

XTP极速交易系统TraderAPI

制作者 中泰证券股份有限公司

Contents

1	XTP 极速行情交易系统 Trader API 2.2.25.5	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	DemoTestTraderSpi类 参考	11
5.1.1	详细描述	12
5.1.2	成员函数说明	12
5.1.2.1	OnFundTransfer()	12
5.1.2.2	OnQueryAsset()	13
5.1.2.3	OnQueryETF()	13
5.1.2.4	OnQueryETFBasket()	14
5.1.2.5	OnQueryFundTransfer()	15
5.1.2.6	OnQueryIPOInfoList()	15
5.1.2.7	OnQueryIPOQuotaInfo()	16
5.1.2.8	OnQueryOptionAuctionInfo()	16
5.1.2.9	OnQueryOrder()	17
5.1.2.10	OnQueryPosition()	18

5.1.2.11	OnQueryStructuredFund()	18
5.1.2.12	OnQueryTrade()	19
5.2	OrderBookStruct结构体 参考	19
5.2.1	详细描述	20
5.3	TraderApi类 参考	20
5.3.1	详细描述	22
5.3.2	成员函数说明	22
5.3.2.1	CancelOptionCombinedOrder()	23
5.3.2.2	CancelOrder()	23
5.3.2.3	CreateTraderApi()	24
5.3.2.4	CreditCashRepay()	24
5.3.2.5	CreditCashRepayDebtInterestFee()	25
5.3.2.6	CreditExtendDebtDate()	25
5.3.2.7	CreditSellStockRepayDebtInterestFee()	26
5.3.2.8	FundTransfer()	26
5.3.2.9	GetAccountByXTPID()	27
5.3.2.10	GetApiLastError()	27
5.3.2.11	GetApiVersion()	28
5.3.2.12	GetClientIDByXTPID()	28
5.3.2.13	GetTradingDay()	28
5.3.2.14	InsertOptionCombinedOrder()	29
5.3.2.15	InsertOrder()	29
5.3.2.16	IsServerRestart()	30
5.3.2.17	Login()	30
5.3.2.18	Logout()	31
5.3.2.19	ModifyUserTerminalInfo()	31
5.3.2.20	QueryAsset()	32
5.3.2.21	QueryCreditAssetDebtInfo()	32
5.3.2.22	QueryCreditCashRepayInfo()	33
5.3.2.23	QueryCreditDebtInfo()	33

5.3.2.24	QueryCreditExcessStock()	34
5.3.2.25	QueryCreditExtendDebtDateOrders()	34
5.3.2.26	QueryCreditFundExtraInfo()	35
5.3.2.27	QueryCreditFundInfo()	35
5.3.2.28	QueryCreditPositionExtraInfo()	35
5.3.2.29	QueryCreditTickerAssignInfo()	36
5.3.2.30	QueryCreditTickerDebtInfo()	36
5.3.2.31	QueryETF()	38
5.3.2.32	QueryETFTickerBasket()	38
5.3.2.33	QueryFundTransfer()	39
5.3.2.34	QueryIPOInfoList()	39
5.3.2.35	QueryIPOQuotaInfo()	40
5.3.2.36	QueryMulCreditExcessStock()	40
5.3.2.37	QueryOptionAuctionInfo()	40
5.3.2.38	QueryOptionCombinedOrderByXTPID()	41
5.3.2.39	QueryOptionCombinedOrders()	41
5.3.2.40	QueryOptionCombinedOrdersByPage()	42
5.3.2.41	QueryOptionCombinedPosition()	43
5.3.2.42	QueryOptionCombinedStrategyInfo()	43
5.3.2.43	QueryOptionCombinedTrades()	44
5.3.2.44	QueryOptionCombinedTradesByPage()	44
5.3.2.45	QueryOptionCombinedTradesByXTPID()	45
5.3.2.46	QueryOptionCombinedUnfinishedOrders()	45
5.3.2.47	QueryOrderByXTPID()	46
5.3.2.48	QueryOrders()	46
5.3.2.49	QueryOrdersByPage()	47
5.3.2.50	QueryPosition()	47
5.3.2.51	QueryStructuredFund()	48
5.3.2.52	QueryTrades()	49
5.3.2.53	QueryTradesByPage()	49

5.3.2.54	QueryTradesByXTPID()	50
5.3.2.55	QueryUnfinishedOrders()	50
5.3.2.56	RegisterSpi()	51
5.3.2.57	Release()	51
5.3.2.58	SetHeartBeatInterval()	51
5.3.2.59	SetSoftwareKey()	51
5.3.2.60	SetSoftwareVersion()	52
5.3.2.61	SubscribePublicTopic()	52
5.4	TraderSpi类 参考	53
5.4.1	详细描述	54
5.4.2	成员函数说明	55
5.4.2.1	OnCancelOptionCombinedOrderError()	55
5.4.2.2	OnCancelOrderError()	55
5.4.2.3	OnCreditCashRepay()	56
5.4.2.4	OnCreditCashRepayDebtInterestFee()	56
5.4.2.5	OnCreditExtendDebtDate()	57
5.4.2.6	OnDisconnected()	57
5.4.2.7	OnError()	58
5.4.2.8	OnFundTransfer()	58
5.4.2.9	OnOptionCombinedOrderEvent()	59
5.4.2.10	OnOptionCombinedTradeEvent()	59
5.4.2.11	OnOrderEvent()	60
5.4.2.12	OnQueryAsset()	60
5.4.2.13	OnQueryCreditAssetDebtInfo()	61
5.4.2.14	OnQueryCreditCashRepayInfo()	61
5.4.2.15	OnQueryCreditDebtInfo()	62
5.4.2.16	OnQueryCreditExcessStock()	63
5.4.2.17	OnQueryCreditExtendDebtDateOrders()	63
5.4.2.18	OnQueryCreditFundExtraInfo()	64
5.4.2.19	OnQueryCreditFundInfo()	64

5.4.2.20	OnQueryCreditPositionExtraInfo()	65
5.4.2.21	OnQueryCreditTickerAssignInfo()	65
5.4.2.22	OnQueryCreditTickerDebtInfo()	66
5.4.2.23	OnQueryETF()	67
5.4.2.24	OnQueryETFBasket()	67
5.4.2.25	OnQueryFundTransfer()	68
5.4.2.26	OnQueryIPOInfoList()	68
5.4.2.27	OnQueryIPOQuotaInfo()	69
5.4.2.28	OnQueryMulCreditExcessStock()	70
5.4.2.29	OnQueryOptionAuctionInfo()	70
5.4.2.30	OnQueryOptionCombinedOrders()	71
5.4.2.31	OnQueryOptionCombinedOrdersByPage()	71
5.4.2.32	OnQueryOptionCombinedPosition()	72
5.4.2.33	OnQueryOptionCombinedStrategyInfo()	73
5.4.2.34	OnQueryOptionCombinedTrades()	73
5.4.2.35	OnQueryOptionCombinedTradesByPage()	74
5.4.2.36	OnQueryOrder()	74
5.4.2.37	OnQueryOrderByPage()	75
5.4.2.38	OnQueryPosition()	76
5.4.2.39	OnQueryStructuredFund()	76
5.4.2.40	OnQueryTrade()	77
5.4.2.41	OnQueryTradeByPage()	78
5.4.2.42	OnTradeEvent()	78
5.5	XTPClientQueryCrdDebtStockReq结构体 参考	79
5.5.1	详细描述	79
5.6	XTPClientQueryCrdPositionStkInfo结构体 参考	79
5.6.1	详细描述	80
5.7	XTPClientQueryCrdPositionStockReq结构体 参考	80
5.7.1	详细描述	80
5.8	XTPClientQueryCrdSurplusStkReqInfo结构体 参考	81

5.8.1 详细描述	81
5.9 XTPClientQueryCrdSurplusStkRspInfo结构体 参考	81
5.9.1 详细描述	81
5.10 XTPCombLegStrategy结构体 参考	82
5.10.1 详细描述	82
5.11 XTPCrdCashRepayDebtInterestFeeRsp结构体 参考	82
5.11.1 详细描述	83
5.12 XTPCrdCashRepayInfo结构体 参考	83
5.12.1 详细描述	83
5.13 XTPCrdCashRepayRsp结构体 参考	83
5.13.1 详细描述	84
5.14 XTPCrdDebtInfo结构体 参考	84
5.14.1 详细描述	85
5.15 XTPCrdDebtStockInfo结构体 参考	85
5.15.1 详细描述	85
5.16 XTPCrdFundExtraInfo结构体 参考	86
5.16.1 详细描述	86
5.17 XTPCrdFundInfo结构体 参考	86
5.17.1 详细描述	87
5.18 XTPCrdPositionExtraInfo结构体 参考	87
5.18.1 详细描述	87
5.19 XTPCreditDebtExtendNotice结构体 参考	87
5.19.1 详细描述	88
5.20 XTPCreditDebtExtendReq结构体 参考	88
5.20.1 详细描述	88
5.21 XTPFundTransferNotice结构体 参考	88
5.21.1 详细描述	89
5.22 XTPFundTransferReq结构体 参考	89
5.22.1 详细描述	89
5.23 XTPMarketDataOptionExData结构体 参考	90

5.23.1 详细描述	90
5.24 XTPMarketDataStockExData结构体 参考	90
5.24.1 详细描述	92
5.25 XTPMarketDataStruct结构体 参考	92
5.25.1 详细描述	93
5.26 XTPOptCombLegInfo结构体 参考	93
5.26.1 详细描述	94
5.27 XTPOptCombOrderInfo结构体 参考	94
5.27.1 详细描述	95
5.27.2 结构体成员变量说明	95
5.27.2.1 market	95
5.28 XTPOptCombOrderInsertInfo结构体 参考	95
5.28.1 详细描述	96
5.29 XTPOptCombPlugin结构体 参考	96
5.29.1 详细描述	96
5.30 XTPOptCombTradeReport结构体 参考	97
5.30.1 详细描述	97
5.31 XTPOrderCancelInfo结构体 参考	98
5.31.1 详细描述	98
5.32 XTPOrderInfo结构体 参考	98
5.32.1 详细描述	99
5.33 XTPOrderInsertInfo结构体 参考	99
5.33.1 详细描述	100
5.34 XTPQueryAssetRsp结构体 参考	100
5.34.1 详细描述	102
5.35 XTPQueryCombineStrategyInfoRsp结构体 参考	102
5.35.1 详细描述	102
5.36 XTPQueryETFBaseReq结构体 参考	102
5.36.1 详细描述	103
5.37 XTPQueryETFBaseRsp结构体 参考	103

