](#)  
成份股名称
- [int64\\_t quantity](#)  
成份股数量
- [XTP\\_MARKET\\_TYPE component\\_market](#)  
成份股交易市场
- [ETF\\_REPLACE\\_TYPE replace\\_type](#)  
成份股替代标识
- [double premium\\_ratio](#)  
溢价比例
- [double amount](#)  
成份股替代标识为必须现金替代时候的总金额

## 5.42.1 详细描述

查询股票ETF成分股信息—响应结构体，旧版本。

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.43 XTPQueryFundTransferLogReq结构体 参考

资金内转流水查询请求与响应

```
#include <xoms_api_struct.h>
```

## 成员变量

- [uint64\\_t serial\\_id](#)  
资金内转编号

## 5.43.1 详细描述

资金内转流水查询请求与响应

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.44 XTPQueryIPOQuotaRsp结构体 参考

查询用户申购额度-包含创业板额度

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- [int32\\_t quantity](#)  
可申购额度
- [int32\\_t tech\\_quantity](#)  
上海科创板额度
- [int32\\_t unused](#)  
保留

### 5.44.1 详细描述

查询用户申购额度-包含创业板额度

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.45 XTPQueryIPOQuotaRspV1结构体 参考

查询用户申购额度-旧版

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- [int32\\_t quantity](#)  
可申购额度

### 5.45.1 详细描述

查询用户申购额度-旧版

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.46 XTPQueryIPOTickerRsp结构体 参考

查询当日可申购新股信息

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
申购代码
- [char ticker\\_name \[XTP\\_TICKER\\_NAME\\_LEN\]](#)  
申购股票名称
- [XTP\\_TICKER\\_TYPE ticker\\_type](#)  
证券类别
- [double price](#)  
申购价格
- [int32\\_t unit](#)  
申购单元
- [int32\\_t qty\\_upper\\_limit](#)  
最大允许申购数量

### 5.46.1 详细描述

查询当日可申购新股信息

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.47 XTPQueryOptCombExecPosReq结构体 参考

查询期权行权合并头寸请求结构体

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
市场
- [char cntrt\\_code\\_1 \[XTP\\_TICKER\\_LEN\]](#)  
成分合约1代码
- [char cntrt\\_code\\_2 \[XTP\\_TICKER\\_LEN\]](#)  
成分合约2代码

### 5.47.1 详细描述

查询期权行权合并头寸请求结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.48 XTPQueryOptCombExecPosRsp结构体 参考

查询期权行权合并头寸的响应

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
市场
- [char cntrt\\_code\\_1 \[XTP\\_TICKER\\_LEN\]](#)  
成分合约1代码
- [char cntrt\\_name\\_1 \[XTP\\_TICKER\\_NAME\\_LEN\]](#)  
成分合约1名称
- [XTP\\_POSITION\\_DIRECTION\\_TYPE position\\_side\\_1](#)  
成分合约1持仓方向
- [XTP\\_OPT\\_CALL\\_OR\\_PUT\\_TYPE call\\_or\\_put\\_1](#)  
成分合约1类型
- [int64\\_t avl\\_qty\\_1](#)  
成分合约1可用持仓数量
- [int64\\_t orig\\_own\\_qty\\_1](#)  
成分合约1昨日持仓数量
- [int64\\_t own\\_qty\\_1](#)  
成分合约1当前持仓数量
- [char cntrt\\_code\\_2 \[XTP\\_TICKER\\_LEN\]](#)  
成分合约2代码
- [char cntrt\\_name\\_2 \[XTP\\_TICKER\\_NAME\\_LEN\]](#)  
成分合约2名称
- [XTP\\_POSITION\\_DIRECTION\\_TYPE position\\_side\\_2](#)  
成分合约2持仓方向
- [XTP\\_OPT\\_CALL\\_OR\\_PUT\\_TYPE call\\_or\\_put\\_2](#)  
成分合约2类型
- [int64\\_t avl\\_qty\\_2](#)  
成分合约2可用持仓数量
- [int64\\_t orig\\_own\\_qty\\_2](#)  
成分合约2昨日持仓数量
- [int64\\_t own\\_qty\\_2](#)  
成分合约2当前持仓数量
- [int64\\_t net\\_qty](#)  
权利仓净头寸



- `int64_t order_qty`  
行权合并委托数量，不含已拒单已撤单。
- `int64_t confirm_qty`  
行权合并已确认数量
- `int64_t avl_qty`  
可行权合并数量
- `uint64_t reserved` [49]  
保留字段

### 5.48.1 详细描述

查询期权行权合并头寸的响应

该结构体的文档由以下文件生成:

- `xoms_api_struct.h`

## 5.49 XTPQueryOptCombOrderByPageReq结构体 参考

查询期权组合策略订单请求-分页查询

```
#include <xoms_api_struct.h>
```

成员变量

- `int64_t req_count`  
需要查询的订单条数
- `int64_t reference`  
上一次收到的查询订单结果中带回来的索引，如果是从头查询，请置0
- `int64_t reserved`  
保留字段

### 5.49.1 详细描述

查询期权组合策略订单请求-分页查询

该结构体的文档由以下文件生成:

- `xoms_api_struct.h`

## 5.50 XTPQueryOptCombOrderReq结构体 参考

期权组合策略报单查询 ////////////////////////////////////////////////// 期权组合策略报单查询请求-条件查询

```
#include <xoms_api_struct.h>
```

## 成员变量

- `char comb_num [XTP_SECONDARY_ORDER_ID_LEN]`  
组合编码（流水号），可以为空，如果为空，则默认查询时间段内的所有成交回报
- `int64_t begin_time`  
格式为YYYYMMDDHHMMSSsss，为0则默认当前交易日0点
- `int64_t end_time`  
格式为YYYYMMDDHHMMSSsss，为0则默认当前时间

### 5.50.1 详细描述

期权组合策略报单查询 ////////////////////////////////////// 期权组合策略报单查询请求-条件查询

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.51 XTPQueryOptCombPositionReq结构体 参考

查询期权组合策略持仓情况请求结构体

```
#include <xoms_api_struct.h>
```

## 成员变量

- `char comb_num [XTP_SECONDARY_ORDER_ID_LEN]`  
组合编码
- `XTP_MARKET_TYPE market`  
交易市场

### 5.51.1 详细描述

查询期权组合策略持仓情况请求结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.52 XTPQueryOptCombPositionRsp结构体 参考

查询期权组合策略持仓信息的响应

```
#include <xoms_api_struct.h>
```

## 成员变量

- char [strategy\\_id](#) [XTP\_STRATEGY\_ID\_LEN]  
组合策略代码
- char [strategy\\_name](#) [XTP\_STRATEGY\_NAME\_LEN]  
组合策略名称
- [XTP\\_MARKET\\_TYPE](#) market  
交易市场
- int64\_t [total\\_qty](#)  
总持仓
- int64\_t [available\\_qty](#)  
可拆分持仓
- int64\_t [yesterday\\_position](#)  
昨日持仓
- [XTPOptCombPlugin](#) opt\_comb\_info  
期权组合策略信息
- uint64\_t [reserved](#) [50]  
保留字段

## 5.52.1 详细描述

查询期权组合策略持仓信息的响应

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.53 XTPQueryOptCombReportByExecIdReq结构体 参考

期权组合策略成交回报查询 ////////////////////////////////////////////////////////////////// 查询期权组合策略成交报告请求-根据执行编号查询（保留字段）

```
#include <xoms_api_struct.h>
```

## 成员变量

- uint64\_t [order\\_xtp\\_id](#)  
XTP订单系统ID
- char [exec\\_id](#) [XTP\_EXEC\_ID\_LEN]  
成交执行编号

## 5.53.1 详细描述

期权组合策略成交回报查询 ////////////////////////////////////////////////////////////////// 查询期权组合策略成交报告请求-根据执行编号查询（保留字段）

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.54 XTPQueryOptCombTraderByPageReq结构体 参考

查询期权组合策略成交回报请求-分页查询

```
#include <xoms_api_struct.h>
```

成员变量

- `int64_t req_count`  
需要查询的成交回报条数
- `int64_t reference`  
上一次收到的查询成交回报结果中带回来的索引，如果是从头查询，请置0
- `int64_t reserved`  
保留字段

### 5.54.1 详细描述

查询期权组合策略成交回报请求-分页查询

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.55 XTPQueryOptCombTraderReq结构体 参考

查询期权组合策略成交回报请求-查询条件

```
#include <xoms_api_struct.h>
```

成员变量

- `char comb_num [XTP_SECONDARY_ORDER_ID_LEN]`  
组合编码（流水号），可以为空，如果为空，则默认查询时间段内的所有成交回报
- `int64_t begin_time`  
开始时间，格式为YYYYMMDDHHMMSSsss，为0则默认当前交易日0点
- `int64_t end_time`  
结束时间，格式为YYYYMMDDHHMMSSsss，为0则默认当前时间

### 5.55.1 详细描述

查询期权组合策略成交回报请求-查询条件

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.56 XTPQueryOptExecInfoRsp结构体 参考

查询期权合约行权信息的响应

```
#include <xoms_api_struct.h>
```

成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
市场
- [char cntrt\\_code \[XTP\\_TICKER\\_LEN\]](#)  
合约代码
- [int64\\_t own\\_qty\\_long](#)  
权利仓数量
- [int64\\_t own\\_qty\\_short](#)  
义务仓数量
- [int64\\_t own\\_qty\\_short\\_cover](#)  
备兑义务仓数量
- [int64\\_t net\\_qty](#)  
净头寸
- [int64\\_t combed\\_qty\\_long](#)  
权利仓已组合数量
- [int64\\_t combed\\_qty\\_short](#)  
义务仓已组合数量
- [int64\\_t combed\\_qty\\_short\\_cover](#)  
备兑义务仓已组合数量
- [int64\\_t total\\_execute\\_gene\\_order\\_qty](#)  
累计普通行权委托数量
- [int64\\_t total\\_execute\\_gene\\_confirm\\_qty](#)  
累计普通行权确认数量
- [int64\\_t total\\_execute\\_comb\\_order\\_qty](#)  
累计行权合并委托数量
- [int64\\_t total\\_execute\\_comb\\_confirm\\_qty](#)  
累计行权合并确认数量
- [uint64\\_t reserved \[50\]](#)  
保留字段

### 5.56.1 详细描述

查询期权合约行权信息的响应

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.57 XTPQueryOptionAuctionInfoReq结构体 参考

查询期权竞价交易业务参考信息-请求结构体,请求参数为:交易市场+8位期权代码

```
#include <xoms_api_struct.h>
```

### 成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
8位期权合约代码

### 5.57.1 详细描述

查询期权竞价交易业务参考信息-请求结构体,请求参数为:交易市场+8位期权代码

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.58 XTPQueryOptionAuctionInfoRsp结构体 参考

查询期权竞价交易业务参考信息

```
#include <xoms_api_struct.h>
```

### 成员变量

- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
合约编码, 报单 *ticker* 采用本字段
- [XTP\\_MARKET\\_TYPE security\\_id\\_source](#)  
证券代码源
- [char symbol \[XTP\\_TICKER\\_NAME\\_LEN\]](#)  
合约简称
- [char contract\\_id \[XTP\\_TICKER\\_NAME\\_LEN\]](#)  
合约交易代码
- [char underlying\\_security\\_id \[XTP\\_TICKER\\_LEN\]](#)  
基础证券代码
- [XTP\\_MARKET\\_TYPE underlying\\_security\\_id\\_source](#)  
基础证券代码源
- [uint32\\_t list\\_date](#)  
上市日期, 格式为 YYYYMMDD
- [uint32\\_t last\\_trade\\_date](#)  
最后交易日, 格式为 YYYYMMDD
- [XTP\\_TICKER\\_TYPE ticker\\_type](#)

- 证券类别
- `int32_t day_trading`  
是否支持当日回转交易, 1-允许, 0-不允许
- `XTP_OPT_CALL_OR_PUT_TYPE call_or_put`  
认购或认沽
- `uint32_t delivery_day`  
行权交割日, 格式为YYYYMMDD
- `uint32_t delivery_month`  
交割月份, 格式为YYYYMM
- `XTP_OPT_EXERCISE_TYPE_TYPE exercise_type`  
行权方式
- `uint32_t exercise_begin_date`  
行权起始日期, 格式为YYYYMMDD
- `uint32_t exercise_end_date`  
行权结束日期, 格式为YYYYMMDD
- `double exercise_price`  
行权价格
- `int64_t qty_unit`  
数量单位, 对于某一证券申报的委托, 其委托数量字段必须为该证券数量单位的整数倍
- `int64_t contract_unit`  
合约单位
- `int64_t contract_position`  
合约持仓量
- `double prev_close_price`  
合约前收盘价
- `double prev_clearing_price`  
合约前结算价
- `int64_t lmt_buy_max_qty`  
限价买最大量
- `int64_t lmt_buy_min_qty`  
限价买最小量
- `int64_t lmt_sell_max_qty`  
限价卖最大量
- `int64_t lmt_sell_min_qty`  
限价卖最小量
- `int64_t mkt_buy_max_qty`  
市价买最大量
- `int64_t mkt_buy_min_qty`  
市价买最小量
- `int64_t mkt_sell_max_qty`  
市价卖最大量
- `int64_t mkt_sell_min_qty`  
市价卖最小量
- `double price_tick`  
最小报价单位
- `double upper_limit_price`  
涨停价
- `double lower_limit_price`  
跌停价
- `double sell_margin`  
今卖开每张保证金

- double [margin\\_ratio\\_param1](#)  
交易所保证金比例计算参数一
- double [margin\\_ratio\\_param2](#)  
交易所保证金比例计算参数二
- uint64\_t [unknown](#) [20]  
(保留字段)

### 5.58.1 详细描述

查询期权竞价交易业务参考信息

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.59 XTPQueryOrderByPageReq结构体 参考

查询订单请求-分页查询

```
#include <xoms_api_struct.h>
```

成员变量

- int64\_t [req\\_count](#)  
需要查询的订单条数
- int64\_t [reference](#)  
上一次收到的查询订单结果中带回来的索引，如果是从头查询，请置0
- int64\_t [reserved](#)  
保留字段

### 5.59.1 详细描述

查询订单请求-分页查询

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.60 XTPQueryOrderReq结构体 参考

报单查询 ////////////////////////////////// 报单查询请求-条件查询

```
#include <xoms_api_struct.h>
```



## 成员变量

- char [ticker](#) [XTP\_TICKER\_LEN]  
证券代码，可以为空，如果为空，则默认查询时间段内的所有成交回报
- int64\_t [begin\\_time](#)  
格式为YYYYMMDDHHMMSSsss，为0则默认当前交易日0点
- int64\_t [end\\_time](#)  
格式为YYYYMMDDHHMMSSsss，为0则默认当前时间

## 5.60.1 详细描述

报单查询 ////////////////////////////////////////////////////////////////// 报单查询请求-条件查询

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.61 XTPQueryReportByExecIdReq结构体 参考

成交回报查询 ////////////////////////////////////////////////////////////////// 查询成交报告请求-根据执行编号查询（保留字段）

```
#include <xoms_api_struct.h>
```