5.37.1 详细描述	104
5.38 XTPQueryETFComponentReq结构体 参考	104
5.38.1 详细描述	104
5.39 XTPQueryETFComponentRsp结构体 参考	104
5.39.1 详细描述	105
5.40 XTPQueryETFComponentRspV1结构体 参考	105
5.40.1 详细描述	106
5.41 XTPQueryFundTransferLogReq结构体 参考	106
5.41.1 详细描述	106
5.42 XTPQueryIPOQuotaRsp结构体 参考	107
5.42.1 详细描述	107
5.43 XTPQueryIPOQuotaRspV1结构体 参考	107
5.43.1 详细描述	107
5.44 XTPQueryIPOTickerRsp结构体 参考	108
5.44.1 详细描述	108
5.45 XTPQueryOptCombOrderByPageReq结构体 参考	108
5.45.1 详细描述	109
5.46 XTPQueryOptCombOrderReq结构体 参考	109
5.46.1 详细描述	109
5.47 XTPQueryOptCombPositionReq结构体 参考	109
5.47.1 详细描述	110
5.48 XTPQueryOptCombPositionRsp结构体 参考	110
5.48.1 详细描述	110
5.49 XTPQueryOptCombReportByExecIdReq结构体 参考	110
5.49.1 详细描述	111
5.50 XTPQueryOptCombTraderByPageReq结构体 参考	111
5.50.1 详细描述	111
5.51 XTPQueryOptCombTraderReq结构体 参考	111
5.51.1 详细描述	112
5.52 XTPQueryOptionAuctionInfoReq结构体 参考	112

5.52.1 详细描述	112
5.53 XTPQueryOptionAuctionInfoRsp结构体 参考	112
5.53.1 详细描述	114
5.54 XTPQueryOrderByPageReq结构体 参考	114
5.54.1 详细描述	115
5.55 XTPQueryOrderReq结构体 参考	115
5.55.1 详细描述	115
5.56 XTPQueryReportByExecIdReq结构体 参考	115
5.56.1 详细描述	116
5.57 XTPQueryStkPositionReq结构体 参考	116
5.57.1 详细描述	116
5.58 XTPQueryStkPositionRsp结构体 参考	116
5.58.1 详细描述	117
5.59 XTPQueryStructuredFundInfoReq结构体 参考	117
5.59.1 详细描述	118
5.60 XTPQueryTraderByPageReq结构体 参考	118
5.60.1 详细描述	118
5.61 XTPQueryTraderReq结构体 参考	118
5.61.1 详细描述	119
5.62 XTPQuoteFullInfo结构体 参考	119
5.62.1 详细描述	120
5.63 XTPQuoteStaticInfo结构体 参考	120
5.63.1 详细描述	121
5.64 XTPRspInfoStruct结构体 参考	121
5.64.1 详细描述	122
5.65 XTPSpecificTickerStruct结构体 参考	122
5.65.1 详细描述	122
5.66 XTPStructuredFundInfo结构体 参考	122
5.66.1 详细描述	123
5.67 XTPTickByTickEntrust结构体 参考	123
5.67.1 详细描述	124
5.68 XTPTickByTickStruct结构体 参考	124
5.68.1 详细描述	124
5.69 XTPTickByTickTrade结构体 参考	124
5.69.1 详细描述	125
5.69.2 结构体成员变量说明	125
5.69.2.1 trade_flag	125
5.70 XTPTickerPricelInfo结构体 参考	125
5.70.1 详细描述	126
5.71 XTPTradeReport结构体 参考	126
5.71.1 详细描述	127
5.72 XTPUserTerminalInfoReq结构体 参考	127
5.72.1 详细描述	128

6 文件说明	129
6.1 demo_test_trade_api.cpp 文件参考	129
6.1.1 详细描述	129
6.1.2 函数说明	129
6.1.2.1 main()	130
6.2 demo_test_trade_spi.h 文件参考	131
6.2.1 详细描述	131
6.3 xoms_api_fund_struct.h 文件参考	132
6.3.1 详细描述	132
6.4 xoms_api_struct.h 文件参考	132
6.4.1 详细描述	136
6.5 xquote_api_struct.h 文件参考	136
6.5.1 详细描述	137
6.6 xtp_api_data_type.h 文件参考	137
6.6.1 详细描述	144
6.6.2 宏定义说明	144
6.6.2.1 XTP_SIDE_STOCK_REPAY_STOCK	144
6.6.3 枚举类型说明	144
6.6.3.1 ETF_REPLACE_TYPE	144
6.6.3.2 XTP_ACCOUNT_TYPE	145
6.6.3.3 XTP_AUTO_SPLIT_TYPE	145
6.6.3.4 XTP_BUSINESS_TYPE	146
6.6.3.5 XTP_CRD_CR_STATUS	146
6.6.3.6 XTP_DEBT_EXTEND_OPER_STATUS	147
6.6.3.7 XTP_EXCHANGE_TYPE	147
6.6.3.8 XTP_EXPIRE_DATE_TYPE	147
6.6.3.9 XTP_FUND_OPER_STATUS	147
6.6.3.10 XTP_FUND_TRANSFER_TYPE	148
6.6.3.11 XTP_LOG_LEVEL	148
6.6.3.12 XTP_MARKET_TYPE	149

6.6.3.13	XTP_OPT_CALL_OR_PUT_TYPE	149
6.6.3.14	XTP_OPT_COVERED_OR_UNCOVERED	149
6.6.3.15	XTP_OPT_EXERCISE_TYPE_TYPE	149
6.6.3.16	XTP_OPT_POSITION_TYPE	150
6.6.3.17	XTP_ORDER_ACTION_STATUS_TYPE	150
6.6.3.18	XTP_ORDER_DETAIL_TYPE	150
6.6.3.19	XTP_ORDER_STATUS_TYPE	151
6.6.3.20	XTP_ORDER_SUBMIT_STATUS_TYPE	151
6.6.3.21	XTP_POSITION_DIRECTION_TYPE	151
6.6.3.22	XTP_PRICE_TYPE	152
6.6.3.23	XTP_PROTOCOL_TYPE	152
6.6.3.24	XTP_QUALIFICATION_TYPE	153
6.6.3.25	XTP_SECURITY_TYPE	153
6.6.3.26	XTP_SPLIT_MERGE_STATUS	154
6.6.3.27	XTP_TBT_TYPE	154
6.6.3.28	XTP_TE_RESUME_TYPE	154
6.6.3.29	XTP_TICKER_TYPE	155
6.6.3.30	XTP_UNDERLYING_TYPE	155
6.6.3.31	XTPTerminalType	155
6.7	xtp_api_struct.h 文件参考	156
6.7.1	详细描述	156
6.8	xtp_api_struct_common.h 文件参考	156
6.8.1	详细描述	156
6.9	xtp_trader_api.h 文件参考	157
6.9.1	详细描述	157
索引		159

Chapter 1

XTP 极速行情交易系统 Trader API 2.2.25.5

本项目是XTP项目中的交易类接口

(1) XTP的交易接口和响应类 [xtp_trader_api.h](#)

(2) 程序化交易接口测试Demo [demo_test_trade_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	19
TraderApi	20
TraderSpi	53
DemoTestTraderSpi	11
XTPClientQueryCrdDebtStockReq	79
XTPClientQueryCrdPositionStkInfo	79
XTPClientQueryCrdPositionStockReq	80
XTPClientQueryCrdSurplusStkReqInfo	81
XTPClientQueryCrdSurplusStkRspInfo	81
XTPCombLegStrategy	82
XTPCrdCashRepayDebtInterestFeeRsp	82
XTPCrdCashRepayInfo	83
XTPCrdCashRepayRsp	83
XTPCrdDebtInfo	84
XTPCrdDebtStockInfo	85
XTPCrdFundExtraInfo	86
XTPCrdFundInfo	86
XTPCrdPositionExtraInfo	87
XTPCreditDebtExtendNotice	87
XTPCreditDebtExtendReq	88
XTPFundTransferNotice	88
XTPFundTransferReq	89
XTPMarketDataOptionExData	90
XTPMarketDataStockExData	90
XTPMarketDataStruct	92
XTPOptCombLegInfo	93
XTPOptCombOrderInfo	94
XTPOptCombOrderInsertInfo	95
XTPOptCombPlugin	96
XTPOptCombTradeReport	97
XTPOrderCancelInfo	98
XTPOrderInfo	98
XTPOrderInsertInfo	99
XTPQueryAssetRsp	100
XTPQueryCombineStrategyInfoRsp	102

XTPQueryETFBaseReq	102
XTPQueryETFBaseRsp	103
XTPQueryETFComponentReq	104
XTPQueryETFComponentRsp	104
XTPQueryETFComponentRspV1	105
XTPQueryFundTransferLogReq	106
XTPQueryIPOQuotaRsp	107
XTPQueryIPOQuotaRspV1	107
XTPQueryIPOTickerRsp	108
XTPQueryOptCombOrderByPageReq	108
XTPQueryOptCombOrderReq	109
XTPQueryOptCombPositionReq	109
XTPQueryOptCombPositionRsp	110
XTPQueryOptCombReportByExeclDReq	110
XTPQueryOptCombTraderByPageReq	111
XTPQueryOptCombTraderReq	111
XTPQueryOptionAuctionInfoReq	112
XTPQueryOptionAuctionInfoRsp	112
XTPQueryOrderByPageReq	114
XTPQueryOrderReq	115
XTPQueryReportByExeclDReq	115
XTPQueryStkPositionReq	116
XTPQueryStkPositionRsp	116
XTPQueryStructuredFundInfoReq	117
XTPQueryTraderByPageReq	118
XTPQueryTraderReq	118
XTPQuoteFullInfo	119
XTPQuoteStaticInfo	120
XTPRspInfoStruct	121
XTPSpecificTickerStruct	122
XTPStructuredFundInfo	122
XTPTickByTickEntrust	123
XTPTickByTickStruct	124
XTPTickByTickTrade	124
XTPTickerPricInfo	125
XTPTradeReport	126
XTPUserTerminalInfoReq	127

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：

DemoTestTraderSpi	
Demo自定义交易接口响应类	11
OrderBookStruct	
定单簿	19
TraderApi	
交易接口类	20
TraderSpi	
交易接口响应类	53
XTPClientQueryCrdDebtStockReq	
融资融券指定证券上的负债未还数量请求结构体	79
XTPClientQueryCrdPositionStkInfo	
融券头寸证券信息	79
XTPClientQueryCrdPositionStockReq	
融券头寸证券查询请求结构体	80
XTPClientQueryCrdSurplusStkReqInfo	
信用业务余券查询请求结构体	81
XTPClientQueryCrdSurplusStkRsplInfo	
信用业务余券信息	81
XTPCombLegStrategy	
期权组合策略的成分合约信息	82
XTPCrdCashRepayDebtInterestFeeRsp	
融资融券现金还息费响应信息	82
XTPCrdCashRepayInfo	
单条融资融券直接还款记录信息	83
XTPCrdCashRepayRsp	
融资融券直接还款响应信息	83
XTPCrdDebtInfo	
单条融资融券负债记录信息	84
XTPCrdDebtStockInfo	
融资融券指定证券的融券负债相关信息	85
XTPCrdFundExtrlInfo	
融资融券帐户附加信息	86
XTPCrdFundInfo	
融资融券特有帐户数据	86
XTPCrdPositionExtrlInfo	
融资融券帐户持仓附加信息	87

XTPCreditDebtExtendNotice	
用户展期请求的通知	87
XTPCreditDebtExtendReq	
用户展期请求	88
XTPFundTransferNotice	
资金内转流水通知	88
XTPFundTransferReq	
用户资金请求	89
XTPMarketDataOptionExData	
期权额外数据	90
XTPMarketDataStockExData	
股票、基金、债券等额外数据	90
XTPMarketDataStruct	
行情	92
XTPOptCombLegInfo	
组合策略腿合约信息结构体	93
XTPOptCombOrderInfo	
期权组合策略报单响应结构体	94
XTPOptCombOrderInsertInfo	
期权组合策略新订单请求	95
XTPOptCombPlugin	
期权组合策略报单附加信息结构体	96
XTPOptCombTradeReport	
期权组合策略报单成交结构体	97
XTPOrderCancelInfo	
撤单失败响应消息	98
XTPOrderInfo	
报单响应结构体	98
XTPOrderInsertInfo	
新订单请求	99
XTPQueryAssetRsp	
账户资金查询响应结构体	100
XTPQueryCombineStrategyInfoRsp	
查询期权组合策略信息的响应	102
XTPQueryETFBaseReq	
查询股票ETF合约基本情况-响应结构体	103
XTPQueryETFBaseRsp	
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	104
XTPQueryETFComponentReq	
查询股票ETF成分股信息-响应结构体	104
XTPQueryETFComponentRsp	
查询股票ETF成分股信息-响应结构体, 旧版本。	105
XTPQueryETFComponentRspV1	
查询股票ETF成分股信息-响应结构体, 旧版本。	105
XTPQueryFundTransferLogReq	
资金内转流水查询请求与响应	106
XTPQueryIPOQuotaRsp	
查询用户申购额度-包含创业板额度	107
XTPQueryIPOQuotaRspV1	
查询用户申购额度-旧版	107
XTPQueryIPOTickerRsp	
查询当日可申购新股信息	108
XTPQueryOptCombOrderByPageReq	
查询期权组合策略订单请求-分页查询	108
XTPQueryOptCombOrderReq	
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	109
XTPQueryOptCombPositionReq	
查询期权组合策略持仓情况请求结构体	109

XTPQueryOptCombPositionRsp	110
查询期权组合策略持仓信息的响应	
XTPQueryOptCombReportByExeclIdReq	110
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	111
查询期权组合策略成交回报请求-分页查询	
XTPQueryOptCombTraderReq	111
查询期权组合策略成交回报请求-查询条件	
XTPQueryOptionAuctionInfoReq	112
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	112
查询期权竞价交易业务参考信息	
XTPQueryOrderByPageReq	114
查询订单请求-分页查询	
XTPQueryOrderReq	115
报单查询 // 报单查询请求-条件查询	
XTPQueryReportByExeclIdReq	115
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	116
查询股票持仓情况请求结构体	
XTPQueryStkPositionRsp	116
查询股票持仓情况	
XTPQueryStructuredFundInfoReq	117
查询分级基金信息结构体	
XTPQueryTraderByPageReq	118
查询成交回报请求-分页查询	
XTPQueryTraderReq	118
查询成交回报请求-查询条件	
XTPQuoteFullInfo	119
股票行情全量静态信息	
XTPQuoteStaticInfo	120
股票行情静态信息	
XTPRspInfoStruct	121
响应信息	
XTPSpecificTickerStruct	122
指定的合约	
XTPStructuredFundInfo	122
查询分级基金信息响应结构体	
XTPTickByTickEntrust	123
逐笔委托(仅适用深交所)	
XTPTickByTickStruct	124
逐笔数据信息	
XTPTickByTickTrade	124
逐笔成交	
XPTickerPriceInfo	125
供查询的最新信息	
XTPTradeReport	126
报单成交结构体	
XTPUserTerminalInfoReq	127
申报用户的ip和mac等信息，仅限授权用户使用	

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_trade_api.cpp	定义控制台测试应用程序的入口点	129
demo_test_trade_spi.h	Demo自定义客户端交易响应接口类	131
xoms_api_fund_struct.h	定义资金划拨相关结构体类型	132
xoms_api_struct.h	定义交易类相关数据结构	132
xquote_api_struct.h	定义行情类相关数据结构	136
xtp_api_data_type.h	定义兼容数据基本类型	137
xtp_api_struct.h	定义业务数据结构	156
xtp_api_struct_common.h	定义业务公共数据结构	156
xtp_trader_api.h	定义客户端交易接口	157

Chapter 5

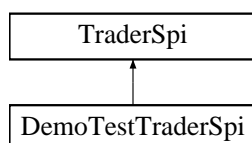
结构体说明

5.1 DemoTestTraderSpi类 参考

Demo自定义交易接口响应类

```
#include <demo_test_trade_spi.h>
```

类 DemoTestTraderSpi 继承关系图:



Public 成员函数

- virtual void [OnDisconnected](#) (uint64_t session_id, int reason)
当客户端某个连接与交易后台通信连接断开
- virtual void [OnError](#) (XTPRI *error_info)
错误应答
- virtual void [OnOrderEvent](#) (XTPOrderInfo *order_info, XTPRI *error_info, uint64_t session_id)
报单通知
- virtual void [OnTradeEvent](#) (XTPTradeReport *trade_info, uint64_t session_id)
成交通知
- virtual void [OnQueryOrder](#) (XTPQueryOrderRsp *order_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryTrade](#) (XTPQueryTradeRsp *trade_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryPosition](#) (XTPQueryStkPositionRsp *position, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryAsset](#) (XTPQueryAssetRsp *asset, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryStructuredFund](#) (XTPStructuredFundInfo *fund_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryFundTransfer](#) (XTPFundTransferNotice *fund_transfer_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)

- virtual void [OnFundTransfer](#) (XTPFundTransferNotice *fund_transfer_info, [XTPRI](#) *error_info, uint64_t session_id)
- virtual void [OnQueryETF](#) (XTPQueryETFBaseRsp *etf_info, [XTPRI](#) *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryETFBasket](#) (XTPQueryETFComponentRsp *etf_component_info, [XTPRI](#) *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryIPOInfoList](#) (XTPQueryIPOTickerRsp *ipo_info, [XTPRI](#) *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryIPOQuotaInfo](#) (XTPQueryIPOQuotaRsp *quota_info, [XTPRI](#) *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryOptionAuctionInfo](#) (XTPQueryOptionAuctionInfoRsp *option_info, [XTPRI](#) *error_info, int request_id, bool is_last, uint64_t session_id)

5.1.1 详细描述

Demo自定义交易接口响应类

作者

中泰证券股份有限公司

日期

十月 2015

5.1.2 成员函数说明

5.1.2.1 OnFundTransfer()

```
virtual void OnFundTransfer (
    XTPFundTransferNotice * fund_transfer_info,
    XTPRI * error_info,
    uint64_t session_id ) [virtual]
```

资金划拨通知

参数

<i>fund_transfer_info</i>	资金划拨通知的具体信息，用户可以通过fund_transfer_info.serial_id来管理订单，通过GetClientIDByXTPID() == client_id来过滤自己的订单。
<i>error_info</i>	资金划拨订单被拒绝或者发生错误时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。当资金划拨方向为一号两中心节点之间划拨，且error_info.error_id=11000384时，error_info.error_msg为结点中可用于划拨的资金（以整数为准），用户需进行stringToInt的转化，可据此填写合适的资金，再次发起划拨请求
<i>session_id</i>	资金账户对应的session_id，登录时得到

备注

当资金划拨订单有状态变化的时候，会被调用，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。所有登录了此用户的客户端都将收到此用户的资金划拨通知。

重载 [TraderSpi](#) .

5.1.2.2 OnQueryAsset()

```
virtual void OnQueryAsset (
    XTPQueryAssetRsp * asset,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询资金账户响应，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

参数

<i>asset</i>	查询到的资金账户情况
<i>error_info</i>	查询资金账户发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

重载 [TraderSpi](#) .

5.1.2.3 OnQueryETF()

```
virtual void OnQueryETF (
    XTPQueryETFBaseRsp * etf_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询ETF清单文件的响应，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

参数

<i>etf_info</i>	查询到的ETF清单文件情况
<i>error_info</i>	查询ETF清单文件发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

重载 [TraderSpi](#) .