## 成员变量

- uint64\_t [order\\_xtp\\_id](#)  
XTP订单系统ID
- char [exec\\_id](#) [XTP\_EXEC\_ID\_LEN]  
成交执行编号

## 5.61.1 详细描述

成交回报查询 ////////////////////////////////////////////////////////////////// 查询成交报告请求-根据执行编号查询（保留字段）

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.62 XTPQueryStkPositionReq结构体 参考

查询股票持仓情况请求结构体

```
#include <xoms_api_struct.h>
```

## 成员变量

- char `ticker` [`XTP_TICKER_LEN`]  
证券代码
- `XTP_MARKET_TYPE` `market`  
交易市场

### 5.62.1 详细描述

查询股票持仓情况请求结构体

该结构体的文档由以下文件生成:

- `xoms_api_struct.h`

## 5.63 XTPQueryStkPositionRsp结构体 参考

查询股票持仓情况

```
#include <xoms_api_struct.h>
```

## 成员变量

- char `ticker` [`XTP_TICKER_LEN`]  
证券代码
- char `ticker_name` [`XTP_TICKER_NAME_LEN`]  
证券名称
- `XTP_MARKET_TYPE` `market`  
交易市场
- int64\_t `total_qty`  
总持仓
- int64\_t `sellable_qty`  
可卖持仓
- double `avg_price`  
持仓成本
- double `unrealized_pnl`  
浮动盈亏 (保留字段)
- int64\_t `yesterday_position`  
昨日持仓
- int64\_t `purchase_redeemable_qty`  
今日申购赎回数量 (申购和赎回数量不可能同时存在, 因此可以共用一个字段)
- `XTP_POSITION_DIRECTION_TYPE` `position_direction`  
持仓方向
- `XTP_POSITION_SECURITY_TYPE` `position_security_type`  
持仓类型(此字段所有账户都可能用到, 可以用来区分股份是否为配售)
- int64\_t `executable_option`  
可行权合约
- int64\_t `lockable_position`

- 可锁定标的
- `int64_t executable_underlying`  
可行权标的
- `int64_t locked_position`  
已锁定标的
- `int64_t usable_locked_position`  
可用已锁定标的
- `double profit_price`  
盈亏成本价
- `double buy_cost`  
买入成本
- `double profit_cost`  
盈亏成本
- `uint64_t unknown` [50 - 9]  
(保留字段)

### 5.63.1 详细描述

查询股票持仓情况

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.64 XTPQueryStructuredFundInfoReq结构体 参考

查询分级基金信息结构体

```
#include <xoms_api_struct.h>
```

成员变量

- `XTP_EXCHANGE_TYPE exchange_id`  
交易所代码, 不可为空
- `char sf_ticker[XTP_TICKER_LEN]`  
分级基金母基金代码, 可以为空, 如果为空, 则默认查询所有的分级基金

### 5.64.1 详细描述

查询分级基金信息结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.65 XTPQueryTraderByPageReq结构体 参考

查询成交回报请求-分页查询

```
#include <xoms_api_struct.h>
```

成员变量

- `int64_t req_count`  
需要查询的成交回报条数
- `int64_t reference`  
上一次收到的查询成交回报结果中带回来的索引，如果是从头查询，请置0
- `int64_t reserved`  
保留字段

### 5.65.1 详细描述

查询成交回报请求-分页查询

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.66 XTPQueryTraderReq结构体 参考

查询成交回报请求-查询条件

```
#include <xoms_api_struct.h>
```

成员变量

- `char ticker [XTP_TICKER_LEN]`  
证券代码，可以为空，如果为空，则默认查询时间段内的所有成交回报
- `int64_t begin_time`  
开始时间，格式为YYYYMMDDHHMMSSsss，为0则默认当前交易日0点
- `int64_t end_time`  
结束时间，格式为YYYYMMDDHHMMSSsss，为0则默认当前时间

### 5.66.1 详细描述

查询成交回报请求-查询条件

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.67 XTPQuoteFullInfo结构体 参考

股票行情全量静态信息

```
#include <xquote_api_struct.h>
```

### 成员变量

- [XTP\\_EXCHANGE\\_TYPE](#) exchange\_id  
交易所代码
- [char ticker](#) [XTP\_TICKER\_LEN]  
证券代码
- [char ticker\\_name](#) [XTP\_TICKER\_NAME\_LEN]  
证券名称
- [XTP\\_SECURITY\\_TYPE](#) security\_type  
合约详细类型
- [XTP\\_QUALIFICATION\\_TYPE](#) ticker\_qualification\_class  
合约适当性类别
- [bool is\\_registration](#)  
是否注册制(仅适用创业板股票, 创新企业股票及存托凭证)
- [bool is\\_VIE](#)  
是否具有协议控制架构(仅适用创业板股票, 创新企业股票及存托凭证)
- [bool is\\_noprofit](#)  
是否尚未盈利(仅适用创业板股票, 创新企业股票及存托凭证)
- [bool is\\_weighted\\_voting\\_rights](#)  
是否存在投票权差异(仅适用创业板股票, 创新企业股票及存托凭证)
- [bool is\\_have\\_price\\_limit](#)  
是否有涨跌幅限制(注: 不提供具体幅度, 可通过涨跌停价和昨收价来计算幅度)
- [double upper\\_limit\\_price](#)  
涨停价 (仅在有涨跌幅限制时有效)
- [double lower\\_limit\\_price](#)  
跌停价 (仅在有涨跌幅限制时有效)
- [double pre\\_close\\_price](#)  
昨收价
- [double price\\_tick](#)  
价格最小变动价位
- [int32\\_t bid\\_qty\\_upper\\_limit](#)  
限价买委托数量上限
- [int32\\_t bid\\_qty\\_lower\\_limit](#)  
限价买委托数量下限
- [int32\\_t bid\\_qty\\_unit](#)  
限价买数量单位
- [int32\\_t ask\\_qty\\_upper\\_limit](#)  
限价卖委托数量上限
- [int32\\_t ask\\_qty\\_lower\\_limit](#)  
限价卖委托数量下限
- [int32\\_t ask\\_qty\\_unit](#)  
限价卖数量单位
- [int32\\_t market\\_bid\\_qty\\_upper\\_limit](#)  
市价买委托数量上限

- `int32_t market_bid_qty_lower_limit`  
市价买委托数量下限
- `int32_t market_bid_qty_unit`  
市价买数量单位
- `int32_t market_ask_qty_upper_limit`  
市价卖委托数量上限
- `int32_t market_ask_qty_lower_limit`  
市价卖委托数量上限
- `int32_t market_ask_qty_unit`  
市价卖数量单位
- `XTP_SECURITY_STATUS security_status`  
证券状态
- `uint32_t unknown1`  
保留字段
- `uint64_t unknown [3]`  
保留字段

### 5.67.1 详细描述

股票行情全量静态信息

该结构体的文档由以下文件生成:

- `xquote_api_struct.h`

## 5.68 XTPQuoteStaticInfo结构体 参考

股票行情静态信息

```
#include <xquote_api_struct.h>
```

### 成员变量

- `XTP_EXCHANGE_TYPE exchange_id`  
交易所代码
- `char ticker [XTP_TICKER_LEN]`  
合约代码（不包含交易所信息），不带空格，以'\0'结尾
- `char ticker_name [XTP_TICKER_NAME_LEN]`  
合约名称
- `XTP_TICKER_TYPE ticker_type`  
合约类型
- `double pre_close_price`  
昨收盘
- `double upper_limit_price`  
涨停板价
- `double lower_limit_price`  
跌停板价
- `double price_tick`  
最小变动价位
- `int32_t buy_qty_unit`  
合约最小交易量(买)
- `int32_t sell_qty_unit`  
合约最小交易量(卖)

### 5.68.1 详细描述

股票行情静态信息

该结构体的文档由以下文件生成:

- [xquote\\_api\\_struct.h](#)

## 5.69 XTPRsplInfoStruct结构体 参考

响应信息

```
#include <xtp_api_struct_common.h>
```

成员变量

- [int32\\_t error\\_id](#)  
错误代码
- [char error\\_msg \[XTP\\_ERR\\_MSG\\_LEN\]](#)  
错误信息

### 5.69.1 详细描述

响应信息

该结构体的文档由以下文件生成:

- [xtp\\_api\\_struct\\_common.h](#)

## 5.70 XTPSpecificTickerStruct结构体 参考

指定的合约

```
#include <xquote_api_struct.h>
```

成员变量

- [XTP\\_EXCHANGE\\_TYPE exchange\\_id](#)  
交易所代码
- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
合约代码（不包含交易所信息）例如“600000”，不带空格，以‘0’结尾

### 5.70.1 详细描述

指定的合约

该结构体的文档由以下文件生成:

- [xquote\\_api\\_struct.h](#)

## 5.71 XTPStructuredFundInfo结构体 参考

查询分级基金信息响应结构体

```
#include <xoms_api_struct.h>
```

### 成员变量

- [XTP\\_EXCHANGE\\_TYPE](#) `exchange_id`  
交易所代码
- `char` [sf\\_ticker](#) [[XTP\\_TICKER\\_LEN](#)]  
分级基金母基金代码
- `char` [sf\\_ticker\\_name](#) [[XTP\\_TICKER\\_NAME\\_LEN](#)]  
分级基金母基金名称
- `char` [ticker](#) [[XTP\\_TICKER\\_LEN](#)]  
分级基金子基金代码
- `char` [ticker\\_name](#) [[XTP\\_TICKER\\_NAME\\_LEN](#)]  
分级基金子基金名称
- [XTP\\_SPLIT\\_MERGE\\_STATUS](#) `split_merge_status`  
基金允许拆分合并状态
- `uint32_t` [ratio](#)  
拆分合并比例
- `uint32_t` [min\\_split\\_qty](#)  
最小拆分数量
- `uint32_t` [min\\_merge\\_qty](#)  
最小合并数量
- `double` [net\\_price](#)  
基金净值

### 5.71.1 详细描述

查询分级基金信息响应结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)



## 5.72 XTPTickByTickEntrust结构体 参考

逐笔委托

```
#include <xquote_api_struct.h>
```

成员变量

- `int32_t channel_no`  
频道代码
- `int64_t seq`
- `double price`  
委托价格
- `int64_t qty`
- `char side`
- `char ord_type`
- `int64_t order_no`

### 5.72.1 详细描述

逐笔委托

### 5.72.2 结构体成员变量说明

#### 5.72.2.1 ord\_type

```
char ord_type
```

SH: 'A': 增加; 'D': 删除 SZ: 订单类别: '1': 市价; '2': 限价; 'U': 本方最优

#### 5.72.2.2 order\_no

```
int64_t order_no
```

SH: 原始订单号 SZ: 无意义

#### 5.72.2.3 qty

```
int64_t qty
```

SH: 剩余委托数量(balance) SZ: 委托数量

#### 5.72.2.4 seq

```
int64_t seq
```

SH: 委托序号(委托单独编号, 同一channel\_no内连续) SZ: 委托序号(委托成交统一编号, 同一channel\_no内连续)

#### 5.72.2.5 side

```
char side
```

SH: 'B':买; 'S':卖 SZ: '1':买; '2':卖; 'G':借入; 'F':出借

该结构体的文档由以下文件生成:

- [xquote\\_api\\_struct.h](#)

### 5.73 XTPTickByTickStruct结构体 参考

逐笔数据信息

```
#include <xquote_api_struct.h>
```

成员变量

- [XTP\\_EXCHANGE\\_TYPE exchange\\_id](#)  
交易所代码
- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
合约代码（不包含交易所信息），不带空格，以'\0'结尾
- [int64\\_t seq](#)
- [int64\\_t data\\_time](#)  
委托时间 *or* 成交时间
- [XTP\\_TBT\\_TYPE type](#)  
委托 *or* 成交
- ```
union {
    XTPTickByTickEntrust entrust
    XTPTickByTickTrade trade
};
```

5.73.1 详细描述

逐笔数据信息

5.73.2 结构体成员变量说明

5.73.2.1 seq

```
int64_t seq
```

SH: 业务序号（委托成交统一编号，同一个channel_no内连续，此seq区别于联合体内的seq，channel_no等同于联合体内的channel_no） SZ: 无意义

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.74 XTPTickByTickTrade结构体 参考

逐笔成交

```
#include <xquote_api_struct.h>
```

成员变量

- `int32_t channel_no`
频道代码
- `int64_t seq`
- `double price`
成交价格
- `int64_t qty`
成交量
- `double money`
成交金额(仅适用上交所)
- `int64_t bid_no`
买方订单号
- `int64_t ask_no`
卖方订单号
- `char trade_flag`

5.74.1 详细描述

逐笔成交

5.74.2 结构体成员变量说明

5.74.2.1 seq

```
int64_t seq
```

SH: 成交序号(成交单独编号, 同一channel_no内连续) SZ: 成交序号(委托成交统一编号, 同一channel_no内连续)

5.74.2.2 trade_flag

```
char trade_flag
```

SH: 内外盘标识('B':主动买; 'S':主动卖; 'N':未知) SZ: 成交标识('4':撤; 'F':成交)

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.75 XTPTickerPriceInfo结构体 参考

供查询的最新信息

```
#include <xquote_api_struct.h>
```

成员变量

- [XTP_EXCHANGE_TYPE exchange_id](#)
交易所代码
- [char ticker \[XTP_TICKER_LEN\]](#)
合约代码（不包含交易所信息），不带空格，以'\0'结尾
- [double last_price](#)
最新价

5.75.1 详细描述

供查询的最新信息

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.76 XTPTradeReport结构体 参考

报单成交结构体

```
#include <xoms_api_struct.h>
```

成员变量

- `uint64_t order_xtp_id`
XTP系统订单ID, 此成交回报相关的订单ID, 在XTP系统中唯一
- `uint32_t order_client_id`
报单引用
- `char ticker [XTP_TICKER_LEN]`
合约代码
- `XTP_MARKET_TYPE market`
交易市场
- `uint64_t local_order_id`
订单号, 引入XTPID后, 该字段实际和`order_xtp_id`重复。接口中暂时保留。
- `char exec_id [XTP_EXEC_ID_LEN]`
成交编号, 深交所唯一, 上交所每笔交易唯一, 当发现2笔成交回报拥有相同的`exec_id`, 则可以认为此笔交易自成交
- `double price`
价格, 此次成交的价格
- `int64_t quantity`
数量, 此次成交的数量, 不是累计数量
- `int64_t trade_time`
成交时间, 格式为YYYYMMDDHHMMSSsss
- `double trade_amount`
成交金额, 此次成交的总金额 = $price * quantity$
- `uint64_t report_index`
成交序号 - 回报记录号, 对于单个账户来说, 深交所每个平台 (不同交易品种) 唯一, 上交所唯一, 对于多账户来说, 不唯一
- `char order_exch_id [XTP_ORDER_EXCH_LEN]`
报单编号 - 交易所单号, 上交所为空, 深交所所有此字段
- `TXTPTradeType trade_type`
成交类型 - 成交回报中的执行类型
- ```

union {
 uint32_t u32
 32位字段, 用来兼容老版本api, 用户无需关心
 struct {
 XTP_SIDE_TYPE side
 买卖方向
 XTP_POSITION_EFFECT_TYPE position_effect
 开平标志
 uint8_t reserved1
 预留字段1
 uint8_t reserved2
 预留字段2
 }
};