5.1.2.4 OnQueryETFBasket()

```
virtual void OnQueryETFBasket (
    XTPQueryETFComponentRsp * etf_component_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询ETF股票篮的响应，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

参数

<i>etf_component_info</i>	查询到的ETF合约的相关成分股信息
<i>error_info</i>	查询ETF股票篮发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

重载 [TraderSpi](#) .

5.1.2.5 OnQueryFundTransfer()

```
virtual void OnQueryFundTransfer (
    XTPFundTransferNotice * fund_transfer_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询资金划拨订单响应，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

参数

<i>fund_transfer_info</i>	查询到的资金账户情况
<i>error_info</i>	查询资金账户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>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的session_id，登录时得到

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

重载 [TraderSpi](#) .

5.1.2.6 OnQueryIPOInfoList()

```
virtual void OnQueryIPOInfoList (
    XTPQueryIPOTickerRsp * ipo_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询今日新股申购信息列表的响应，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

参数

<i>ipo_info</i>	查询到的今日新股申购的一只股票信息
<i>error_info</i>	查询今日新股申购信息列表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>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的session_id，登录时得到

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

重载 [TraderSpi](#) .

5.1.2.7 OnQueryIPOQuotaInfo()

```
virtual void OnQueryIPOQuotaInfo (
    XTPQueryIPOQuotaRsp * quota_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询用户新股申购额度信息的响应，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

参数

<i>quota_info</i>	查询到的用户某个市场的今日新股申购额度信息
<i>error_info</i>	查查询用户新股申购额度信息发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

重载 [TraderSpi](#) .

5.1.2.8 OnQueryOptionAuctionInfo()

```
virtual void OnQueryOptionAuctionInfo (
    XTPQueryOptionAuctionInfoRsp * option_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询期权合约的响应，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

参数

<i>option_info</i>	查询到的期权合约情况
<i>error_info</i>	查询期权合约发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

重载 [TraderSpi](#) .

5.1.2.9 OnQueryOrder()

```
virtual void OnQueryOrder (
    XTPQueryOrderRsp * order_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询报单响应

参数

<i>order_info</i>	查询到的一个报单
<i>error_info</i>	查询报单时发生错误时，返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

由于支持分时段查询，一个查询请求可能对应多个响应，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

重载 [TraderSpi](#) .

5.1.2.10 OnQueryPosition()

```
virtual void OnQueryPosition (
    XTPQueryStkPositionRsp * position,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询投资者持仓响应

参数

<i>position</i>	查询到的一只股票的持仓情况
<i>error_info</i>	查询账户持仓发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

由于用户可能持有多个股票，一个查询请求可能对应多个响应，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

重载 [TraderSpi](#) .

5.1.2.11 OnQueryStructuredFund()

```
virtual void OnQueryStructuredFund (
    XTPStructuredFundInfo * fund_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询分级基金信息响应，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

参数

<i>fund_info</i>	查询到的分级基金情况
<i>error_info</i>	查询分级基金发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

制作者 Doxygen

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

重载 [TraderSpi](#) .

5.1.2.12 OnQueryTrade()

```
virtual void OnQueryTrade (
    XTPQueryTradeRsp * trade_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [virtual]
```

请求查询成交响应

参数

<i>trade_info</i>	查询到的一个成交回报
<i>error_info</i>	查询成交回报发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

由于支持分时段查询，一个查询请求可能对应多个响应，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

重载 [TraderSpi](#) .

该类的文档由以下文件生成:

- [demo_test_trade_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 TraderApi类 参考

交易接口类

```
#include <xtp_trader_api.h>
```

Public 成员函数

- virtual void [Release](#) ()=0
- virtual const char * [GetTradingDay](#) ()=0
- virtual void [RegisterSpi](#) (TraderSpi *spi)=0
- virtual XTPRI * [GetApiLastError](#) ()=0
- virtual const char * [GetApiVersion](#) ()=0
- virtual uint8_t [GetClientIDByXTPID](#) (uint64_t order_xtp_id)=0
- virtual const char * [GetAccountByXTPID](#) (uint64_t order_xtp_id)=0
- virtual void [SubscribePublicTopic](#) (XTP_TE_RESUME_TYPE resume_type)=0
- virtual void [SetSoftwareVersion](#) (const char *version)=0
- virtual void [SetSoftwareKey](#) (const char *key)=0
- virtual void [SetHeartBeatInterval](#) (uint32_t interval)=0
- virtual uint64_t [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](#) (uint64_t session_id)=0
- virtual bool [IsServerRestart](#) (uint64_t session_id)=0
- virtual int [ModifyUserTerminalInfo](#) (const XTPUserTerminalInfoReq *info, uint64_t session_id)=0
- virtual uint64_t [InsertOrder](#) (XTPOrderInsertInfo *order, uint64_t session_id)=0
- virtual uint64_t [CancelOrder](#) (const uint64_t order_xtp_id, uint64_t session_id)=0
- virtual int [QueryOrderByXTPID](#) (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int [QueryOrders](#) (const XTPQueryOrderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryUnfinishedOrders](#) (uint64_t session_id, int request_id)=0
- virtual int [QueryOrdersByPage](#) (const XTPQueryOrderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryTradesByXTPID](#) (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int [QueryTrades](#) (XTPQueryTraderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryTradesByPage](#) (const XTPQueryTraderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryPosition](#) (const char *ticker, uint64_t session_id, int request_id, XTP_MARKET_TYPE market=XTP_MKT_INIT)=0
- virtual int [QueryAsset](#) (uint64_t session_id, int request_id)=0
- virtual int [QueryStructuredFund](#) (XTPQueryStructuredFundInfoReq *query_param, uint64_t session_id, int request_id)=0
- virtual uint64_t [FundTransfer](#) (XTPFundTransferReq *fund_transfer, uint64_t session_id)=0
- virtual int [QueryFundTransfer](#) (XTPQueryFundTransferLogReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryETF](#) (XTPQueryETFBaseReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryETFTickerBasket](#) (XTPQueryETFComponentReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryIPOInfoList](#) (uint64_t session_id, int request_id)=0
- virtual int [QueryIPOQuotaInfo](#) (uint64_t session_id, int request_id)=0
- virtual int [QueryOptionAuctionInfo](#) (XTPQueryOptionAuctionInfoReq *query_param, uint64_t session_id, int request_id)=0
- virtual uint64_t [CreditCashRepay](#) (double amount, uint64_t session_id)=0
- virtual uint64_t [CreditCashRepayDebtInterestFee](#) (const char *debt_id, double amount, uint64_t session_id)=0
- virtual uint64_t [CreditSellStockRepayDebtInterestFee](#) (XTPOrderInsertInfo *order, const char *debt_id, uint64_t session_id)=0
- virtual int [QueryCreditCashRepayInfo](#) (uint64_t session_id, int request_id)=0
- virtual int [QueryCreditFundInfo](#) (uint64_t session_id, int request_id)=0
- virtual int [QueryCreditDebtInfo](#) (uint64_t session_id, int request_id)=0
- virtual int [QueryCreditTickerDebtInfo](#) (XTPClientQueryCrddDebtStockReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryCreditAssetDebtInfo](#) (uint64_t session_id, int request_id)=0

- virtual int QueryCreditTickerAssignInfo (XTPClientQueryCrdPositionStockReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryCreditExcessStock (XTPClientQueryCrdSurplusStkReqInfo *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryMulCreditExcessStock (XTPClientQueryCrdSurplusStkReqInfo *query_param, uint64_t session_id, int request_id)=0
- virtual uint64_t CreditExtendDebtDate (XTPCreditDebtExtendReq *debt_extend, uint64_t session_id)=0
- virtual int QueryCreditExtendDebtDateOrders (uint64_t xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryCreditFundExtraInfo (uint64_t session_id, int request_id)=0
- virtual int QueryCreditPositionExtraInfo (XTPClientQueryCrdPositionStockReq *query_param, uint64_t session_id, int request_id)=0
- virtual uint64_t InsertOptionCombinedOrder (XTPOptCombOrderInsertInfo *order, uint64_t session_id)=0
- virtual uint64_t CancelOptionCombinedOrder (const uint64_t order_xtp_id, uint64_t session_id)=0
- virtual int QueryOptionCombinedUnfinishedOrders (uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedOrderByXTPID (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedOrders (const XTPQueryOptCombOrderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedOrdersByPage (const XTPQueryOptCombOrderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedTradesByXTPID (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedTrades (const XTPQueryOptCombTraderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedTradesByPage (const XTPQueryOptCombTraderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedPosition (const XTPQueryOptCombPositionReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedStrategyInfo (uint64_t session_id, int request_id)=0

静态 Public 成员函数

- static TraderApi * CreateTraderApi (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

交易接口类

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 CancelOptionCombinedOrder()

```
virtual uint64_t CancelOptionCombinedOrder (
    const uint64_t order_xtp_id,
    uint64_t session_id ) [pure virtual]
```

期权组合策略报单撤单请求

返回

撤单在XTP系统中的ID,如果为'0'表示撤单发送失败,此时用户可以调用GetApiLastError()来获取错误代码,非"0"表示撤单发送成功,用户需要记录下返回的order_cancel_xtp_id,它保证一个交易日内唯一,不同的交易日不保证唯一性

参数

<i>order_xtp_id</i>	需要撤销的期权组合策略委托单在XTP系统中的ID
<i>session_id</i>	资金账户对应的session_id,登录时得到

备注

如果撤单成功,会在报单响应函数OnOptionCombinedOrderEvent()里返回原单部撤或者全撤的消息,如果不成功,会在OnCancelOrderError()响应函数中返回错误原因

5.3.2.2 CancelOrder()

```
virtual uint64_t CancelOrder (
    const uint64_t order_xtp_id,
    uint64_t session_id ) [pure virtual]
```

报单操作请求

返回

撤单在XTP系统中的ID,如果为'0'表示撤单发送失败,此时用户可以调用GetApiLastError()来获取错误代码,非"0"表示撤单发送成功,用户需要记录下返回的order_cancel_xtp_id,它保证一个交易日内唯一,不同的交易日不保证唯一性