```
- `XTP_BUSINESS_TYPE business_type`  
业务类型
- `char branch_pbu [XTP_BRANCH_PBU_LEN]`  
交易所交易员代码

### 5.76.1 详细描述

报单成交结构体

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## 5.77 XTPUserTerminalInfoReq结构体 参考

申报用户的ip和mac等信息，仅限授权用户使用

```
#include <xoms_api_struct.h>
```

成员变量

- char [local\\_ip](#) [[XTP\\_INET\\_ADDRESS\\_STR\\_LEN](#)]  
本地IP地址
- char [mac\\_addr](#) [[XTP\\_MAC\\_ADDRESS\\_LEN](#)]  
MAC地址
- char [hd](#) [[XTP\\_HARDDISK\\_SN\\_LEN](#)]  
硬盘序列号
- [XTPTerminalType](#) [term\\_type](#)  
终端类型
- char [internet\\_ip](#) [[XTP\\_INET\\_ADDRESS\\_STR\\_LEN](#)]  
公网IP地址
- [int32\\_t](#) [internet\\_port](#)  
公网端口号
- [XTPVersionType](#) [client\\_version](#)  
客户端版本号
- char [macos\\_sno](#) [[XTP\\_MACOS\\_SNO\\_LEN](#)]  
MacOS系统的序列号，仅为MacOS系统需要填写
- char [unused](#) [27]  
预留

### 5.77.1 详细描述

申报用户的ip和mac等信息，仅限授权用户使用

该结构体的文档由以下文件生成:

- [xoms\\_api\\_struct.h](#)

## Chapter 6

# 文件说明

### 6.1 demo\_test\_quote\_api.cpp 文件参考

定义控制台测试应用程序的入口点

```
#include <string>
#include <map>
#include <iostream>
#include "xtp_quote_api.h"
#include "demo_test_quote_spi.h"
```

#### 函数

- `int main ()`  
`int main() {`

#### 变量

- `XTP::API::QuoteApi * user_api_pointer`  
*UserApi*对象
- `char username [] = "00092"`  
投资者代码
- `char password [] = "888888"`  
用户密码
- `char * ppTicker [] = { "000001" }`  
行情订阅列表
- `int ticker_count = 1`
- `int client_id = 1`  
客户端标识
- `char filepath [] = "c:\\log\\"`  
可读写存储路径

### 6.1.1 详细描述

定义控制台测试应用程序的入口点

作者

中泰证券股份有限公司

### 6.1.2 函数说明

#### 6.1.2.1 main()

```
int main ()
```

```
int main() {
```

测试Demo入口函数

// 初始化UserApi

```
user_api_pointer = XTP::API::QuoteApi::CreateQuoteApi(client_id,filepath,XTP_LOG_LEVEL_DEBUG); // 创建UserApi
```

```
user_spi_pointer->SetHeartBeatInterval(15);//设置心跳超时时间间隔，单位为秒
```

```
user_api_pointer->SetUDPBufferSize(128);//设置UDP接收缓冲区大小，单位为MB
```

```
DemoTestMdSpi* user_spi_pointer = new DemoTestMdSpi(); //创建响应类实例
```

```
user_api_pointer->RegisterSpi(user_spi_pointer); // 注册事件类
```

```
int loginResult = user_api_pointer->Login("192.168.1.1", 6666, username, password, XTP_PROTOCOL_UDP); //采用UDP方式登陆行情订阅服务器
```

```
if (loginResult == 0)
```

```
{ //登录成功
```

```
user_api_pointer->SubscribeMarketData(ppTicker, ticker_count, XTP_EXCHANGE_SZ); //订阅股票行情
```

```
}
```

```
return 0;
```

```
}
```

### 6.1.3 变量说明



### 6.1.3.1 ticker\_count

```
int ticker_count = 1
```

行情订阅数量

## 6.2 demo\_test\_quote\_spi.h 文件参考

Demo自定义客户端行情订阅响应接口类

```
#include "xtp_quote_api.h"
```

结构体

- class [DemoTestMdSpi](#)  
*Demo*自定义行情订阅接口响应类

### 6.2.1 详细描述

Demo自定义客户端行情订阅响应接口类

作者

中泰证券股份有限公司

## 6.3 xoms\_api\_fund\_struct.h 文件参考

定义资金划拨相关结构体类型

```
#include "xtp_api_data_type.h"
#include "xoms_api_struct.h"
#include "xtp_api_struct_common.h"
```

结构体

- struct [XTPFundTransferReq](#)  
用户资金请求
- struct [XTPFundQueryReq](#)  
用户资金查询请求结构体
- struct [XTPFundQueryRsp](#)  
用户资金查询响应结构体

## 宏定义

- `#define XTP_ACCOUNT_PASSWORD_LEN 64`  
用户资金账户的密码字符串长度

## 类型定义

- `typedef struct XTPFundTransferNotice XTPFundTransferAck`  
用户资金划转请求的响应-复用资金通知结构体

### 6.3.1 详细描述

定义资金划拨相关结构体类型

作者

中泰证券股份有限公司

## 6.4 xoms\_api\_struct.h 文件参考

定义交易类相关数据结构

```
#include "xtp_api_data_type.h"
#include "stddef.h"
#include "xtp_api_struct_common.h"
```

## 结构体

- `struct XTPOrderInsertInfo`  
新订单请求
- `struct XTPOrderCancelInfo`  
撤单失败响应消息
- `struct XTPOrderInfo`  
报单响应结构体
- `struct XTPTradeReport`  
报单成交结构体
- `struct XTPQueryOrderReq`  
报单查询 //////////////////////////////////////////////////////////////////// 报单查询请求-条件查询
- `struct XTPQueryOrderByPageReq`  
查询订单请求-分页查询
- `struct XTPQueryReportByExecIdReq`  
成交回报查询 //////////////////////////////////////////////////////////////////// 查询成交报告请求-根据执行编号查询（保留字段）
- `struct XTPQueryTraderReq`  
查询成交回报请求-查询条件
- `struct XTPQueryTraderByPageReq`  
查询成交回报请求-分页查询

- struct [XTPQueryAssetRsp](#)  
账户资金查询响应结构体
- struct [XTPQueryStkPositionReq](#)  
查询股票持仓情况请求结构体
- struct [XTPQueryStkPositionRsp](#)  
查询股票持仓情况
- struct [XTPCreditDebtExtendNotice](#)  
用户展期请求的通知
- struct [XTPFundTransferNotice](#)  
资金内转流水通知
- struct [XTPQueryFundTransferLogReq](#)  
资金内转流水查询请求与响应
- struct [XTPQueryStructuredFundInfoReq](#)  
查询分级基金信息结构体
- struct [XTPStructuredFundInfo](#)  
查询分级基金信息响应结构体
- struct [XTPQueryETFBaseReq](#)
- struct [XTPQueryETFBaseRsp](#)  
查询股票ETF合约基本情况-响应结构体
- struct [XTPQueryETFComponentReq](#)  
查询股票ETF合约成分股信息-请求结构体,请求参数为:交易市场+ETF买卖代码
- struct [XTPQueryETFComponentRspV1](#)  
查询股票ETF成分股信息-响应结构体, 旧版本。
- struct [XTPQueryETFComponentRsp](#)  
查询股票ETF成分股信息-响应结构体
- struct [XTPQueryIPOTickerRsp](#)  
查询当日可申购新股信息
- struct [XTPQueryIPOQuotaRspV1](#)  
查询用户申购额度-旧版
- struct [XTPQueryIPOQuotaRsp](#)  
查询用户申购额度-包含创业板额度
- struct [XTPUserTerminalInfoReq](#)  
申报用户的ip和mac等信息, 仅限授权用户使用
- struct [XTPQueryOptionAuctionInfoReq](#)  
查询期权竞价交易业务参考信息-请求结构体,请求参数为:交易市场+8位期权代码
- struct [XTPQueryOptionAuctionInfoRsp](#)  
查询期权竞价交易业务参考信息
- struct [XTPCombLegStrategy](#)  
期权组合策略的成分合约信息
- struct [XTPQueryCombineStrategyInfoRsp](#)  
查询期权组合策略信息的响应
- struct [XTPOptCombLegInfo](#)  
组合策略腿合约信息结构体
- struct [XTPOptCombPlugin](#)  
期权组合策略报单附加信息结构体
- struct [XTPQueryOptCombPositionReq](#)  
查询期权组合策略持仓情况请求结构体
- struct [XTPQueryOptCombPositionRsp](#)  
查询期权组合策略持仓信息的响应
- struct [XTPQueryOptExecInfoRsp](#)  
查询期权合约行权信息的响应



## 宏定义

- `#define XTP_ACCOUNT_PASSWORD_LEN 64`  
用户资金账户的密码字符串长度

## 类型定义

- `typedef struct XTPOrderInfo XTPQueryOrderRsp`  
报单查询响应结构体
- `typedef struct XTPTradeReport XTPQueryTradeRsp`  
成交回报查询响应结构体
- `typedef struct XTPFundTransferNotice XTPFundTransferLog`  
资金内转流水记录结构体
- `typedef struct XTPQueryETFBaseRsp XTPQueryETFBaseRsp`  
查询股票ETF合约基本情况-响应结构体
- `typedef struct XTPQueryETFComponentReq XTPQueryETFComponentReq`  
查询股票ETF合约成分股信息-请求结构体,请求参数为:交易市场+ETF买卖代码
- `typedef struct XTPOrderCancelInfo XTOptCombOrderCancelInfo`  
期权组合策略撤单错误响应结构体
- `typedef struct XTOptCombLegInfo XTOptCombLegInfo`  
组合策略腿合约信息结构体
- `typedef struct XTOptCombPlugin XTOptCombPlugin`  
期权组合策略报单附加信息结构体
- `typedef struct XTPCrdDebtInfo XTPCrdDebtInfo`  
单条融资融券负债记录信息
- `typedef struct XTPCrdFundInfo XTPCrdFundInfo`  
融资融券特有帐户数据
- `typedef struct XTPClientQueryCrdDebtStockReq XTPClientQueryCrdDebtStockReq`  
融资融券指定证券上的负债未还数量请求结构体
- `typedef struct XTPCrdDebtStockInfo XTPCrdDebtStockInfo`  
融资融券指定证券的融券负债相关信息
- `typedef struct XTPClientQueryCrdPositionStockReq XTPClientQueryCrdPositionStockReq`  
融券头寸证券查询请求结构体
- `typedef struct XTPClientQueryCrdPositionStkInfo XTPClientQueryCrdPositionStkInfo`  
融券头寸证券信息
- `typedef struct XTPClientQueryCrdSurplusStkReqInfo XTPClientQueryCrdSurplusStkReqInfo`  
信用业务融券查询请求结构体
- `typedef struct XTPClientQueryCrdSurplusStkRspInfo XTPClientQueryCrdSurplusStkRspInfo`  
信用业务融券信息
- `typedef struct XTPCreditDebtExtendNotice XTPCreditDebtExtendAck`  
用户展期请求的响应结构
- `typedef struct XTPCrdFundExtraInfo XTPCrdFundExtraInfo`  
融资融券帐户附加信息
- `typedef struct XTPCrdPositionExtraInfo XTPCrdPositionExtraInfo`  
融资融券帐户持仓附加信息
- `typedef struct XTOptCombOrderInfo XTPQueryOptCombOrderRsp`  
期权组合策略报单查询响应结构体
- `typedef struct XTOptCombTradeReport XTPQueryOptCombTradeRsp`  
成交回报查询响应结构体

### 6.4.1 详细描述

定义交易类相关数据结构

作者

中泰证券股份有限公司

## 6.5 xquote\_api\_struct.h 文件参考

定义行情类相关数据结构

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

结构体

- struct [XTPSpecificTickerStruct](#)  
指定的合约
- struct [XTPMarketDataStockExData](#)  
股票、基金、债券等额外数据
- struct [XTPMarketDataOptionExData](#)  
期权额外数据
- struct [XTPMarketDataStruct](#)  
行情
- struct [XTPQuoteStaticInfo](#)  
股票行情静态信息
- struct [OrderBookStruct](#)  
订单簿
- struct [XTPTickByTickEntrust](#)  
逐笔委托
- struct [XTPTickByTickTrade](#)  
逐笔成交
- struct [XTPTickByTickStruct](#)  
逐笔数据信息
- struct [XTPTickerPricelInfo](#)  
供查询的最新信息
- struct [XTPQuoteFullInfo](#)  
股票行情全量静态信息

## 类型定义

- typedef struct [XTPSpecificTickerStruct](#) XTPST  
指定的合约
- typedef struct [XTPMarketDataStruct](#) XTPMD  
行情
- typedef struct [XTPQuoteStaticInfo](#) XTPQSI  
股票行情静态信息
- typedef struct [OrderBookStruct](#) XTPOB  
订单簿
- typedef struct [XTPTickByTickStruct](#) XTPTBT  
逐笔数据信息
- typedef struct [XTPTickerPriceInfo](#) XTPTPI  
供查询的最新信息
- typedef struct [XTPQuoteFullInfo](#) XTPQFI  
股票行情全量静态信息

## 枚举

- enum [XTP\\_MARKETDATA\\_TYPE](#) { XTP\_MARKETDATA\_ACTUAL = 0, XTP\_MARKETDATA\_OPTION = 1 }
- [XTP\\_MARKETDATA\\_TYPE](#)是行情快照数据类型

### 6.5.1 详细描述

定义行情类相关数据结构

作者

中泰证券股份有限公司

## 6.6 xtp\_api\_data\_type.h 文件参考

定义兼容数据基本类型

## 宏定义

- #define [MAX\\_TGW\\_CNT\\_PER\\_PBU](#) 10  
每个PBU最多被10个TGW使用。
- #define [XTP\\_VERSION\\_LEN](#) 16  
存放版本号的字符串长度
- #define [XTP\\_TRADING\\_DAY\\_LEN](#) 9  
可交易日字符串长度
- #define [XTP\\_TICKER\\_LEN](#) 16  
存放证券代码的字符串长度
- #define [XTP\\_TICKER\\_NAME\\_LEN](#) 64  
存放证券名称的字符串长度

- `#define XTP_LOCAL_ORDER_LEN` 11  
本地报单编号的字符串长度
- `#define XTP_ORDER_EXCH_LEN` 17  
交易所单号的字符串长度
- `#define XTP_EXEC_ID_LEN` 18  
成交执行编号的字符串长度
- `#define XTP_BRANCH_PBU_LEN` 7  
交易所交易员代码字符串长度
- `#define XTP_ACCOUNT_NAME_LEN` 16  
用户资金账户的字符串长度
- `#define XTP_CREDIT_DEBT_ID_LEN` 33  
信用业务合约负债编号长度
- `#define XTP_INET_ADDRESS_STR_LEN` 64  
IP地址的字符串长度
- `#define XTP_MAC_ADDRESS_LEN` 16  
MAC地址的字符串长度
- `#define XTP_HARDDISK_SN_LEN` 24  
硬盘序列号的字符串长度
- `#define XTP_MACOS_SNO_LEN` 21  
MacOS系统序列号的字符串长度
- `#define XTP_STRATEGE_LEG_NUM` 4  
期权组合策略最多腿数
- `#define XTP_STRATEGY_ID_LEN` 10  
期权组合策略代码字符串长度
- `#define XTP_STRATEGY_NAME_LEN` 32  
期权组合策略名称字符串长度
- `#define XTP_SECONDARY_ORDER_ID_LEN` 18  
期权组合策略组合编码字符串长度
- `#define XTP_CNTRT_COMB_STRA_LIST_LEN` 2048  
期权合约可支持的组合策略列表字符串长度
- `#define XTP_COMBINED_EXECUTION_LEG_NUM` 2  
期权行权合并最多成分合约数量
- `#define XTP_SIDE_BUY` 1  
买（新股申购，ETF买，配股，信用交易中担保品买）
- `#define XTP_SIDE_SELL` 2  
卖（逆回购，ETF卖，信用交易中担保品卖）
- `#define XTP_SIDE_PURCHASE` 7  
申购
- `#define XTP_SIDE_REDEMPTION` 8  
赎回
- `#define XTP_SIDE_SPLIT` 9  
拆分
- `#define XTP_SIDE_MERGE` 10  
合并
- `#define XTP_SIDE_COVER` 11  
改版之后的side的备兑，暂不支持
- `#define XTP_SIDE_FREEZE` 12  
改版之后的side锁定（对应开平标识为开）/解锁（对应开平标识为平）
- `#define XTP_SIDE_MARGIN_TRADE` 21  
融资买入
- `#define XTP_SIDE_SHORT_SELL` 22



- 融券卖出
- `#define XTP_SIDE_REPAY_MARGIN` 23
- 卖券还款
- `#define XTP_SIDE_REPAY_STOCK` 24
- 买券还券
- `#define XTP_SIDE_STOCK_REPAY_STOCK` 26
- 现金还款（不放在普通订单协议，另加请求和查询协议）
- `#define XTP_SIDE_SURSTK_TRANS` 27
- 余券划转
- `#define XTP_SIDE_GRTSTK_TRANSIN` 28
- 担保品转入
- `#define XTP_SIDE_GRTSTK_TRANSOUT` 29
- 担保品转出
- `#define XTP_SIDE_OPT_COMBINE` 31
- 组合策略的组合
- `#define XTP_SIDE_OPT_SPLIT` 32
- 组合策略的拆分
- `#define XTP_SIDE_OPT_SPLIT_FORCE` 33
- 组合策略的管理员强制拆分
- `#define XTP_SIDE_OPT_SPLIT_FORCE_EXCH` 34
- 组合策略的交易所强制拆分
- `#define XTP_SIDE_UNKNOWN` 50
- 未知或者无效买卖方向
- `#define XTP_POSITION_EFFECT_INIT` 0
- 初始值或未知值开平标识，除期权外，均使用此值
- `#define XTP_POSITION_EFFECT_OPEN` 1
- 开
- `#define XTP_POSITION_EFFECT_CLOSE` 2
- 平
- `#define XTP_POSITION_EFFECT_FORCECLOSE` 3
- 强平
- `#define XTP_POSITION_EFFECT_CLOSETODAY` 4
- 平今
- `#define XTP_POSITION_EFFECT_CLOSEYESTERDAY` 5
- 平昨
- `#define XTP_POSITION_EFFECT_FORCEOFF` 6
- 强减
- `#define XTP_POSITION_EFFECT_LOCALFORCECLOSE` 7
- 本地强平
- `#define XTP_POSITION_EFFECT_CREDIT_FORCE_COVER` 8
- 信用业务追保强平
- `#define XTP_POSITION_EFFECT_CREDIT_FORCE_CLEAR` 9
- 信用业务清偿强平
- `#define XTP_POSITION_EFFECT_CREDIT_FORCE_DEBT` 10
- 信用业务合约到期强平
- `#define XTP_POSITION_EFFECT_CREDIT_FORCE_UNCOND` 11
- 信用业务无条件强平
- `#define XTP_POSITION_EFFECT_UNKNOWN` 12
- 未知的开平标识类型
- `#define XTP_TRDT_COMMON` '0'
- 普通成交

- `#define XTP_TRDT_CASH '1'`  
现金替代
- `#define XTP_TRDT_PRIMARY '2'`  
一级市场成交
- `#define XTP_TRDT_CROSS_MKT_CASH '3'`  
跨市场资金成交
- `#define XTP_ORDT_Normal '0'`  
正常
- `#define XTP_ORDT_DeriveFromQuote '1'`  
报价衍生
- `#define XTP_ORDT_DeriveFromCombination '2'`  
组合衍生
- `#define XTP_ORDT_Combination '3'`  
组合报单
- `#define XTP_ORDT_ConditionalOrder '4'`  
条件单
- `#define XTP_ORDT_Swap '5'`  
互换单

## 类型定义

- `typedef char XTPVersionType[XTP_VERSION_LEN]`  
版本号类型
- `typedef enum XTP_LOG_LEVEL XTP_LOG_LEVEL`  
`XTP_LOG_LEVEL`是日志输出级别类型
- `typedef enum XTP_PROTOCOL_TYPE XTP_PROTOCOL_TYPE`  
`XTP_PROTOCOL_TYPE`是通讯传输协议方式
- `typedef enum XTP_EXCHANGE_TYPE XTP_EXCHANGE_TYPE`  
`XTP_EXCHANGE_TYPE`是交易所类型，行情里使用
- `typedef enum XTP_MARKET_TYPE XTP_MARKET_TYPE`  
`XTP_MARKET_TYPE`市场类型，交易里使用
- `typedef enum XTP_PRICE_TYPE XTP_PRICE_TYPE`  
`XTP_PRICE_TYPE`是价格类型
- `typedef uint8_t XTP_SIDE_TYPE`  
`XTP_SIDE_TYPE`是买卖方向类型
- `typedef uint8_t XTP_POSITION_EFFECT_TYPE`  
`XTP_POSITION_EFFECT_TYPE`是开平标识类型
- `typedef enum XTP_ORDER_ACTION_STATUS_TYPE XTP_ORDER_ACTION_STATUS_TYPE`  
`XTP_ORDER_ACTION_STATUS_TYPE`是报单操作状态类型
- `typedef enum XTP_ORDER_STATUS_TYPE XTP_ORDER_STATUS_TYPE`  
`XTP_ORDER_STATUS_TYPE`是报单状态类型
- `typedef enum XTP_ORDER_SUBMIT_STATUS_TYPE XTP_ORDER_SUBMIT_STATUS_TYPE`  
`XTP_ORDER_SUBMIT_STATUS_TYPE`是报单提交状态类型
- `typedef enum XTP_TE_RESUME_TYPE XTP_TE_RESUME_TYPE`  
`XTP_TE_RESUME_TYPE`是公有流（订单响应、成交回报）重传方式
- `typedef enum ETF_REPLACE_TYPE ETF_REPLACE_TYPE`  
`ETF_REPLACE_TYPE`现金替代标识定义
- `typedef enum XTP_TICKER_TYPE XTP_TICKER_TYPE`  
`XTP_TICKER_TYPE`证券类型
- `typedef enum XTP_BUSINESS_TYPE XTP_BUSINESS_TYPE`

- XTP\_BUSINESS\_TYPE* 证券业务类型
- typedef enum [XTP\\_ACCOUNT\\_TYPE](#) *XTP\_ACCOUNT\_TYPE*  
*XTP\_ACCOUNT\_TYPE* 账户类型
- typedef enum [XTP\\_FUND\\_TRANSFER\\_TYPE](#) *XTP\_FUND\_TRANSFER\_TYPE*  
*XTP\_FUND\_TRANSFER\_TYPE* 是资金流转方向类型
- typedef enum [XTP\\_FUND\\_QUERY\\_TYPE](#) *XTP\_FUND\_QUERY\_TYPE*  
*XTP\_FUND\_QUERY\_TYPE* 是柜台资金查询类型
- typedef enum [XTP\\_FUND\\_OPER\\_STATUS](#) *XTP\_FUND\_OPER\_STATUS*  
*XTP\_FUND\_OPER\_STATUS* 柜台资金操作结果
- typedef enum [XTP\\_DEBT\\_EXTEND\\_OPER\\_STATUS](#) *XTP\_DEBT\_EXTEND\_OPER\_STATUS*  
*XTP\_DEBT\_EXTEND\_OPER\_STATUS* 柜台负债展期操作状态
- typedef enum [XTP\\_SPLIT\\_MERGE\\_STATUS](#) *XTP\_SPLIT\_MERGE\_STATUS*  
*XTP\_SPLIT\_MERGE\_STATUS* 是一个基金当天拆分合并状态类型
- typedef enum [XTP\\_TBT\\_TYPE](#) *XTP\_TBT\_TYPE*  
*XTP\_TBT\_TYPE* 是一个逐笔回报类型
- typedef enum [XTP\\_OPT\\_CALL\\_OR\\_PUT\\_TYPE](#) *XTP\_OPT\_CALL\_OR\_PUT\_TYPE*  
*XTP\_OPT\_CALL\_OR\_PUT\_TYPE* 是一个认沽或认购类型
- typedef enum [XTP\\_OPT\\_EXERCISE\\_TYPE](#) *XTP\_OPT\_EXERCISE\_TYPE*  
*XTP\_OPT\_EXERCISE\_TYPE* 是一个行权方式类型
- typedef enum [XTP\\_POSITION\\_DIRECTION\\_TYPE](#) *XTP\_POSITION\_DIRECTION\_TYPE*  
*XTP\_POSITION\_DIRECTION\_TYPE* 是一个持仓方向类型
- typedef enum [XTP\\_OPT\\_COVERED\\_OR\\_UNCOVERED](#) *XTP\_OPT\_COVERED\_OR\_UNCOVERED*  
*XTP\_OPT\_COVERED\_OR\_UNCOVERED* 是否备兑的标签
- typedef enum [XTP\\_CRD\\_CR\\_STATUS](#) *XTP\_CRD\_CR\_STATUS*  
*XTP\_CRD\_CASH\_REPAY\_STATUS* 是一个融资融券直接还款状态类型
- typedef enum [XTP\\_OPT\\_POSITION\\_TYPE](#) *XTP\_OPT\_POSITION\_TYPE*  
*XTP\_OPT\_POSITION\_TYPE* 是一个期权持仓类型
- typedef char [TXTPTradeType](#)  
*TXTPTradeType* 是成交类型类型
- typedef char [TXTPOrderType](#)  
*TXTPOrderType* 是报单类型类型
- typedef enum [XTP\\_EXPIRE\\_DATE\\_TYPE](#) *XTP\_EXPIRE\_DATE\_TYPE*  
*XTP\_EXPIRE\_DATE\_TYPE* 是一个期权组合策略合约到期日要求类型
- typedef enum [XTP\\_UNDERLYING\\_TYPE](#) *XTP\_UNDERLYING\_TYPE*  
*XTP\_UNDERLYING\_TYPE* 是一个期权组合策略标的要求类型
- typedef enum [XTP\\_AUTO\\_SPLIT\\_TYPE](#) *XTP\_AUTO\_SPLIT\_TYPE*  
*XTP\_AUTO\_SPLIT\_TYPE* 是一个期权组合策略自动解除枚举类型
- typedef char [TXTPExerciseSeqType](#)  
行权价顺序类型，从1开始，1表示行权价最高，2次之。如果行权价相同，则填写相同数字，用A表示行权价大于等于B，B大于等于C依次类推（C、D）
- typedef enum [XTP\\_QUALIFICATION\\_TYPE](#) *XTP\_QUALIFICATION\_TYPE*  
*XTP\_QUALIFICATION\_TYPE* 是一个证券适当性枚举类型
- typedef enum [XTP\\_SECURITY\\_TYPE](#) *XTP\_SECURITY\_TYPE*  
*XTP\_SECURITY\_TYPE* 是一个证券详细分类枚举类型
- typedef enum [XTP\\_POSITION\\_SECURITY\\_TYPE](#) *XTP\_POSITION\_SECURITY\_TYPE*  
*XTP\_POSITION\_SECURITY\_TYPE* 是一个持仓证券枚举类型
- typedef enum [XTP\\_SECURITY\\_STATUS](#) *XTP\_SECURITY\_STATUS*  
*XTP\_SECURITY\_STATUS* 是一个证券状态枚举类型