参数

<i>order_xtp_id</i>	需要撤销的委托单在XTP系统中的ID
<i>session_id</i>	资金账户对应的session_id,登录时得到

备注

如果撤单成功，会在报单响应函数OnOrderEvent()里返回原单部撤或者全撤的消息，如果不成功，会在OnCancelOrderError()响应函数中返回错误原因

5.3.2.3 CreateTraderApi()

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

创建TraderApi

参数

client_id	(必须输入) 客户端id，用于区分同一用户的不同客户端，由用户自定义
save_file_path	(必须输入) 存贮订阅信息文件的目录，请设定一个真实存在的有可写权限的路径
log_level	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。系统不支持过夜，请确保每天开盘前重新启动

5.3.2.4 CreditCashRepay()

```
virtual uint64_t CreditCashRepay (
    double amount,
    uint64_t session_id ) [pure virtual]
```

融资融券业务中现金直接还款请求

返回

现金直接还款订单在XTP系统中的ID,如果为'0'表示消息发送失败，此时用户可以调用GetApiLastError()来获取错误代码，非“0”表示消息发送成功，用户需要记录下返回的xtp_id，它保证一个交易日内唯一，不同的交易日不保证唯一性

参数

<i>amount</i>	现金还款的金额
<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到

5.3.2.5 CreditCashRepayDebtInterestFee()

```
virtual uint64_t CreditCashRepayDebtInterestFee (  
    const char * debt_id,  
    double amount,  
    uint64_t session_id ) [pure virtual]
```

融资融券业务中现金还指定负债合约息费请求

返回

现金还息订单在XTP系统中的ID,如果为'0'表示消息发送失败,此时用户可以调用GetApiLastError()来获取错误代码,非"0"表示消息发送成功,用户需要记录下返回的xtp_id,它保证一个交易日内唯一,不同的交易日不保证唯一性

参数

<i>debt_id</i>	指定的负债合约编号
<i>amount</i>	现金还息的金额
<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到

5.3.2.6 CreditExtendDebtDate()

```
virtual uint64_t CreditExtendDebtDate (  
    XTPCreditDebtExtendReq * debt_extend,  
    uint64_t session_id ) [pure virtual]
```

融资融券业务中请求负债合约展期

返回

负债合约展期订单在XTP系统中的ID,如果为'0'表示消息发送失败,此时用户可以调用GetApiLastError()来获取错误代码,非"0"表示消息发送成功,用户需要记录下返回的xtp_id,它保证一个交易日内唯一,不同的交易日不保证唯一性

参数

<i>debt_extend</i>	负债合约展期的请求信息
<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到

5.3.2.7 CreditSellStockRepayDebtInterestFee()

```
virtual uint64_t CreditSellStockRepayDebtInterestFee (
    XTPOrderInsertInfo * order,
    const char * debt_id,
    uint64_t session_id ) [pure virtual]
```

融资融券业务中卖券还指定负债合约息费请求

返回

卖券还息订单在XTP系统中的ID,如果为'0'表示消息发送失败,此时用户可以调用GetApiLastError()来获取错误代码,非"0"表示消息发送成功,用户需要记录下返回的xtp_id,它保证一个交易日内唯一,不同的交易日不保证唯一性

参数

<i>order</i>	卖券的报单录入信息,其中order.order_client_id字段是用户自定义字段,用户输入什么值,订单响应OnOrderEvent()返回时就会带回什么值,类似于备注,方便用户自己定位订单。当然,如果你什么都不填,也是可以的。order.order_xtp_id字段无需用户填写,order.ticker必须不带空格,以'\0'结尾
<i>debt_id</i>	指定的负债合约编号
<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到

5.3.2.8 FundTransfer()

```
virtual uint64_t FundTransfer (
    XTPFundTransferReq * fund_transfer,
    uint64_t session_id ) [pure virtual]
```

资金划拨请求

返回

资金划拨订单在XTP系统中的ID,如果为'0'表示消息发送失败,此时用户可以调用GetApiLastError()来获取错误代码,非"0"表示消息发送成功,用户需要记录下返回的serial_id,它保证一个交易日内唯一,不同的交易日不保证唯一性

参数

<i>fund_transfer</i>	资金划拨的请求信息
<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到

备注

此函数支持一号两中心节点之间的资金划拨，注意资金划拨的方向。

5.3.2.9 GetAccountByXTPID()

```
virtual const char* GetAccountByXTPID (
    uint64_t order_xtp_id ) [pure virtual]
```

通过报单在xtp系统中的ID获取相关资金账户名

返回

返回资金账户名

参数

<i>order_xtp_id</i>	报单在xtp系统中的ID
---------------------	--------------

备注

只有资金账户登录成功后,才能得到正确的信息

5.3.2.10 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、InsertOrder、CancelOrder返回值为0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

5.3.2.11 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

5.3.2.12 GetClientIDByXTPID()

```
virtual uint8_t GetClientIDByXTPID (
    uint64_t order_xtp_id ) [pure virtual]
```

通过报单在xtp系统中的ID获取下单的客户端id

返回

返回客户端id，可以用此方法过滤自己下的订单

参数

<i>order_xtp_id</i>	报单在xtp系统中的ID
---------------------	--------------

备注

由于系统允许同一用户在不同客户端上登录操作，每个客户端通过不同的client_id进行区分

5.3.2.13 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

5.3.2.14 InsertOptionCombinedOrder()

```
virtual uint64_t InsertOptionCombinedOrder (
    XTPOptCombOrderInsertInfo * order,
    uint64_t session_id ) [pure virtual]
```

期权组合策略报单录入请求

返回

报单在XTP系统中的ID,如果为'0'表示报单发送失败，此时用户可以调用GetApiLastError()来获取错误代码，非“0”表示报单发送成功，用户需要记录下返回的order_xtp_id，它保证一个交易日内唯一，不同的交易日不保证唯一性

参数

order	报单录入信息，其中order.order_client_id字段是用户自定义字段，用户输入什么值，订单响应OnOptionCombinedOrderEvent()返回时就会带回什么值，类似于备注，方便用户自己定位订单。当然，如果你什么都不填，也是可以的。order.order_xtp_id字段无需用户填写，order.ticker必须不带空格，以'\0'结尾
session↔_id	资金账户对应的session_id,登录时得到

备注

交易所接收订单后，会在报单响应函数OnOptionCombinedOrderEvent()中返回报单未成交的状态，之后所有的订单状态改变（除了部成状态）都会通过报单响应函数返回

5.3.2.15 InsertOrder()

```
virtual uint64_t InsertOrder (
    XTPOrderInsertInfo * order,
    uint64_t session_id ) [pure virtual]
```

报单录入请求

返回

报单在XTP系统中的ID,如果为'0'表示报单发送失败，此时用户可以调用GetApiLastError()来获取错误代码，非“0”表示报单发送成功，用户需要记录下返回的order_xtp_id，它保证一个交易日内唯一，不同的交易日不保证唯一性

参数

order	报单录入信息，其中order.order_client_id字段是用户自定义字段，用户输入什么值，订单响应OnOrderEvent()返回时就会带回什么值，类似于备注，方便用户自己定位订单。当然，如果你什么都不填，也是可以的。order.order_xtp_id字段无需用户填写，order.ticker必须不带空格，以'\0'结尾
session↔_id	资金账户对应的session_id,登录时得到

备注

交易所接收订单后，会在报单响应函数OnOrderEvent()中返回报单未成交的状态，之后所有的订单状态改变（除了部成状态）都会通过报单响应函数返回

5.3.2.16 IsServerRestart()

```
virtual bool IsServerRestart (
    uint64_t session_id ) [pure virtual]
```

服务器是否重启过

返回

“true”表示重启过，“false”表示没有重启过

参数

<code>session_id</code>	资金账户对应的session_id,登录时得到
-------------------------	-------------------------

备注

此函数必须在Login之后调用

5.3.2.17 Login()

```
virtual uint64_t 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

session_id表明此资金账号登录是否成功，“0”表示登录失败，可以调用GetApiLastError()来获取错误代码，非“0”表示登录成功，此时需要记录下这个返回值session_id，与登录的资金账户对应

参数

<code>ip</code>	服务器地址，类似“127.0.0.1”
-----------------	---------------------

参数

<i>port</i>	服务器端口号
<i>user</i>	登录用户名
<i>password</i>	登录密码
<i>sock_type</i>	“1”代表TCP，“2”代表UDP，目前暂时只支持TCP
<i>local_ip</i>	本地网卡地址，类似“127.0.0.1”

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可支持多个账户连接，但是同一个账户同一个client_id只能有一个session连接，后面的登录在前一个session存续期间，无法连接

5.3.2.18 Logout()

```
virtual int Logout (
    uint64_t session_id ) [pure virtual]
```

登出请求

返回

登出是否成功，“0”表示登出成功，“-1”表示登出失败

参数

<i>session_id</i>	资金账户对应的session_id,登录时得到
-------------------	-------------------------

5.3.2.19 ModifyUserTerminalInfo()

```
virtual int ModifyUserTerminalInfo (
    const XTPUserTerminalInfoReq * info,
    uint64_t session_id ) [pure virtual]
```

修改已登录用户的硬件信息，仅限授权系统使用

返回

发送消息是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>info</i>	需要修改成的用户硬件信息
<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到

备注

此函数必须在Login之后调用，且仅限授权系统使用，一般客户无需使用

5.3.2.20 QueryAsset()

```
virtual int QueryAsset (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询资产

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.21 QueryCreditAssetDebtInfo()

```
virtual int QueryCreditAssetDebtInfo (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询信用账户待还资金信息

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session</i> ↔ <i>_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到
<i>request</i> ↔ <i>_id</i>	用于用户定位查询响应的ID, 由用户自定义

5.3.2.22 QueryCreditCashRepayInfo()

```
virtual int QueryCreditCashRepayInfo (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询融资融券业务中的现金直接还款报单

返回

查询是否成功, “0”表示成功, 非“0”表示出错, 此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session</i> ↔ <i>_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到
<i>request</i> ↔ <i>_id</i>	用于用户定位查询响应的ID, 由用户自定义