## 枚举

- enum `XTP_LOG_LEVEL` {  
`XTP_LOG_LEVEL_FATAL`, `XTP_LOG_LEVEL_ERROR`, `XTP_LOG_LEVEL_WARNING`, `XTP_LOG_LEVEL_INFO`,  
`XTP_LOG_LEVEL_DEBUG`, `XTP_LOG_LEVEL_TRACE` }  
`XTP_LOG_LEVEL`是日志输出级别类型
- enum `XTP_PROTOCOL_TYPE` { `XTP_PROTOCOL_TCP` = 1, `XTP_PROTOCOL_UDP` }  
`XTP_PROTOCOL_TYPE`是通讯传输协议方式
- enum `XTP_EXCHANGE_TYPE` { `XTP_EXCHANGE_SH` = 1, `XTP_EXCHANGE_SZ`, `XTP_EXCHANGE_UNKNOWN` }  
`XTP_EXCHANGE_TYPE`是交易所类型，行情里使用
- enum `XTP_MARKET_TYPE` { `XTP_MKT_INIT` = 0, `XTP_MKT_SZ_A` = 1, `XTP_MKT_SH_A`, `XTP_MKT_UNKNOWN` }  
`XTP_MARKET_TYPE`市场类型，交易里使用
- enum `XTP_PRICE_TYPE` {  
`XTP_PRICE_LIMIT` = 1, `XTP_PRICE_BEST_OR_CANCEL`, `XTP_PRICE_BEST5_OR_LIMIT`, `XTP_PRICE_BEST5_OR_CANCEL`,  
`XTP_PRICE_ALL_OR_CANCEL`, `XTP_PRICE_FORWARD_BEST`, `XTP_PRICE_REVERSE_BEST_LIMIT`,  
`XTP_PRICE_LIMIT_OR_CANCEL`,  
`XTP_PRICE_TYPE_UNKNOWN` }  
`XTP_PRICE_TYPE`是价格类型
- enum `XTP_ORDER_ACTION_STATUS_TYPE` { `XTP_ORDER_ACTION_STATUS_SUBMITTED` = 1,  
`XTP_ORDER_ACTION_STATUS_ACCEPTED`, `XTP_ORDER_ACTION_STATUS_REJECTED` }  
`XTP_ORDER_ACTION_STATUS_TYPE`是报单操作状态类型
- enum `XTP_ORDER_STATUS_TYPE` {  
`XTP_ORDER_STATUS_INIT` = 0, `XTP_ORDER_STATUS_ALLTRADED` = 1, `XTP_ORDER_STATUS_PARTTRADEDQUEUEING`,  
`XTP_ORDER_STATUS_PARTTRADEDNOTQUEUEING`,  
`XTP_ORDER_STATUS_NOTRADEQUEUEING`, `XTP_ORDER_STATUS_CANCELED`, `XTP_ORDER_STATUS_REJECTED`,  
`XTP_ORDER_STATUS_UNKNOWN` }  
`XTP_ORDER_STATUS_TYPE`是报单状态类型
- enum `XTP_ORDER_SUBMIT_STATUS_TYPE` {  
`XTP_ORDER_SUBMIT_STATUS_INSERT_SUBMITTED` = 1, `XTP_ORDER_SUBMIT_STATUS_INSERT_ACCEPTED`,  
`XTP_ORDER_SUBMIT_STATUS_INSERT_REJECTED`, `XTP_ORDER_SUBMIT_STATUS_CANCEL_SUBMITTED`,  
`XTP_ORDER_SUBMIT_STATUS_CANCEL_REJECTED`, `XTP_ORDER_SUBMIT_STATUS_CANCEL_ACCEPTED`  
}   
`XTP_ORDER_SUBMIT_STATUS_TYPE`是报单提交状态类型
- enum `XTP_TE_RESUME_TYPE` { `XTP_TERT_RESTART` = 0, `XTP_TERT_RESUME`, `XTP_TERT_QUICK` }  
`XTP_TE_RESUME_TYPE`是公有流（订单响应、成交回报）重传方式
- enum `ETF_REPLACE_TYPE` {  
`ERT_CASH_FORBIDDEN` = 0, `ERT_CASH_OPTIONAL`, `ERT_CASH_MUST`, `ERT_CASH_RECOMPUTE_INTER_SZ`,  
`ERT_CASH_MUST_INTER_SZ`, `ERT_CASH_RECOMPUTE_INTER_OTHER`, `ERT_CASH_MUST_INTER_OTHER`,  
`ERT_CASH_RECOMPUTE_INTER_HK`,  
`ERT_CASH_MUST_INTER_HK`, `EPT_INVALID` }  
`ETF_REPLACE_TYPE`现金替代标识定义
- enum `XTP_TICKER_TYPE` {  
`XTP_TICKER_TYPE_STOCK` = 0, `XTP_TICKER_TYPE_INDEX`, `XTP_TICKER_TYPE_FUND`, `XTP_TICKER_TYPE_BOND`,  
`XTP_TICKER_TYPE_OPTION`, `XTP_TICKER_TYPE_TECH_STOCK`, `XTP_TICKER_TYPE_UNKNOWN` }  
`XTP_TICKER_TYPE`证券类型
- enum `XTP_BUSINESS_TYPE` {  
`XTP_BUSINESS_TYPE_CASH` = 0, `XTP_BUSINESS_TYPE_IPOS`, `XTP_BUSINESS_TYPE_REPO`,  
`XTP_BUSINESS_TYPE_ETF`,  
`XTP_BUSINESS_TYPE_MARGIN`, `XTP_BUSINESS_TYPE_DESIGNATION`, `XTP_BUSINESS_TYPE_ALLOTMENT`,  
`XTP_BUSINESS_TYPE_STRUCTURED_FUND_PURCHASE_REDEMPTION`,  
`XTP_BUSINESS_TYPE_STRUCTURED_FUND_SPLIT_MERGE`, `XTP_BUSINESS_TYPE_MONEY_FUND`,  
`XTP_BUSINESS_TYPE_OPTION`, `XTP_BUSINESS_TYPE_EXECUTE`,  
`XTP_BUSINESS_TYPE_FREEZE`, `XTP_BUSINESS_TYPE_OPTION_COMBINE`, `XTP_BUSINESS_TYPE_EXECUTE_COMBINE`,  
`XTP_BUSINESS_TYPE_UNKNOWN` }

- XTP\_BUSINESS\_TYPE* 证券业务类型

  - enum *XTP\_ACCOUNT\_TYPE* { *XTP\_ACCOUNT\_NORMAL* = 0, *XTP\_ACCOUNT\_CREDIT*, *XTP\_ACCOUNT\_DERIVE*, *XTP\_ACCOUNT\_UNKNOWN* }

*XTP\_ACCOUNT\_TYPE* 账户类型
- enum *XTP\_FUND\_TRANSFER\_TYPE* { *XTP\_FUND\_TRANSFER\_OUT* = 0, *XTP\_FUND\_TRANSFER\_IN*, *XTP\_FUND\_INTER\_TRANSFER\_OUT*, *XTP\_FUND\_INTER\_TRANSFER\_IN*, *XTP\_FUND\_TRANSFER\_UNKNOWN* }

*XTP\_FUND\_TRANSFER\_TYPE* 是资金流转方向类型
- enum *XTP\_FUND\_QUERY\_TYPE* { *XTP\_FUND\_QUERY\_JZ* = 0, *XTP\_FUND\_QUERY\_INTERNAL*, *XTP\_FUND\_QUERY\_UNKNOWN* }

*XTP\_FUND\_QUERY\_TYPE* 是柜台资金查询类型
- enum *XTP\_FUND\_OPER\_STATUS* { *XTP\_FUND\_OPER\_PROCESSING* = 0, *XTP\_FUND\_OPER\_SUCCESS*, *XTP\_FUND\_OPER\_FAILED*, *XTP\_FUND\_OPER\_SUBMITTED*, *XTP\_FUND\_OPER\_UNKNOWN* }

*XTP\_FUND\_OPER\_STATUS* 柜台资金操作结果
- enum *XTP\_DEBT\_EXTEND\_OPER\_STATUS* { *XTP\_DEBT\_EXTEND\_OPER\_PROCESSING* = 0, *XTP\_DEBT\_EXTEND\_OPER\_SUBMITTED*, *XTP\_DEBT\_EXTEND\_OPER\_FAILED*, *XTP\_DEBT\_EXTEND\_OPER\_UNKNOWN* }

*XTP\_DEBT\_EXTEND\_OPER\_STATUS* 柜台负债展期操作状态
- enum *XTP\_SPLIT\_MERGE\_STATUS* { *XTP\_SPLIT\_MERGE\_STATUS\_ALLOW* = 0, *XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_SPLIT*, *XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_MERGE*, *XTP\_SPLIT\_MERGE\_STATUS\_FORBIDDEN* }

*XTP\_SPLIT\_MERGE\_STATUS* 是一个基金当天拆合并状态类型
- enum *XTP\_TBT\_TYPE* { *XTP\_TBT\_ENTRUST* = 1, *XTP\_TBT\_TRADE* = 2 }

*XTP\_TBT\_TYPE* 是一个逐笔回报类型
- enum *XTP\_OPT\_CALL\_OR\_PUT\_TYPE* { *XTP\_OPT\_CALL* = 1, *XTP\_OPT\_PUT* = 2 }

*XTP\_OPT\_CALL\_OR\_PUT\_TYPE* 是一个认沽或认购类型
- enum *XTP\_OPT\_EXERCISE\_TYPE\_TYPE* { *XTP\_OPT\_EXERCISE\_TYPE\_EUR* = 1, *XTP\_OPT\_EXERCISE\_TYPE\_AME* = 2 }

*XTP\_OPT\_EXERCISE\_TYPE\_TYPE* 是一个行权方式类型
- enum *XTP\_POSITION\_DIRECTION\_TYPE* { *XTP\_POSITION\_DIRECTION\_NET* = 0, *XTP\_POSITION\_DIRECTION\_LONG*, *XTP\_POSITION\_DIRECTION\_SHORT*, *XTP\_POSITION\_DIRECTION\_COVERED* }

*XTP\_POSITION\_DIRECTION\_TYPE* 是一个持仓方向类型
- enum *XTP\_OPT\_COVERED\_OR\_UNCOVERED* { *XTP\_POSITION\_UNCOVERED* = 0, *XTP\_POSITION\_COVERED* }

*XTP\_OPT\_COVERED\_OR\_UNCOVERED* 是否备兑的标签
- enum *XTP\_CRD\_CR\_STATUS* { *XTP\_CRD\_CR\_INIT* = 0, *XTP\_CRD\_CR\_SUCCESS*, *XTP\_CRD\_CR\_FAILED* }

*XTP\_CRD\_CASH\_REPAY\_STATUS* 是一个融资融券直接还款状态类型
- enum *XTP\_OPT\_POSITION\_TYPE* { *XTP\_OPT\_POSITION\_TYPE\_CONTRACT* = 0, *XTP\_OPT\_POSITION\_TYPE\_COMBINE* = 1 }

*XTP\_OPT\_POSITION\_TYPE* 是一个期权持仓类型
- enum *XTP\_ORDER\_DETAIL\_TYPE* { *XTP\_ORDER\_DETAIL\_TYPE\_NEW\_ORDER* = 0, *XTP\_ORDER\_DETAIL\_TYPE\_CANCEL* = 1, *XTP\_ORDER\_DETAIL\_TYPE\_OPT\_COMB\_NEW\_ORDER* = 2, *XTP\_ORDER\_DETAIL\_TYPE\_OPT\_COMB\_CANCEL* = 3 }

*XTP\_ORDER\_TYPE* 是一个订单的类型
- enum *XTPTerminalType* { *XTP\_TERMINAL\_PC* = 1, *XTP\_TERMINAL\_ANDROID*, *XTP\_TERMINAL\_IOS*, *XTP\_TERMINAL\_WP*, *XTP\_TERMINAL\_STATION*, *XTP\_TERMINAL\_TEL*, *XTP\_TERMINAL\_PC\_LINUX* }

*XTPTerminalType* 是一种终端类型枚举, 仅供授权系统使用
- enum *XTP\_EXPIRE\_DATE\_TYPE* { *XTP\_EXP\_DATE\_SAME* = 0, *XTP\_EXP\_DATE\_DIFF*, *XTP\_EXP\_DATE\_NON* }

- XTP\_EXPIRE\_DATE\_TYPE*是一个期权组合策略合约到期日要求类型
- enum *XTP\_UNDERLYING\_TYPE* { *XTP\_UNDERLYING\_SAME* = 0, *XTP\_UNDERLYING\_DIFF*, *XTP\_UNDERLYING\_NON* }
- XTP\_UNDERLYING\_TYPE*是一个期权组合策略标的要求类型
- enum *XTP\_AUTO\_SPLIT\_TYPE* { *XTP\_AUTO\_SPLIT\_EXPDAY* = 0, *XTP\_AUTO\_SPLIT\_PREDAY*, *XTP\_AUTO\_SPLIT\_PRE2DAY*, *XTP\_AUTO\_SPLIT\_NON* }
- XTP\_AUTO\_SPLIT\_TYPE*是一个期权组合策略自动解除枚举类型
- enum *XTP\_QUALIFICATION\_TYPE* { *XTP\_QUALIFICATION\_PUBLIC* = 0, *XTP\_QUALIFICATION\_COMMON* = 1, *XTP\_QUALIFICATION\_ORGANIZATION* = 2, *XTP\_QUALIFICATION\_UNKNOWN* = 3 }
- XTP\_QUALIFICATION\_TYPE*是一个证券适当性枚举类型
- enum *XTP\_SECURITY\_TYPE* {  
*XTP\_SECURITY\_MAIN\_BOARD* = 0, *XTP\_SECURITY\_SECOND\_BOARD*, *XTP\_SECURITY\_STARTUP\_BOARD*,  
*XTP\_SECURITY\_INDEX*,  
*XTP\_SECURITY\_TECH\_BOARD* = 4, *XTP\_SECURITY\_STATE\_BOND* = 5, *XTP\_SECURITY\_ENTERPRICE\_BOND* = 6, *XTP\_SECURITY\_COMPANEY\_BOND* = 7,  
*XTP\_SECURITY\_CONVERTABLE\_BOND* = 8, *XTP\_SECURITY\_NATIONAL\_BOND\_REVERSE\_REPO* = 12, *XTP\_SECURITY ETF\_SINGLE\_MARKET\_STOCK* = 14, *XTP\_SECURITY ETF\_INTER\_MARKET\_STOCK*,  
***XTP\_SECURITY ETF\_CROSS\_BORDER\_STOCK*** = 16, *XTP\_SECURITY ETF\_SINGLE\_MARKET\_BOND* = 17, *XTP\_SECURITY ETF\_GOLD* = 19, *XTP\_SECURITY\_STRUCTURED\_FUND\_CHILD* = 24,  
*XTP\_SECURITY\_SZSE\_RECREATION\_FUND* = 26, *XTP\_SECURITY\_STOCK\_OPTION* = 29, *XTP\_SECURITY ETF\_OPTION* = 30, *XTP\_SECURITY\_ALLOTMENT* = 100,  
*XTP\_SECURITY\_MONETARY\_FUND\_SHCR* = 110, *XTP\_SECURITY\_MONETARY\_FUND\_SHTR* = 111,  
*XTP\_SECURITY\_MONETARY\_FUND\_SZ* = 112, *XTP\_SECURITY\_OTHERS* = 255 }
- XTP\_SECURITY\_TYPE*是一个证券详细分类枚举类型
- enum *XTP\_POSITION\_SECURITY\_TYPE* { *XTP\_POSITION\_SECURITY\_NORMAL* = 0, *XTP\_POSITION\_SECURITY\_PLACE* = 1, *XTP\_POSITION\_SECURITY\_UNKNOWN* = 2 }
- XTP\_POSITION\_SECURITY\_TYPE*是一个持仓证券枚举类型
- enum *XTP\_SECURITY\_STATUS* {  
*XTP\_SECURITY\_STATUS\_ST* = 0, *XTP\_SECURITY\_STATUS\_N\_IPO*, *XTP\_SECURITY\_STATUS\_COMMON*,  
*XTP\_SECURITY\_STATUS\_RESUME*,  
*XTP\_SECURITY\_STATUS\_DELISTING* = 10, *XTP\_SECURITY\_STATUS\_OTHERS* = 255 }
- XTP\_SECURITY\_STATUS*是一个证券状态枚举类型

## 6.6.1 详细描述

定义兼容数据基本类型

作者

中泰证券股份有限公司

## 6.6.2 宏定义说明

### 6.6.2.1 XTP\_SIDE\_STOCK\_REPAY\_STOCK

```
#define XTP_SIDE_STOCK_REPAY_STOCK 26
```

现金还款（不放在普通订单协议，另加请求和查询协议）

现券还券

### 6.6.3 枚举类型说明

#### 6.6.3.1 ETF\_REPLACE\_TYPE

enum [ETF\\_REPLACE\\_TYPE](#)

ETF\_REPLACE\_TYPE现金替代标识定义

枚举值

|                                |                                  |
|--------------------------------|----------------------------------|
| ERT_CASH_FORBIDDEN             | 禁止现金替代                           |
| ERT_CASH_OPTIONAL              | 可以现金替代                           |
| ERT_CASH_MUST                  | 必须现金替代                           |
| ERT_CASH_RECOMPUTE_INTER_SZ    | 深市退补现金替代                         |
| ERT_CASH_MUST_INTER_SZ         | 深市必须现金替代                         |
| ERT_CASH_RECOMPUTE_INTER_OTHER | 非沪深市场成分证券退补现金替代（不适用于跨沪深港ETF产品）   |
| ERT_CASH_MUST_INTER_OTHER      | 表示非沪深市场成份证券必须现金替代（不适用于跨沪深港ETF产品） |
| ERT_CASH_MUST_INTER_HK         | 港市退补现金替代（仅适用于跨沪深港ETF产品）          |
| EPT_INVALID                    | 港市必须现金替代（仅适用于跨沪深港ETF产品） 无效值      |

#### 6.6.3.2 XTP\_ACCOUNT\_TYPE

enum [XTP\\_ACCOUNT\\_TYPE](#)

XTP\_ACCOUNT\_TYPE账户类型

枚举值

|                     |        |
|---------------------|--------|
| XTP_ACCOUNT_NORMAL  | 普通账户   |
| XTP_ACCOUNT_CREDIT  | 信用账户   |
| XTP_ACCOUNT_DERIVE  | 衍生品账户  |
| XTP_ACCOUNT_UNKNOWN | 未知账户类型 |

#### 6.6.3.3 XTP\_AUTO\_SPLIT\_TYPE

enum [XTP\\_AUTO\\_SPLIT\\_TYPE](#)

XTP\_AUTO\_SPLIT\_TYPE是一个期权组合策略自动解除枚举类型

枚举值

|                        |          |
|------------------------|----------|
| XTP_AUTO_SPLIT_EXPDAY  | 到期日自动解除  |
| XTP_AUTO_SPLIT_PREDAY  | E-1日自动解除 |
| XTP_AUTO_SPLIT_PRE2DAY | E-2日自动解除 |
| XTP_AUTO_SPLIT_NON     | 无效值      |

#### 6.6.3.4 XTP\_BUSINESS\_TYPE

enum XTP\_BUSINESS\_TYPE

XTP\_BUSINESS\_TYPE证券业务类型

枚举值

|                                                            |                                        |
|------------------------------------------------------------|----------------------------------------|
| XTP_BUSINESS_TYPE_CASH                                     | 普通股票业务（股票买卖，ETF买卖，沪市交易型货币基金等）          |
| XTP_BUSINESS_TYPE_IPOS                                     | 新股申购业务（对应的price type需选择限价类型）           |
| XTP_BUSINESS_TYPE_REPO                                     | 回购业务（国债逆回购业务对应的price type填为限价，side填为卖） |
| XTP_BUSINESS_TYPE ETF                                      | ETF申赎业务                                |
| XTP_BUSINESS_TYPE_MARGIN                                   | 融资融券业务                                 |
| XTP_BUSINESS_TYPE DESIGNATION                              | 转托管（未支持）                               |
| XTP_BUSINESS_TYPE_ALLOTMENT                                | 配股业务（对应的price type需选择限价类型,side填为买）     |
| XTP_BUSINESS_TYPE_STRUCTURED_FUND_↔<br>PURCHASE_REDEMPTION | 分级基金申赎业务                               |
| XTP_BUSINESS_TYPE_STRUCTURED_FUND_↔<br>SPLIT_MERGE         | 分级基金拆分合并业务                             |
| XTP_BUSINESS_TYPE_MONEY_FUND                               | 货币基金申赎业务（暂未支持，沪市交易型货币基金的买卖请使用普通股票业务）   |
| XTP_BUSINESS_TYPE_OPTION                                   | 期权业务                                   |
| XTP_BUSINESS_TYPE_EXECUTE                                  | 行权                                     |
| XTP_BUSINESS_TYPE_FREEZE                                   | 锁定解锁，暂不支持                              |
| XTP_BUSINESS_TYPE_OPTION_COMBINE                           | 期权组合策略 组合和拆分业务                         |
| XTP_BUSINESS_TYPE_EXECUTE_COMBINE                          | 期权行权合并业务                               |
| XTP_BUSINESS_TYPE_UNKNOWN                                  | 未知类型                                   |

#### 6.6.3.5 XTP\_CRD\_CR\_STATUS

enum XTP\_CRD\_CR\_STATUS



XTP\_CRD\_CASH\_REPAY\_STATUS是一个融资融券直接还款状态类型

枚举值

|                    |          |
|--------------------|----------|
| XTP_CRD_CR_INIT    | 初始、未处理状态 |
| XTP_CRD_CR_SUCCESS | 已成功处理状态  |
| XTP_CRD_CR_FAILED  | 处理失败状态   |

#### 6.6.3.6 XTP\_DEBT\_EXTEND\_OPER\_STATUS

enum [XTP\\_DEBT\\_EXTEND\\_OPER\\_STATUS](#)

XTP\_DEBT\_EXTEND\_OPER\_STATUS柜台负债展期操作状态

枚举值

|                                 |              |
|---------------------------------|--------------|
| XTP_DEBT_EXTEND_OPER_PROCESSING | XTP已收到，正在处理中 |
| XTP_DEBT_EXTEND_OPER_SUBMITTED  | 已提交到集中柜台处理   |
| XTP_DEBT_EXTEND_OPER_SUCCESS    | 成功           |
| XTP_DEBT_EXTEND_OPER_FAILED     | 失败           |
| XTP_DEBT_EXTEND_OPER_UNKNOWN    | 未知           |

#### 6.6.3.7 XTP\_EXCHANGE\_TYPE

enum [XTP\\_EXCHANGE\\_TYPE](#)

XTP\_EXCHANGE\_TYPE是交易所类型，行情里使用

枚举值

|                      |           |
|----------------------|-----------|
| XTP_EXCHANGE_SH      | 上证        |
| XTP_EXCHANGE_SZ      | 深证        |
| XTP_EXCHANGE_UNKNOWN | 不存在的交易所类型 |

#### 6.6.3.8 XTP\_EXPIRE\_DATE\_TYPE

enum [XTP\\_EXPIRE\\_DATE\\_TYPE](#)

XTP\_EXPIRE\_DATE\_TYPE是一个期权组合策略合约到期日要求类型

枚举值

|                   |        |
|-------------------|--------|
| XTP_EXP_DATE_SAME | 相同到期日  |
| XTP_EXP_DATE_DIFF | 不同到期日  |
| XTP_EXP_DATE_NON  | 无到期日要求 |

#### 6.6.3.9 XTP\_FUND\_OPER\_STATUS

enum [XTP\\_FUND\\_OPER\\_STATUS](#)

XTP\_FUND\_OPER\_STATUS柜台资金操作结果

枚举值

|                          |              |
|--------------------------|--------------|
| XTP_FUND_OPER_PROCESSING | XTP已收到，正在处理中 |
| XTP_FUND_OPER_SUCCESS    | 成功           |
| XTP_FUND_OPER_FAILED     | 失败           |
| XTP_FUND_OPER_SUBMITTED  | 已提交到集中柜台处理   |
| XTP_FUND_OPER_UNKNOWN    | 未知           |

#### 6.6.3.10 XTP\_FUND\_QUERY\_TYPE

enum [XTP\\_FUND\\_QUERY\\_TYPE](#)

XTP\_FUND\_QUERY\_TYPE是柜台资金查询类型

枚举值

|                         |                     |
|-------------------------|---------------------|
| XTP_FUND_QUERY_JZ       | 查询金证主柜台可转资金         |
| XTP_FUND_QUERY_INTERNAL | 查询一账号两中心设置时，对方节点的资金 |
| XTP_FUND_QUERY_UNKNOWN  | 未知类型                |

#### 6.6.3.11 XTP\_FUND\_TRANSFER\_TYPE

enum [XTP\\_FUND\\_TRANSFER\\_TYPE](#)

XTP\_FUND\_TRANSFER\_TYPE是资金流转方向类型

枚举值

|                             |                                                 |
|-----------------------------|-------------------------------------------------|
| XTP_FUND_TRANSFER_OUT       | 转出 从XTP转出到柜台                                    |
| XTP_FUND_TRANSFER_IN        | 转入 从柜台转入XTP                                     |
| XTP_FUND_INTER_TRANSFER_OUT | 跨节点转出 从本XTP节点1，转出到对端XTP节点2，XTP服务器之间划拨，只能跨账户用户使用 |
| XTP_FUND_INTER_TRANSFER_IN  | 跨节点转入 从对端XTP节点2，转入到本XTP节点1，XTP服务器之间划拨，只能跨账户用户使用 |
| XTP_FUND_TRANSFER_UNKNOWN   | 未知类型                                            |

#### 6.6.3.12 XTP\_LOG\_LEVEL

enum [XTP\\_LOG\\_LEVEL](#)

XTP\_LOG\_LEVEL是日志输出级别类型

枚举值

|                       |         |
|-----------------------|---------|
| XTP_LOG_LEVEL_FATAL   | 严重错误级别  |
| XTP_LOG_LEVEL_ERROR   | 错误级别    |
| XTP_LOG_LEVEL_WARNING | 警告级别    |
| XTP_LOG_LEVEL_INFO    | info级别  |
| XTP_LOG_LEVEL_DEBUG   | debug级别 |
| XTP_LOG_LEVEL_TRACE   | trace级别 |

#### 6.6.3.13 XTP\_MARKET\_TYPE

enum [XTP\\_MARKET\\_TYPE](#)

XTP\_MARKET\_TYPE市场类型，交易里使用

枚举值

|                 |          |
|-----------------|----------|
| XTP_MKT_INIT    | 初始化值或者未知 |
| XTP_MKT_SZ_A    | 深圳A股     |
| XTP_MKT_SH_A    | 上海A股     |
| XTP_MKT_UNKNOWN | 未知交易市场类型 |

#### 6.6.3.14 XTP\_OPT\_CALL\_OR\_PUT\_TYPE

enum XTP\_OPT\_CALL\_OR\_PUT\_TYPE

XTP\_OPT\_CALL\_OR\_PUT\_TYPE是一个认沽或认购类型

枚举值

|              |    |
|--------------|----|
| XTP_OPT_CALL | 认购 |
| XTP_OPT_PUT  | 认沽 |

#### 6.6.3.15 XTP\_OPT\_COVERED\_OR\_UNCOVERED

enum XTP\_OPT\_COVERED\_OR\_UNCOVERED

XTP\_OPT\_COVERED\_OR\_UNCOVERED是否备兑的标签

枚举值

|                        |     |
|------------------------|-----|
| XTP_POSITION_UNCOVERED | 非备兑 |
| XTP_POSITION_COVERED   | 备兑  |

#### 6.6.3.16 XTP\_OPT\_EXERCISE\_TYPE\_TYPE

enum XTP\_OPT\_EXERCISE\_TYPE\_TYPE

XTP\_OPT\_EXERCISE\_TYPE\_TYPE是一个行权方式类型

枚举值

|                           |    |
|---------------------------|----|
| XTP_OPT_EXERCISE_TYPE_EUR | 欧式 |
| XTP_OPT_EXERCISE_TYPE_AME | 美式 |

#### 6.6.3.17 XTP\_OPT\_POSITION\_TYPE

enum XTP\_OPT\_POSITION\_TYPE

XTP\_OPT\_POSITION\_TYPE是一个期权持仓类型

枚举值

|                                |        |
|--------------------------------|--------|
| XTP_OPT_POSITION_TYPE_CONTRACT | 单合约持仓  |
| XTP_OPT_POSITION_TYPE_COMBINED | 组合策略持仓 |

#### 6.6.3.18 XTP\_ORDER\_ACTION\_STATUS\_TYPE

enum [XTP\\_ORDER\\_ACTION\\_STATUS\\_TYPE](#)

XTP\_ORDER\_ACTION\_STATUS\_TYPE是报单操作状态类型

枚举值

|                                   |       |
|-----------------------------------|-------|
| XTP_ORDER_ACTION_STATUS_SUBMITTED | 已经提交  |
| XTP_ORDER_ACTION_STATUS_ACCEPTED  | 已经接受  |
| XTP_ORDER_ACTION_STATUS_REJECTED  | 已经被拒绝 |

#### 6.6.3.19 XTP\_ORDER\_DETAIL\_TYPE

enum [XTP\\_ORDER\\_DETAIL\\_TYPE](#)

XTP\_ORDER\_TYPE是一个订单的类型

枚举值

|                                             |        |
|---------------------------------------------|--------|
| XTP_ORDER_DETAIL_TYPE_NEW_ORDER             | 新订单    |
| XTP_ORDER_DETAIL_TYPE_CANCEL_ORDER          | 新订单撤单  |
| XTP_ORDER_DETAIL_TYPE_OPT_COMB_NEW_ORDER    | 组合订单   |
| XTP_ORDER_DETAIL_TYPE_OPT_COMB_CANCEL_ORDER | 组合订单撤单 |

#### 6.6.3.20 XTP\_ORDER\_STATUS\_TYPE

enum [XTP\\_ORDER\\_STATUS\\_TYPE](#)

XTP\_ORDER\_STATUS\_TYPE是报单状态类型

枚举值

|                       |     |
|-----------------------|-----|
| XTP_ORDER_STATUS_INIT | 初始化 |
|-----------------------|-----|

枚举值

|                                        |        |
|----------------------------------------|--------|
| XTP_ORDER_STATUS_ALLTRADED             | 全部成交   |
| XTP_ORDER_STATUS_PARTTRADEDQUEUEING    | 部分成交   |
| XTP_ORDER_STATUS_PARTTRADEDNOTQUEUEING | 部分撤单   |
| XTP_ORDER_STATUS_NOTRADEQUEUEING       | 未成交    |
| XTP_ORDER_STATUS_CANCELED              | 已撤单    |
| XTP_ORDER_STATUS_REJECTED              | 已拒绝    |
| XTP_ORDER_STATUS_UNKNOWN               | 未知订单状态 |

### 6.6.3.21 XTP\_ORDER\_SUBMIT\_STATUS\_TYPE

enum [XTP\\_ORDER\\_SUBMIT\\_STATUS\\_TYPE](#)

XTP\_ORDER\_SUBMIT\_STATUS\_TYPE是报单提交状态类型

枚举值

|                                          |         |
|------------------------------------------|---------|
| XTP_ORDER_SUBMIT_STATUS_INSERT_SUBMITTED | 订单已经提交  |
| XTP_ORDER_SUBMIT_STATUS_INSERT_ACCEPTED  | 订单已经被接受 |
| XTP_ORDER_SUBMIT_STATUS_INSERT_REJECTED  | 订单已经被拒绝 |
| XTP_ORDER_SUBMIT_STATUS_CANCEL_SUBMITTED | 撤单已经提交  |
| XTP_ORDER_SUBMIT_STATUS_CANCEL_REJECTED  | 撤单已经被拒绝 |
| XTP_ORDER_SUBMIT_STATUS_CANCEL_ACCEPTED  | 撤单已经被接受 |

### 6.6.3.22 XTP\_POSITION\_DIRECTION\_TYPE

enum [XTP\\_POSITION\\_DIRECTION\\_TYPE](#)

XTP\_POSITION\_DIRECTION\_TYPE是一个持仓方向类型

枚举值

|                                |               |
|--------------------------------|---------------|
| XTP_POSITION_DIRECTION_NET     | 净             |
| XTP_POSITION_DIRECTION_LONG    | 多（期权则为权利方）    |
| XTP_POSITION_DIRECTION_SHORT   | 空（期权则为义务方）    |
| XTP_POSITION_DIRECTION_COVERED | 备兑（期权则为备兑义务方） |

### 6.6.3.23 XTP\_POSITION\_SECURITY\_TYPE

enum `XTP_POSITION_SECURITY_TYPE`

`XTP_POSITION_SECURITY_TYPE`是一个持仓证券枚举类型

枚举值

|                                              |                  |
|----------------------------------------------|------------------|
| <code>XTP_POSITION_SECURITY_NORMAL</code>    | 普通持仓             |
| <code>XTP_POSITION_SECURITY_PLACEMENT</code> | 配售类型的持仓，包含配股、配债等 |
| <code>XTP_POSITION_SECURITY_UNKNOWN</code>   | 未知类型             |