5.3.2.23 QueryCreditDebtInfo()

```
virtual int QueryCreditDebtInfo (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询信用账户负债合约信息

返回

查询是否成功, “0”表示成功, 非“0”表示出错, 此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session</i> ↔ <i>_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到
<i>request</i> ↔ <i>_id</i>	用于用户定位查询响应的ID, 由用户自定义

5.3.2.24 QueryCreditExcessStock()

```
virtual int QueryCreditExcessStock (
    XTPClientQueryCrdSurplusStkReqInfo * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

融资融券业务中请求查询指定证券的余券

返回
查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询的余券信息，不可以为空，需要明确指定
<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注
该方法中用户必须提供了证券代码和所在市场

5.3.2.25 QueryCreditExtendDebtDateOrders()

```
virtual int QueryCreditExtendDebtDateOrders (
    uint64_t xtp_id,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

融资融券业务中请求查询负债合约展期

返回
查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>xtp_id</i>	需要查询的负债合约展期订单筛选条件， <i>xtp_id</i> 可以为0，则默认所有负债合约展期订单，如果不为0，则请求特定的负债合约展期订单
<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.26 QueryCreditFundExtraInfo()

```
virtual int QueryCreditFundExtraInfo (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询融资融券业务中账的附加信息

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.27 QueryCreditFundInfo()

```
virtual int QueryCreditFundInfo (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询信用账户特有信息，除资金账户以外的信息

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.28 QueryCreditPositionExtraInfo()

```
virtual int QueryCreditPositionExtraInfo (
    XTPClientQueryCrdPositionStockReq * query_param,
```

```
uint64_t session_id,  
int request_id ) [pure virtual]
```

请求查询融资融券业务中账指定证券的附加信息

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要指定的证券，筛选条件中ticker可以全填0，如果不为0，请不带空格，并以'\0'结尾
<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.29 QueryCreditTickerAssignInfo()

```
virtual int QueryCreditTickerAssignInfo (  
    XTPClientQueryCrdPositionStockReq * query_param,  
    uint64_t session_id,  
    int request_id ) [pure virtual]
```

请求查询信用账户可融券头寸信息

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询的证券，筛选条件中ticker可以全填0，如果不为0，请不带空格，并以'\0'结尾
<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.30 QueryCreditTickerDebtInfo()

```
virtual int QueryCreditTickerDebtInfo (  
    XTPClientQueryCrdDebtStockReq * query_param,  
    uint64_t session_id,  
    int request_id ) [pure virtual]
```

请求查询指定证券负债未还信息

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询的指定证券，筛选条件中ticker可以全填0，如果不为0，请不带空格，并以'0'结尾
<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.31 QueryETF()

```
virtual int QueryETF (  
    XTPQueryETFBaseReq * query_param,  
    uint64_t session_id,  
    int request_id ) [pure virtual]
```

请求查询ETF清单文件

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询的ETF清单文件的筛选条件，其中合约代码可以为空，则默认所有存在的ETF合约代码，market字段也可以为初始值，则默认所有市场的ETF合约
<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.32 QueryETFTickerBasket()

```
virtual int QueryETFTickerBasket (  
    XTPQueryETFComponentReq * query_param,  
    uint64_t session_id,  
    int request_id ) [pure virtual]
```

请求查询ETF股票篮

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询股票篮的的ETF合约，其中合约代码不可以为空，market字段也必须指定
<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.33 QueryFundTransfer()

```
virtual int QueryFundTransfer (
    XTPQueryFundTransferLogReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询资金划拨

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询的资金划拨订单筛选条件，其中serial_id可以为0，则默认所有资金划拨订单，如果不为0，则请求特定的资金划拨订单
<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.34 QueryIPOInfoList()

```
virtual int QueryIPOInfoList (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询今日新股申购信息列表

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.35 QueryIPOQuotaInfo()

```
virtual int QueryIPOQuotaInfo (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询用户新股申购额度信息

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.36 QueryMulCreditExcessStock()

```
virtual int QueryMulCreditExcessStock (
    XTPClientQueryCrdSurplusStkReqInfo * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

融资融券业务中请求查询余券

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询的余券信息。若填入市场和股票代码，返回单支股票信息；若市场代码为空，股票代码非空，是无效查询，会在SPI中返回错误；若市场和股票代码均为空，返回全市场信息；若市场代码非空，股票代码为空，返回单市场信息。
<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.37 QueryOptionAuctionInfo()

```
virtual int QueryOptionAuctionInfo (
    XTPQueryOptionAuctionInfoReq * query_param,
```

```
uint64_t session_id,
int request_id ) [pure virtual]
```

请求查询期权合约

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

query_param	需要查询的期权合约的筛选条件，可以为NULL（为NULL表示查询所有的期权合约）
session_id	资金账户对应的session_id,登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

5.3.2.38 QueryOptionCombinedOrderByXTPID()

```
virtual int QueryOptionCombinedOrderByXTPID (
    const uint64_t order_xtp_id,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

根据报单ID请求查询期权组合策略报单

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

order_xtp_id	需要查询的报单在xtp系统中的ID，即InsertOrder()成功时返回的order_xtp_id
session_id	资金账户对应的session_id，登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

5.3.2.39 QueryOptionCombinedOrders()

```
virtual int QueryOptionCombinedOrders (
    const XTPQueryOptCombOrderReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询期权组合策略报单

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询的订单相关筛选条件，其中合约代码可以为空，则默认所有存在的合约代码，如果不为空，请不带空格，并以'\0'结尾，其中起始时间格式为YYYYMMDDHHMMSSsss，为0则默认当前交易日0点，结束时间格式为YYYYMMDDHHMMSSsss，为0则默认当前时间
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

该方法支持分时段查询，如果股票代码为空，则默认查询时间段内的所有报单，否则查询时间段内所有跟股票代码相关的报单，此函数查询出的结果可能对应多个查询结果响应。此函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.3.2.40 QueryOptionCombinedOrdersByPage()

```
virtual int QueryOptionCombinedOrdersByPage (
    const XTPQueryOptCombOrderByPageReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

分页请求查询期权组合策略报单

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要分页查询订单的条件，如果第一次查询，那么 <i>query_param.reference</i> 填0
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

该方法支持分页查询，注意用户需要记录下最后一笔查询结果的*reference*以使用户下次查询使用

5.3.2.41 QueryOptionCombinedPosition()

```
virtual int QueryOptionCombinedPosition (
    const XTPQueryOptCombPositionReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询投资者期权组合策略持仓

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

query_param	需要查询持仓的筛选条件，其中组合策略代码可以初始化为空，表示查询所有，如果不为空，请不带空格，并以'\0'结尾，注意需与market匹配，不匹配的话，可能导致查询不到所需的持仓
session_id	资金账户对应的session_id,登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

备注

该方法如果用户提供了合约代码，则会查询此合约的持仓信息（注意请指定market，如果market为0，可能会查询到2个市场的持仓，如果market为其他非有效值，则查询结果会返回找不到持仓），如果合约代码为空，则默认查询所有持仓信息。

5.3.2.42 QueryOptionCombinedStrategyInfo()

```
virtual int QueryOptionCombinedStrategyInfo (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询期权组合策略信息

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

session_id	资金账户对应的session_id,登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

备注

该方法仅支持精确查询，不支持模糊查询

5.3.2.43 QueryOptionCombinedTrades()

```
virtual int QueryOptionCombinedTrades (
    const XTPQueryOptCombTraderReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询期权组合策略的成交回报

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

query_param	需要查询的成交回报筛选条件，其中合约代码可以为空，则默认所有存在的合约代码，如果不为空，请不带空格，并以'\0'结尾，其中起始时间格式为YYYYMMDDHHMMSSsss，为0则默认当前交易日0点，结束时间格式为YYYYMMDDHHMMSSsss，为0则默认当前时间
session_id	资金账户对应的session_id,登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

备注

该方法支持分时段查询，如果股票代码为空，则默认查询时间段内的所有成交回报，否则查询时间段内所有跟股票代码相关的成交回报，此函数查询出的结果可能对应多个查询结果响应。此函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.3.2.44 QueryOptionCombinedTradesByPage()

```
virtual int QueryOptionCombinedTradesByPage (
    const XTPQueryOptCombTraderByPageReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

分页请求查询期权组合策略成交回报

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要分页查询成交回报的条件，如果第一次查询，那么reference填0
<i>session_id</i>	资金账户对应的session_id，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

该方法支持分页查询，注意用户需要记录下最后一笔查询结果的reference以使用户下次查询使用

5.3.2.45 QueryOptionCombinedTradesByXTPID()

```
virtual int QueryOptionCombinedTradesByXTPID (
    const uint64_t order_xtp_id,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

根据期权组合策略委托编号请求查询相关成交

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>order_xtp_id</i>	需要查询的委托编号，即InsertOrder()成功时返回的order_xtp_id
<i>session_id</i>	资金账户对应的session_id，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

此函数查询出的结果可能对应多个查询结果响应

5.3.2.46 QueryOptionCombinedUnfinishedOrders()

```
virtual int QueryOptionCombinedUnfinishedOrders (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询期权组合策略未完结报单

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.47 QueryOrderByXTPID()

```
virtual int QueryOrderByXTPID (
    const uint64_t order_xtp_id,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

根据报单ID请求查询报单

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>order_xtp_id</i>	需要查询的报单在xtp系统中的ID，即InsertOrder()成功时返回的 <i>order_xtp_id</i>
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.48 QueryOrders()

```
virtual int QueryOrders (
    const XTPQueryOrderReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询报单

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询的订单相关筛选条件，其中合约代码可以为空，则默认所有存在的合约代码，如果不为空，请不带空格，并以'\0'结尾，其中起始时间格式为YYYYMMDDHHMMSSsss，为0则默认当前交易日0点，结束时间格式为YYYYMMDDHHMMSSsss，为0则默认当前时间
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