### 6.6.3.24 XTP\_PRICE\_TYPE

enum `XTP_PRICE_TYPE`

`XTP_PRICE_TYPE`是价格类型

枚举值

|                                           |                                                 |
|-------------------------------------------|-------------------------------------------------|
| <code>XTP_PRICE_LIMIT</code>              | 限价单-沪 / 深 / 沪期权 / 深期权（除普通股票业务外，其余未特指的业务均使用此种类型） |
| <code>XTP_PRICE_BEST_OR_CANCEL</code>     | 即时成交剩余转撤销，市价单-深 / 沪期权 / 深期权                     |
| <code>XTP_PRICE_BEST5_OR_LIMIT</code>     | 最优五档即时成交剩余转限价，市价单-沪                             |
| <code>XTP_PRICE_BEST5_OR_CANCEL</code>    | 最优5档即时成交剩余转撤销，市价单-沪深 / 深期权                      |
| <code>XTP_PRICE_ALL_OR_CANCEL</code>      | 全部成交或撤销,市价单-深 / 沪期权 / 深期权                       |
| <code>XTP_PRICE_FORWARD_BEST</code>       | 本方最优，市价单-深 / 深期权 / 沪科创板                         |
| <code>XTP_PRICE_REVERSE_BEST_LIMIT</code> | 对方最优剩余转限价，市价单-深 / 沪期权 / 深期权 / 沪科创板              |
| <code>XTP_PRICE_LIMIT_OR_CANCEL</code>    | 期权限价申报FOK                                       |
| <code>XTP_PRICE_TYPE_UNKNOWN</code>       | 未知或者无效价格类型                                      |

### 6.6.3.25 XTP\_PROTOCOL\_TYPE

enum `XTP_PROTOCOL_TYPE`

`XTP_PROTOCOL_TYPE`是通讯传输协议方式

枚举值

|                               |                    |
|-------------------------------|--------------------|
| <code>XTP_PROTOCOL_TCP</code> | 采用TCP方式传输          |
| <code>XTP_PROTOCOL_UDP</code> | 采用UDP方式传输(仅行情接口支持) |



### 6.6.3.26 XTP\_QUALIFICATION\_TYPE

enum [XTP\\_QUALIFICATION\\_TYPE](#)

XTP\_QUALIFICATION\_TYPE是一个证券适当性枚举类型

枚举值

|                                |                     |
|--------------------------------|---------------------|
| XTP_QUALIFICATION_PUBLIC       | 公众投资者，合格投资者与机构投资者均可 |
| XTP_QUALIFICATION_COMMON       | 仅合格投资者与公众投资者        |
| XTP_QUALIFICATION_ORGANIZATION | 仅限机构投资者             |
| XTP_QUALIFICATION_UNKNOWN      | 未知，期权等可能为此种类型       |

### 6.6.3.27 XTP\_SECURITY\_STATUS

enum [XTP\\_SECURITY\\_STATUS](#)

XTP\_SECURITY\_STATUS是一个证券状态枚举类型

枚举值

|                               |       |
|-------------------------------|-------|
| XTP_SECURITY_STATUS_ST        | 风险警示板 |
| XTP_SECURITY_STATUS_N_IPO     | 首日上市  |
| XTP_SECURITY_STATUS_COMMON    | 普通    |
| XTP_SECURITY_STATUS_RESUME    | 恢复上市  |
| XTP_SECURITY_STATUS_DELISTING | 退市整理期 |
| XTP_SECURITY_STATUS_OTHERS    | 其他    |

### 6.6.3.28 XTP\_SECURITY\_TYPE

enum [XTP\\_SECURITY\\_TYPE](#)

XTP\_SECURITY\_TYPE是一个证券详细分类枚举类型

枚举值

|                            |       |
|----------------------------|-------|
| XTP_SECURITY_MAIN_BOARD    | 主板股票  |
| XTP_SECURITY_SECOND_BOARD  | 中小板股票 |
| XTP_SECURITY_STARTUP_BOARD | 创业板股票 |
| XTP_SECURITY_INDEX         | 指数    |

枚举值

|                                         |             |
|-----------------------------------------|-------------|
| XTP_SECURITY_TECH_BOARD                 | 科创板股票(上海)   |
| XTP_SECURITY_STATE_BOND                 | 国债          |
| XTP_SECURITY_ENTERPRICE_BOND            | 企业债         |
| XTP_SECURITY_COMPANY_BOND               | 公司债         |
| XTP_SECURITY_CONVERTABLE_BOND           | 转换债券        |
| XTP_SECURITY_NATIONAL_BOND_REVERSE_REPO | 国债逆回购       |
| XTP_SECURITY ETF_SINGLE_MARKET_STOCK    | 本市场股票 ETF   |
| XTP_SECURITY ETF_INTER_MARKET_STOCK     | 跨市场股票 ETF   |
| XTP_SECURITY ETF_SINGLE_MARKET_BOND     | 本市场实物债券 ETF |
| XTP_SECURITY ETF_GOLD                   | 黄金 ETF      |
| XTP_SECURITY_STRUCTURED_FUND_CHILD      | 分级基金子基金     |
| XTP_SECURITY_SZSE_RECREATION_FUND       | 深交所仅申赎基金    |
| XTP_SECURITY_STOCK_OPTION               | 个股期权        |
| XTP_SECURITY ETF_OPTION                 | ETF期权       |
| XTP_SECURITY_ALLOTMENT                  | 配股          |
| XTP_SECURITY_MONETARY_FUND_SHCR         | 上交所申赎型货币基金  |
| XTP_SECURITY_MONETARY_FUND_SHTR         | 上交所交易型货币基金  |
| XTP_SECURITY_MONETARY_FUND_SZ           | 深交所货币基金     |
| XTP_SECURITY_OTHERS                     | 其他          |

#### 6.6.3.29 XTP\_SPLIT\_MERGE\_STATUS

enum XTP\_SPLIT\_MERGE\_STATUS

XTP\_SPLIT\_MERGE\_STATUS是一个基金当天拆分合并状态类型

枚举值

|                                   |             |
|-----------------------------------|-------------|
| XTP_SPLIT_MERGE_STATUS_ALLOW      | 允许拆分和合并     |
| XTP_SPLIT_MERGE_STATUS_ONLY_SPLIT | 只允许拆分，不允许合并 |
| XTP_SPLIT_MERGE_STATUS_ONLY_MERGE | 只允许合并，不允许拆分 |
| XTP_SPLIT_MERGE_STATUS_FORBIDDEN  | 不允许拆分合并     |

#### 6.6.3.30 XTP\_TBT\_TYPE

enum XTP\_TBT\_TYPE

XTP\_TBT\_TYPE是一个逐笔回报类型

枚举值

|                 |      |
|-----------------|------|
| XTP_TBT_ENTRUST | 逐笔委托 |
| XTP_TBT_TRADE   | 逐笔成交 |

### 6.6.3.31 XTP\_TE\_RESUME\_TYPE

enum [XTP\\_TE\\_RESUME\\_TYPE](#)

XTP\_TE\_RESUME\_TYPE是公有流（订单响应、成交回报）重传方式

枚举值

|                  |                         |
|------------------|-------------------------|
| XTP_TERT_RESTART | 从本交易日开始重传               |
| XTP_TERT_RESUME  | 从上次收到的续传（暂未支持）          |
| XTP_TERT_QUICK   | 只传送登录后公有流（订单响应、成交回报）的内容 |

### 6.6.3.32 XTP\_TICKER\_TYPE

enum [XTP\\_TICKER\\_TYPE](#)

XTP\_TICKER\_TYPE证券类型

枚举值

|                            |           |
|----------------------------|-----------|
| XTP_TICKER_TYPE_STOCK      | 普通股票      |
| XTP_TICKER_TYPE_INDEX      | 指数        |
| XTP_TICKER_TYPE_FUND       | 基金        |
| XTP_TICKER_TYPE_BOND       | 债券        |
| XTP_TICKER_TYPE_OPTION     | 期权        |
| XTP_TICKER_TYPE_TECH_STOCK | 科创板股票（上海） |
| XTP_TICKER_TYPE_UNKNOWN    | 未知类型      |

### 6.6.3.33 XTP\_UNDERLYING\_TYPE

enum [XTP\\_UNDERLYING\\_TYPE](#)

XTP\_UNDERLYING\_TYPE是一个期权组合策略标的要求类型

枚举值

|                     |       |
|---------------------|-------|
| XTP_UNDERLYING_SAME | 相同标的  |
| XTP_UNDERLYING_DIFF | 不同标的  |
| XTP_UNDERLYING_NON  | 无标的要求 |

6.6.3.34 XTPTerminalType

enum XTPTerminalType

XTPTerminalType是一种终端类型枚举， 仅供授权系统使用

枚举值

|                       |                           |
|-----------------------|---------------------------|
| XTP_TERMINAL_PC       | "PC",PC-windows及MacOS     |
| XTP_TERMINAL_ANDROID  | "MA",Mobile-Android       |
| XTP_TERMINAL_IOS      | "MI",Mobile-IOS           |
| XTP_TERMINAL_WP       | "MW",Mobile-Windows Phone |
| XTP_TERMINAL_STATION  | "WP",无盘站                  |
| XTP_TERMINAL_TEL      | "TO",电话委托                 |
| XTP_TERMINAL_PC_LINUX | "OH",PC-linux及其他终端        |

6.7 xtp\_api\_struct.h 文件参考

定义业务数据结构

```
#include "xtp_api_struct_common.h"
#include "xquote_api_struct.h"
#include "xoms_api_struct.h"
#include "xoms_api_fund_struct.h"
```

6.7.1 详细描述

定义业务数据结构

作者

中泰证券股份有限公司

## 6.8 xtp\_api\_struct\_common.h 文件参考

定义业务公共数据结构

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

结构体

- struct [XTPRspInfoStruct](#)  
响应信息

宏定义

- #define [XTP\\_ERR\\_MSG\\_LEN](#) 124  
错误信息的字符串长度

类型定义

- typedef struct [XTPRspInfoStruct](#) [XTPRI](#)  
响应信息

### 6.8.1 详细描述

定义业务公共数据结构

作者

中泰证券股份有限公司

## 6.9 xtp\_quote\_api.h 文件参考

定义行情订阅客户端接口

```
#include "xtp_api_struct.h"
```

结构体

- class [QuoteSpi](#)  
行情回调类
- class [QuoteApi](#)  
行情订阅接口类

### 6.9.1 详细描述

定义行情订阅客户端接口

作者

中泰证券股份有限公司



# Index

CreateQuoteApi  
    XTP::API::QuoteApi, [15](#)

demo\_test\_quote\_api.cpp, [99](#)  
    main, [100](#)  
    ticker\_count, [100](#)

demo\_test\_quote\_spi.h, [101](#)

DemoTestMdSpi, [11](#)

ETF\_REPLACE\_TYPE  
    xtp\_api\_data\_type.h, [115](#)

GetApiLastError  
    XTP::API::QuoteApi, [15](#)

GetApiVersion  
    XTP::API::QuoteApi, [15](#)

GetTradingDay  
    XTP::API::QuoteApi, [16](#)

Login  
    XTP::API::QuoteApi, [16](#)

Logout  
    XTP::API::QuoteApi, [17](#)

main  
    demo\_test\_quote\_api.cpp, [100](#)

market  
    XTPOptCombOrderInfo, [62](#)

OnDepthMarketData  
    XTP::API::QuoteSpi, [32](#)

OnDisconnected  
    XTP::API::QuoteSpi, [33](#)

OnError  
    XTP::API::QuoteSpi, [33](#)

OnOrderBook  
    XTP::API::QuoteSpi, [33](#)

OnQueryAllTickers  
    XTP::API::QuoteSpi, [34](#)

OnQueryAllTickersFullInfo  
    XTP::API::QuoteSpi, [34](#)

OnQueryTickersPriceInfo  
    XTP::API::QuoteSpi, [35](#)

OnSubMarketData  
    XTP::API::QuoteSpi, [35](#)

OnSubOrderBook  
    XTP::API::QuoteSpi, [35](#)

OnSubTickByTick  
    XTP::API::QuoteSpi, [40](#)

OnSubscribeAllMarketData  
    XTP::API::QuoteSpi, [36](#)

OnSubscribeAllOptionMarketData  
    XTP::API::QuoteSpi, [36](#)

OnSubscribeAllOptionOrderBook  
    XTP::API::QuoteSpi, [37](#)

OnSubscribeAllOptionTickByTick  
    XTP::API::QuoteSpi, [37](#)

OnSubscribeAllOrderBook  
    XTP::API::QuoteSpi, [39](#)

OnSubscribeAllTickByTick  
    XTP::API::QuoteSpi, [39](#)

OnTickByTick  
    XTP::API::QuoteSpi, [40](#)

OnUnSubMarketData  
    XTP::API::QuoteSpi, [41](#)

OnUnSubOrderBook  
    XTP::API::QuoteSpi, [41](#)

OnUnSubTickByTick  
    XTP::API::QuoteSpi, [44](#)

OnUnSubscribeAllMarketData  
    XTP::API::QuoteSpi, [42](#)

OnUnSubscribeAllOptionMarketData  
    XTP::API::QuoteSpi, [42](#)

OnUnSubscribeAllOptionOrderBook  
    XTP::API::QuoteSpi, [43](#)

OnUnSubscribeAllOptionTickByTick  
    XTP::API::QuoteSpi, [43](#)

OnUnSubscribeAllOrderBook  
    XTP::API::QuoteSpi, [44](#)

OnUnSubscribeAllTickByTick  
    XTP::API::QuoteSpi, [44](#)

ord\_type  
    XTPTickByTickEntrust, [93](#)

order\_no  
    XTPTickByTickEntrust, [93](#)

OrderBookStruct, [13](#)

qty  
    XTPTickByTickEntrust, [93](#)

QueryAllTickers  
    XTP::API::QuoteApi, [17](#)

QueryAllTickersFullInfo  
    XTP::API::QuoteApi, [17](#)

QueryAllTickersPriceInfo  
    XTP::API::QuoteApi, [18](#)

QueryTickersPriceInfo  
    XTP::API::QuoteApi, [18](#)

QuoteApi, [13](#)

QuoteSpi, [31](#)

RegisterSpi

- XTP::API::QuoteApi, 18
- Release
  - XTP::API::QuoteApi, 19
- seq
  - XTPTickByTickEntrust, 93
  - XTPTickByTickStruct, 95
  - XTPTickByTickTrade, 95
- SetHeartBeatInterval
  - XTP::API::QuoteApi, 19
- SetUDPBufferSize
  - XTP::API::QuoteApi, 19
- SetUDPParseThreadAffinity
  - XTP::API::QuoteApi, 20
- SetUDPParseThreadAffinityArray
  - XTP::API::QuoteApi, 20
- SetUDPRecvThreadAffinity
  - XTP::API::QuoteApi, 20
- SetUDPRecvThreadAffinityArray
  - XTP::API::QuoteApi, 21
- SetUDPSeqLogOutPutFlag
  - XTP::API::QuoteApi, 21
- side
  - XTPTickByTickEntrust, 94
- SubscribeAllMarketData
  - XTP::API::QuoteApi, 22
- SubscribeAllOptionMarketData
  - XTP::API::QuoteApi, 22
- SubscribeAllOptionOrderBook
  - XTP::API::QuoteApi, 23
- SubscribeAllOptionTickByTick
  - XTP::API::QuoteApi, 23
- SubscribeAllOrderBook
  - XTP::API::QuoteApi, 24
- SubscribeAllTickByTick
  - XTP::API::QuoteApi, 24
- SubscribeMarketData
  - XTP::API::QuoteApi, 24
- SubscribeOrderBook
  - XTP::API::QuoteApi, 25
- SubscribeTickByTick
  - XTP::API::QuoteApi, 26
- ticker\_count
  - demo\_test\_quote\_api.cpp, 100
- trade\_flag
  - XTPTickByTickTrade, 96
- UnSubscribeAllMarketData
  - XTP::API::QuoteApi, 26
- UnSubscribeAllOptionMarketData
  - XTP::API::QuoteApi, 27
- UnSubscribeAllOptionOrderBook
  - XTP::API::QuoteApi, 27
- UnSubscribeAllOptionTickByTick
  - XTP::API::QuoteApi, 28
- UnSubscribeAllOrderBook
  - XTP::API::QuoteApi, 28
- UnSubscribeAllTickByTick
  - XTP::API::QuoteApi, 28
- XTP::API::QuoteApi, 29
  - UnSubscribeMarketData
    - XTP::API::QuoteApi, 29
  - UnSubscribeOrderBook
    - XTP::API::QuoteApi, 30
  - UnSubscribeTickByTick
    - XTP::API::QuoteApi, 30
- XTP::API::QuoteApi
  - CreateQuoteApi, 15
  - GetApiLastError, 15
  - GetApiVersion, 15
  - GetTradingDay, 16
  - Login, 16
  - Logout, 17
  - QueryAllTickers, 17
  - QueryAllTickersFullInfo, 17
  - QueryAllTickersPriceInfo, 18
  - QueryTickersPriceInfo, 18
  - RegisterSpi, 18
  - Release, 19
  - SetHeartBeatInterval, 19
  - SetUDPBufferSize, 19
  - SetUDPParseThreadAffinity, 20
  - SetUDPParseThreadAffinityArray, 20
  - SetUDPRecvThreadAffinity, 20
  - SetUDPRecvThreadAffinityArray, 21
  - SetUDPSeqLogOutPutFlag, 21
  - SubscribeAllMarketData, 22
  - SubscribeAllOptionMarketData, 22
  - SubscribeAllOptionOrderBook, 23
  - SubscribeAllOptionTickByTick, 23
  - SubscribeAllOrderBook, 24
  - SubscribeAllTickByTick, 24
  - SubscribeMarketData, 24
  - SubscribeOrderBook, 25
  - SubscribeTickByTick, 26
  - UnSubscribeAllMarketData, 26
  - UnSubscribeAllOptionMarketData, 27
  - UnSubscribeAllOptionOrderBook, 27
  - UnSubscribeAllOptionTickByTick, 28
  - UnSubscribeAllOrderBook, 28
  - UnSubscribeAllTickByTick, 29
  - UnSubscribeMarketData, 29
  - UnSubscribeOrderBook, 30
  - UnSubscribeTickByTick, 30
- XTP::API::QuoteSpi
  - OnDepthMarketData, 32
  - OnDisconnected, 33
  - OnError, 33
  - OnOrderBook, 33
  - OnQueryAllTickers, 34
  - OnQueryAllTickersFullInfo, 34
  - OnQueryTickersPriceInfo, 35
  - OnSubMarketData, 35
  - OnSubOrderBook, 35
  - OnSubTickByTick, 40
  - OnSubscribeAllMarketData, 36
  - OnSubscribeAllOptionMarketData, 36



- OnSubscribeAllOptionOrderBook, [37](#)
- OnSubscribeAllOptionTickByTick, [37](#)
- OnSubscribeAllOrderBook, [39](#)
- OnSubscribeAllTickByTick, [39](#)
- OnTickByTick, [40](#)
- OnUnSubMarketData, [41](#)
- OnUnSubOrderBook, [41](#)
- OnUnSubTickByTick, [44](#)
- OnUnSubscribeAllMarketData, [42](#)
- OnUnSubscribeAllOptionMarketData, [42](#)
- OnUnSubscribeAllOptionOrderBook, [43](#)
- OnUnSubscribeAllOptionTickByTick, [43](#)
- OnUnSubscribeAllOrderBook, [44](#)
- OnUnSubscribeAllTickByTick, [44](#)
- XTP\_ACCOUNT\_TYPE
  - [xtp\\_api\\_data\\_type.h, 115](#)
- XTP\_AUTO\_SPLIT\_TYPE
  - [xtp\\_api\\_data\\_type.h, 115](#)
- XTP\_BUSINESS\_TYPE
  - [xtp\\_api\\_data\\_type.h, 116](#)
- XTP\_CRD\_CR\_STATUS
  - [xtp\\_api\\_data\\_type.h, 116](#)
- XTP\_DEBT\_EXTEND\_OPER\_STATUS
  - [xtp\\_api\\_data\\_type.h, 118](#)
- XTP\_EXCHANGE\_TYPE
  - [xtp\\_api\\_data\\_type.h, 118](#)
- XTP\_EXPIRE\_DATE\_TYPE
  - [xtp\\_api\\_data\\_type.h, 118](#)
- XTP\_FUND\_OPER\_STATUS
  - [xtp\\_api\\_data\\_type.h, 119](#)
- XTP\_FUND\_QUERY\_TYPE
  - [xtp\\_api\\_data\\_type.h, 119](#)
- XTP\_FUND\_TRANSFER\_TYPE
  - [xtp\\_api\\_data\\_type.h, 119](#)
- XTP\_LOG\_LEVEL
  - [xtp\\_api\\_data\\_type.h, 120](#)
- XTP\_MARKET\_TYPE
  - [xtp\\_api\\_data\\_type.h, 120](#)
- XTP\_OPT\_CALL\_OR\_PUT\_TYPE
  - [xtp\\_api\\_data\\_type.h, 120](#)
- XTP\_OPT\_COVERED\_OR\_UNCOVERED
  - [xtp\\_api\\_data\\_type.h, 121](#)
- XTP\_OPT\_EXERCISE\_TYPE\_TYPE
  - [xtp\\_api\\_data\\_type.h, 121](#)
- XTP\_OPT\_POSITION\_TYPE
  - [xtp\\_api\\_data\\_type.h, 121](#)
- XTP\_ORDER\_ACTION\_STATUS\_TYPE
  - [xtp\\_api\\_data\\_type.h, 122](#)
- XTP\_ORDER\_DETAIL\_TYPE
  - [xtp\\_api\\_data\\_type.h, 122](#)
- XTP\_ORDER\_STATUS\_TYPE
  - [xtp\\_api\\_data\\_type.h, 122](#)
- XTP\_ORDER\_SUBMIT\_STATUS\_TYPE
  - [xtp\\_api\\_data\\_type.h, 123](#)
- XTP\_POSITION\_DIRECTION\_TYPE
  - [xtp\\_api\\_data\\_type.h, 123](#)
- XTP\_POSITION\_SECURITY\_TYPE
  - [xtp\\_api\\_data\\_type.h, 123](#)
- XTP\_PRICE\_TYPE
  - [xtp\\_api\\_data\\_type.h, 124](#)
- XTP\_PROTOCOL\_TYPE
  - [xtp\\_api\\_data\\_type.h, 124](#)
- XTP\_QUALIFICATION\_TYPE
  - [xtp\\_api\\_data\\_type.h, 125](#)
- XTP\_SECURITY\_STATUS
  - [xtp\\_api\\_data\\_type.h, 125](#)
- XTP\_SECURITY\_TYPE
  - [xtp\\_api\\_data\\_type.h, 125](#)
- XTP\_SIDE\_STOCK\_REPAY\_STOCK
  - [xtp\\_api\\_data\\_type.h, 114](#)
- XTP\_SPLIT\_MERGE\_STATUS
  - [xtp\\_api\\_data\\_type.h, 126](#)
- XTP\_TBT\_TYPE
  - [xtp\\_api\\_data\\_type.h, 126](#)
- XTP\_TE\_RESUME\_TYPE
  - [xtp\\_api\\_data\\_type.h, 127](#)
- XTP\_TICKER\_TYPE
  - [xtp\\_api\\_data\\_type.h, 127](#)
- XTP\_UNDERLYING\_TYPE
  - [xtp\\_api\\_data\\_type.h, 127](#)
- XTPClientQueryCrdDebtStockReq, [45](#)
- XTPClientQueryCrdPositionStkInfo, [46](#)
- XTPClientQueryCrdPositionStockReq, [46](#)
- XTPClientQueryCrdSurplusStkReqInfo, [47](#)
- XTPClientQueryCrdSurplusStkRspInfo, [47](#)
- XTPCombLegStrategy, [48](#)
- XTPCrdCashRepayDebtInterestFeeRsp, [48](#)
- XTPCrdCashRepayInfo, [49](#)
- XTPCrdCashRepayRsp, [50](#)
- XTPCrdDebtInfo, [50](#)
- XTPCrdDebtStockInfo, [51](#)
- XTPCrdFundExtraInfo, [52](#)
- XTPCrdFundInfo, [52](#)
- XTPCrdPositionExtraInfo, [53](#)
- XTPCreditDebtExtendNotice, [53](#)
- XTPCreditDebtExtendReq, [54](#)
- XTPFundQueryReq, [54](#)
- XTPFundQueryRsp, [55](#)
- XTPFundTransferNotice, [55](#)
- XTPFundTransferReq, [56](#)
- XTPMarketDataOptionExData, [57](#)
- XTPMarketDataStockExData, [57](#)
- XTPMarketDataStruct, [59](#)
- XTPOptCombLegInfo, [60](#)
- XTPOptCombOrderInfo, [61](#)
  - [market, 62](#)
- XTPOptCombOrderInsertInfo, [62](#)
- XTPOptCombPlugin, [63](#)
- XTPOptCombTradeReport, [64](#)
- XTPOrderCancelInfo, [65](#)
- XTPOrderInfo, [65](#)
- XTPOrderInsertInfo, [66](#)
- XTPQueryAssetRsp, [67](#)
- XTPQueryCombineStrategyInfoRsp, [69](#)
- XTPQueryETFBBaseReq, [70](#)
- XTPQueryETFBBaseRsp, [70](#)

- XTPQueryETFComponentReq, 71
- XTPQueryETFComponentRsp, 71
- XTPQueryETFComponentRspV1, 72
- XTPQueryFundTransferLogReq, 73
- XTPQueryIPOQuotaRsp, 74
- XTPQueryIPOQuotaRspV1, 74
- XTPQueryIPOTickerRsp, 75
- XTPQueryOptCombExecPosReq, 75
- XTPQueryOptCombExecPosRsp, 76
- XTPQueryOptCombOrderByPageReq, 77
- XTPQueryOptCombOrderReq, 77
- XTPQueryOptCombPositionReq, 78
- XTPQueryOptCombPositionRsp, 78
- XTPQueryOptCombReportByExecIdReq, 79
- XTPQueryOptCombTraderByPageReq, 80
- XTPQueryOptCombTraderReq, 80
- XTPQueryOptExecInfoRsp, 81
- XTPQueryOptionAuctionInfoReq, 82
- XTPQueryOptionAuctionInfoRsp, 82
- XTPQueryOrderByPageReq, 84
- XTPQueryOrderReq, 84
- XTPQueryReportByExecIdReq, 85
- XTPQueryStkPositionReq, 85
- XTPQueryStkPositionRsp, 86
- XTPQueryStructuredFundInfoReq, 87
- XTPQueryTraderByPageReq, 88
- XTPQueryTraderReq, 88
- XTPQuoteFullInfo, 89
- XTPQuoteStaticInfo, 90
- XTPRspInfoStruct, 91
- XTPSpecificTickerStruct, 91
- XTPStructuredFundInfo, 92
- XTPTerminalType
  - xtp\_api\_data\_type.h, 128
- XTPTickByTickEntrust, 93
  - ord\_type, 93
  - order\_no, 93
  - qty, 93
  - seq, 93
  - side, 94
- XTPTickByTickStruct, 94
  - seq, 95
- XTPTickByTickTrade, 95
  - seq, 95
  - trade\_flag, 96
- XPTTickerPriceInfo, 96
- XTPTradeReport, 96
- XTPUserTerminalInfoReq, 98
- xoms\_api\_fund\_struct.h, 101
- xoms\_api\_struct.h, 102
- xquote\_api\_struct.h, 106
- xtp\_api\_data\_type.h, 107
  - ETF\_REPLACE\_TYPE, 115
  - XTP\_ACCOUNT\_TYPE, 115
  - XTP\_AUTO\_SPLIT\_TYPE, 115
  - XTP\_BUSINESS\_TYPE, 116
  - XTP\_CRD\_CR\_STATUS, 116
  - XTP\_DEBT\_EXTEND\_OPER\_STATUS, 118
  - XTP\_EXCHANGE\_TYPE, 118
  - XTP\_EXPIRE\_DATE\_TYPE, 118
  - XTP\_FUND\_OPER\_STATUS, 119
  - XTP\_FUND\_QUERY\_TYPE, 119
  - XTP\_FUND\_TRANSFER\_TYPE, 119
  - XTP\_LOG\_LEVEL, 120
  - XTP\_MARKET\_TYPE, 120
  - XTP\_OPT\_CALL\_OR\_PUT\_TYPE, 120
  - XTP\_OPT\_COVERED\_OR\_UNCOVERED, 121
  - XTP\_OPT\_EXERCISE\_TYPE\_TYPE, 121
  - XTP\_OPT\_POSITION\_TYPE, 121
  - XTP\_ORDER\_ACTION\_STATUS\_TYPE, 122
  - XTP\_ORDER\_DETAIL\_TYPE, 122
  - XTP\_ORDER\_STATUS\_TYPE, 122
  - XTP\_ORDER\_SUBMIT\_STATUS\_TYPE, 123
  - XTP\_POSITION\_DIRECTION\_TYPE, 123
  - XTP\_POSITION\_SECURITY\_TYPE, 123
  - XTP\_PRICE\_TYPE, 124
  - XTP\_PROTOCOL\_TYPE, 124
  - XTP\_QUALIFICATION\_TYPE, 125
  - XTP\_SECURITY\_STATUS, 125
  - XTP\_SECURITY\_TYPE, 125
  - XTP\_SIDE\_STOCK\_REPAY\_STOCK, 114
  - XTP\_SPLIT\_MERGE\_STATUS, 126
  - XTP\_TBT\_TYPE, 126
  - XTP\_TE\_RESUME\_TYPE, 127
  - XTP\_TICKER\_TYPE, 127
  - XTP\_UNDERLYING\_TYPE, 127
  - XTPTerminalType, 128
  - xtp\_api\_struct.h, 128
  - xtp\_api\_struct\_common.h, 129
  - xtp\_quote\_api.h, 129