该方法支持分时段查询，如果股票代码为空，则默认查询时间段内的所有报单，否则查询时间段内所有跟股票代码相关的报单，此函数查询出的结果可能对应多个查询结果响应。此函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.3.2.49 QueryOrdersByPage()

```
virtual int QueryOrdersByPage (
    const XTPQueryOrderByPageReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

分页请求查询报单

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要分页查询订单的条件，如果第一次查询，那么 <i>query_param.reference</i> 填0
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

该方法支持分页查询，注意用户需要记录下最后一笔查询结果的*reference*以便用户下次查询使用

5.3.2.50 QueryPosition()

```
virtual int QueryPosition (
    const char * ticker,
    uint64_t session_id,
    int request_id,
    XTP_MARKET_TYPE market = XTP_MKT_INIT ) [pure virtual]
```

请求查询投资者持仓

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>ticker</i>	需要查询持仓的合约代码，可以为NULL，表示查询全市场，如果不为NULL，请不带空格，并以'\0'结尾，注意需与market匹配，不匹配的话，可能由于证券代码沪深2个市场有重复，而导致查询不到所需的持仓
<i>market</i>	需要查询持仓的合约所在市场，默认为0，仅在合约代码不为NULL的时候，才会使用。market不指定或者为非0的其他非有效值情况下，可能由于证券代码沪深2个市场有重复，而导致查询不到所需的持仓。如果想正确查询指定持仓，请指定market
<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

该方法如果用户提供了合约代码，则会查询此合约的持仓信息（注意请指定market，如果market为0，可能会查询到2个市场的持仓，如果market为其他非有效值，则查询结果会返回找不到持仓），如果合约代码为空，则默认查询所有持仓信息。

5.3.2.51 QueryStructuredFund()

```
virtual int QueryStructuredFund (
    XTPQueryStructuredFundInfoReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询分级基金

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>query_param</i>	需要查询的分级基金筛选条件，其中母基金代码可以为空，则默认所有存在的母基金，如果不为空，请不带空格，并以'\0'结尾，其中交易市场不能为空
<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

此函数查询出的结果可能对应多个查询结果响应

5.3.2.52 QueryTrades()

```
virtual int QueryTrades (
    XTPQueryTraderReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询已成交

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

query_param	需要查询的成交回报筛选条件，其中合约代码可以为空，则默认所有存在的合约代码，如果不为空，请不带空格，并以'\0'结尾，其中起始时间格式为YYYYMMDDHHMMSSsss，为0则默认当前交易日0点，结束时间格式为YYYYMMDDHHMMSSsss，为0则默认当前时间
session_id	资金账户对应的session_id,登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

备注

该方法支持分时段查询，如果股票代码为空，则默认查询时间段内的所有成交回报，否则查询时间段内所有跟股票代码相关的成交回报，此函数查询出的结果可能对应多个查询结果响应。此函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.3.2.53 QueryTradesByPage()

```
virtual int QueryTradesByPage (
    const XTPQueryTraderByPageReq * query_param,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

分页请求查询成交回报

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

query_param	需要分页查询成交回报的条件，如果第一次查询，那么reference填0
session_id	资金账户对应的session_id，登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

备注

该方法支持分页查询，注意用户需要记录下最后一笔查询结果的reference以便用户下次查询使用

5.3.2.54 QueryTradesByXTPID()

```
virtual int QueryTradesByXTPID (
    const uint64_t order_xtp_id,
    uint64_t session_id,
    int request_id ) [pure virtual]
```

根据委托编号请求查询相关成交

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>order_xtp_id</i>	需要查询的委托编号，即InsertOrder()成功时返回的order_xtp_id
<i>session_id</i>	资金账户对应的session_id，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

此函数查询出的结果可能对应多个查询结果响应

5.3.2.55 QueryUnfinishedOrders()

```
virtual int QueryUnfinishedOrders (
    uint64_t session_id,
    int request_id ) [pure virtual]
```

请求查询未完结报单

返回

查询是否成功，“0”表示成功，非“0”表示出错，此时用户可以调用GetApiLastError()来获取错误代码

参数

<i>session_id</i>	资金账户对应的session_id，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

5.3.2.56 RegisterSpi()

```
virtual void RegisterSpi (
    TraderSpi * 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
------------	----------------------

5.3.2.57 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

5.3.2.58 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

5.3.2.59 SetSoftwareKey()

```
virtual void SetSoftwareKey (
    const char * key ) [pure virtual]
```

设置软件开发Key

参数

<i>key</i>	用户开发软件Key，用户申请开户时给予，以'\0'结尾
------------	-----------------------------

备注

此函数必须在Login之前调用

5.3.2.60 SetSoftwareVersion()

```
virtual void SetSoftwareVersion (
    const char * version ) [pure virtual]
```

设置软件开发版本号

参数

<i>version</i>	用户开发软件版本号，非api发行版本号，长度不超过15位，以'\0'结尾
----------------	--------------------------------------

备注

此函数必须在Login之前调用，标识的是客户端版本号，而不是API的版本号，由用户自定义

5.3.2.61 SubscribePublicTopic()

```
virtual void SubscribePublicTopic (
    XTP_TE_RESUME_TYPE resume_type ) [pure virtual]
```

订阅公共流。

参数

<i>resume_type</i>	公共流（订单响应、成交回报）重传方式 XTP_TERT_RESTART:从本交易日开始重传 XTP_TERT_RESUME:(保留字段，此方式暂未支持)从上次收到的续传 XTP_TERT_QUICK:只传送登录后公共流的内容
--------------------	---

备注

该方法要在Login方法前调用。若不调用则不会收到公共流的数据。注意在用户断线后，如果不登出就login()，公共流订阅方式不会起作用。用户只会收到断线后的所有消息。如果先logout()再login()，那么公共流订阅方式会起作用，用户收到的数据会根据用户的选择方式而定。

该类的文档由以下文件生成:

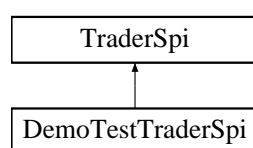
- [xtp_trader_api.h](#)

5.4 TraderSpi类 参考

交易接口响应类

```
#include <xtp_trader_api.h>
```

类 TraderSpi 继承关系图:



Public 成员函数

- virtual void [OnDisconnected](#) (uint64_t session_id, int reason)
- virtual void [OnError](#) (XTPRI *error_info)
- virtual void [OnOrderEvent](#) (XTPOrderInfo *order_info, XTPRI *error_info, uint64_t session_id)
- virtual void [OnTradeEvent](#) (XTPTradeReport *trade_info, uint64_t session_id)
- virtual void [OnCancelOrderError](#) (XTPOrderCancelInfo *cancel_info, XTPRI *error_info, uint64_t session_id)
- virtual void [OnQueryOrder](#) (XTPQueryOrderRsp *order_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryOrderByPage](#) (XTPQueryOrderRsp *order_info, int64_t req_count, int64_t order_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryTrade](#) (XTPQueryTradeRsp *trade_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryTradeByPage](#) (XTPQueryTradeRsp *trade_info, int64_t req_count, int64_t trade_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryPosition](#) (XTPQueryStkPositionRsp *position, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryAsset](#) (XTPQueryAssetRsp *asset, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryStructuredFund](#) (XTPStructuredFundInfo *fund_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryFundTransfer](#) (XTPFundTransferNotice *fund_transfer_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnFundTransfer](#) (XTPFundTransferNotice *fund_transfer_info, XTPRI *error_info, uint64_t session_id)
- virtual void [OnQueryETF](#) (XTPQueryETFBaseRsp *etf_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryETFBasket](#) (XTPQueryETFComponentRsp *etf_component_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryIPOInfoList](#) (XTPQueryIPOTickerRsp *ipo_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void [OnQueryIPOQuotaInfo](#) (XTPQueryIPOQuotaRsp *quota_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)

- virtual void `OnQueryOptionAuctionInfo` (`XTPQueryOptionAuctionInfoRsp` *option_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnCreditCashRepay` (`XTPCrdCashRepayRsp` *cash_repay_info, `XTPRI` *error_info, uint64_t session_id)
- virtual void `OnCreditCashRepayDebtInterestFee` (`XTPCrdCashRepayDebtInterestFeeRsp` *cash_repay_info, `XTPRI` *error_info, uint64_t session_id)
- virtual void `OnQueryCreditCashRepayInfo` (`XTPCrdCashRepayInfo` *cash_repay_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryCreditFundInfo` (`XTPCrdFundInfo` *fund_info, `XTPRI` *error_info, int request_id, uint64_t session_id)
- virtual void `OnQueryCreditDebtInfo` (`XTPCrdDebtInfo` *debt_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryCreditTickerDebtInfo` (`XTPCrdDebtStockInfo` *debt_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryCreditAssetDebtInfo` (double remain_amount, `XTPRI` *error_info, int request_id, uint64_t session_id)
- virtual void `OnQueryCreditTickerAssignInfo` (`XTPClientQueryCrdPositionStkInfo` *assign_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryCreditExcessStock` (`XTPClientQueryCrdSurplusStkRspInfo` *stock_info, `XTPRI` *error_info, int request_id, uint64_t session_id)
- virtual void `OnQueryMulCreditExcessStock` (`XTPClientQueryCrdSurplusStkRspInfo` *stock_info, `XTPRI` *error_info, int request_id, uint64_t session_id, bool is_last)
- virtual void `OnCreditExtendDebtDate` (`XTPCreditDebtExtendNotice` *debt_extend_info, `XTPRI` *error_info, uint64_t session_id)
- virtual void `OnQueryCreditExtendDebtDateOrders` (`XTPCreditDebtExtendNotice` *debt_extend_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryCreditFundExtraInfo` (`XTPCrdFundExtraInfo` *fund_info, `XTPRI` *error_info, int request_id, uint64_t session_id)
- virtual void `OnQueryCreditPositionExtraInfo` (`XTPCrdPositionExtraInfo` *fund_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnOptionCombinedOrderEvent` (`XTPOptCombOrderInfo` *order_info, `XTPRI` *error_info, uint64_t session_id)
- virtual void `OnOptionCombinedTradeEvent` (`XTPOptCombTradeReport` *trade_info, uint64_t session_id)
- virtual void `OnCancelOptionCombinedOrderError` (`XTPOptCombOrderCancelInfo` *cancel_info, `XTPRI` *error_info, uint64_t session_id)
- virtual void `OnQueryOptionCombinedOrders` (`XTPQueryOptCombOrderRsp` *order_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryOptionCombinedOrdersByPage` (`XTPQueryOptCombOrderRsp` *order_info, int64_t req_count, int64_t order_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryOptionCombinedTrades` (`XTPQueryOptCombTradeRsp` *trade_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryOptionCombinedTradesByPage` (`XTPQueryOptCombTradeRsp` *trade_info, int64_t req_count, int64_t trade_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryOptionCombinedPosition` (`XTPQueryOptCombPositionRsp` *position_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void `OnQueryOptionCombinedStrategyInfo` (`XTPQueryCombineStrategyInfoRsp` *strategy_info, `XTPRI` *error_info, int request_id, bool is_last, uint64_t session_id)

5.4.1 详细描述

交易接口响应类

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 OnCancelOptionCombinedOrderError()

```
virtual void OnCancelOptionCombinedOrderError (
    XTPOptCombOrderCancelInfo * cancel_info,
    XTPRI * error_info,
    uint64_t session_id ) [inline], [virtual]
```

期权组合策略撤单出错响应

参数

cancel_info	撤单具体信息，包括撤单的order_cancel_xtp_id和待撤单的order_xtp_id
error_info	撤单被拒绝或者发生错误时错误代码和错误信息，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，当error_info为空，或者error_info.error_id为0时，表明没有错误
session_id	资金账户对应的session_id，登录时得到

备注

此响应只会在撤单发生错误时被回调

5.4.2.2 OnCancelOrderError()

```
virtual void OnCancelOrderError (
    XTPOrderCancelInfo * cancel_info,
    XTPRI * error_info,
    uint64_t session_id ) [inline], [virtual]
```

撤单出错响应

参数

cancel_info	撤单具体信息，包括撤单的order_cancel_xtp_id和待撤单的order_xtp_id
error_info	撤单被拒绝或者发生错误时错误代码和错误信息，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，当error_info为空，或者error_info.error_id为0时，表明没有错误
session_id	资金账户对应的session_id，登录时得到

备注

此响应只会在撤单发生错误时被回调

5.4.2.3 OnCreditCashRepay()

```
virtual void OnCreditCashRepay (
    XTPCrdCashRepayRsp * cash_repay_info,
    XTPRI * error_info,
    uint64_t session_id ) [inline], [virtual]
```

融资融券业务中现金直接还款的响应

参数

<i>cash_repay_info</i>	现金直接还款通知的具体信息，用户可以通过 <i>cash_repay_info.xtp_id</i> 来管理订单，通过 <i>GetClientIDByXTPID() == client_id</i> 来过滤自己的订单。
<i>error_info</i>	现金还款发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

5.4.2.4 OnCreditCashRepayDebtInterestFee()

```
virtual void OnCreditCashRepayDebtInterestFee (
    XTPCrdCashRepayDebtInterestFeeRsp * cash_repay_info,
    XTPRI * error_info,
    uint64_t session_id ) [inline], [virtual]
```

融资融券业务中现金还息的响应

参数

<i>cash_repay_info</i>	现金还息通知的具体信息，用户可以通过 <i>cash_repay_info.xtp_id</i> 来管理订单，通过 <i>GetClientIDByXTPID() == client_id</i> 来过滤自己的订单。
<i>error_info</i>	现金还息发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

5.4.2.5 OnCreditExtendDebtDate()

```
virtual void OnCreditExtendDebtDate (
    XTPCreditDebtExtendNotice * debt_extend_info,
    XTPRI * error_info,
    uint64_t session_id ) [inline], [virtual]
```

融资融券业务中负债合约展期的通知

参数

debt_extend_info	负债合约展期通知的具体信息，用户可以通过debt_extend_info.xtpid来管理订单，通过GetClientIDByXTPID() == client_id来过滤自己的订单。
error_info	负债合约展期订单被拒绝或者发生错误时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。
session_id	资金账户对应的session_id，登录时得到

备注

当负债合约展期订单有状态变化的时候，会被调用，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。所有登录了此用户的客户端都将收到此用户的负债合约展期通知。

5.4.2.6 OnDisconnected()

```
virtual void OnDisconnected (
    uint64_t session_id,
    int reason ) [inline], [virtual]
```

当客户端的某个连接与交易后台通信连接断开时，该方法被调用。

参数

reason	错误原因，请与错误代码表对应
session↔_id	资金账户对应的session_id，登录时得到

备注

用户主动调用logout导致的断线，不会触发此函数。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，并更新session_id，此时用户收到的数据跟断线之前是连续的

被 [DemoTestTraderSpi](#) 重载.

5.4.2.7 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 [DemoTestTraderSpi](#) 重载.

5.4.2.8 OnFundTransfer()

```
virtual void OnFundTransfer (
    XTPFundTransferNotice * fund_transfer_info,
    XTPRI * error_info,
    uint64_t session_id ) [inline], [virtual]
```

资金划拨通知

参数

<i>fund_transfer_info</i>	资金划拨通知的具体信息, 用户可以通过fund_transfer_info.serial_id来管理订单, 通过GetClientIDByXTPID() == client_id来过滤自己的订单。
<i>error_info</i>	资金划拨订单被拒绝或者发生错误时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。当资金划拨方向为一号两中心节点之间划拨, 且error_info.error_id=11000384时, error_info.error_msg为结点中可用于划拨的资金(以整数为准), 用户需进行stringToInt的转化, 可据此填写合适的资金, 再次发起划拨请求
<i>session_id</i>	资金账户对应的session_id, 登录时得到

备注

当资金划拨订单有状态变化的时候, 会被调用, 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。所有登录了此用户的客户端都将收到此用户的资金划拨通知。

被 `DemoTestTraderSpi` 重载.

5.4.2.9 OnOptionCombinedOrderEvent()

```
virtual void OnOptionCombinedOrderEvent (
    XTOptCombOrderInfo * order_info,
    XTPRI * error_info,
    uint64_t session_id ) [inline], [virtual]
```

期权组合策略报单通知

参数

<i>order_info</i>	订单响应具体信息，用户可以通过 <i>order_info.order_xtp_id</i> 来管理订单，通过 <i>GetClientIDByXTPID() == client_id</i> 来过滤自己的订单， <i>order_info.qty_left</i> 字段在订单为未成交、部成、全成、废单状态时，表示此订单还没有成交的数量，在部撤、全撤状态时，表示此订单被撤的数量。 <i>order_info.order_cancel_xtp_id</i> 为其所对应的撤单ID，不为0时表示此单被撤成功
<i>error_info</i>	订单被拒绝或者发生错误时错误代码和错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

每次订单状态更新时，都会被调用，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，在订单未成交、全部成交、全部撤单、部分撤单、已拒绝这些状态时会有响应，对于部分成交的情况，请由订单的成交回报来自行确认。所有登录了此用户的客户端都将收到此用户的订单响应

5.4.2.10 OnOptionCombinedTradeEvent()

```
virtual void OnOptionCombinedTradeEvent (
    XTOptCombTradeReport * trade_info,
    uint64_t session_id ) [inline], [virtual]
```

期权组合策略成交通知

参数

<i>trade_info</i>	成交回报的具体信息，用户可以通过 <i>trade_info.order_xtp_id</i> 来管理订单，通过 <i>GetClientIDByXTPID() == client_id</i> 来过滤自己的订单。对于上交所， <i>exec_id</i> 可以唯一标识一笔成交。当发现2笔成交回报拥有相同的 <i>exec_id</i> ，则可以认为此笔交易自成交了。对于深交所， <i>exec_id</i> 是唯一的，暂时无此判断机制。 <i>report_index+market</i> 字段可以组成唯一标识表示成交回报。
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

订单有成交发生的时候，会被调用，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。所有登录了此用户的客户端都将收到此用户的成交回报。相关订单为部成状态，需要用户通过成交回报的成交数量来确定，`OnOrderEvent()`不会推送部成状态。

5.4.2.11 OnOrderEvent()

```
virtual void OnOrderEvent (
    XTPOrderInfo * order_info,
    XTPRI * error_info,
    uint64_t session_id ) [inline], [virtual]
```

报单通知

参数

<i>order_info</i>	订单响应具体信息，用户可以通过 <i>order_info.order_xtp_id</i> 来管理订单，通过 <i>GetClientIDByXTPID() == client_id</i> 来过滤自己的订单， <i>order_info.qty_left</i> 字段在订单为未成交、部成、全成、废单状态时，表示此订单还没有成交的数量，在部撤、全撤状态时，表示此订单被撤的数量。 <i>order_info.order_cancel_xtp_id</i> 为其所对应的撤单ID，不为0时表示此单被撤成功
<i>error_info</i>	订单被拒绝或者发生错误时错误代码和错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

每次订单状态更新时，都会被调用，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，在订单未成交、全部成交、全部撤单、部分撤单、已拒绝这些状态时会有响应，对于部分成交的情况，请由订单的成交回报来自行确认。所有登录了此用户的客户端都将收到此用户的订单响应

被 `DemoTestTraderSpi` 重载.

5.4.2.12 OnQueryAsset()

```
virtual void OnQueryAsset (
    XTPQueryAssetRsp * asset,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询资金账户响应，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

参数

<i>asset</i>	查询到的资金账户情况
<i>error_info</i>	查询资金账户发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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 [DemoTestTraderSpi](#) 重载.

5.4.2.13 OnQueryCreditAssetDebtInfo()

```
virtual void OnQueryCreditAssetDebtInfo (
    double remain_amount,
    XTPRI * error_info,
    int request_id,
    uint64_t session_id ) [inline], [virtual]
```

请求查询信用账户待还资金的响应，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

参数

<i>remain_amount</i>	查询到的信用账户待还资金
<i>error_info</i>	查询信用账户待还资金发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

5.4.2.14 OnQueryCreditCashRepayInfo()

```
virtual void OnQueryCreditCashRepayInfo (
    XTPCrdCashRepayInfo * cash_repay_info,
```

```

XTPRI * error_info,
int request_id,
bool is_last,
uint64_t session_id ) [inline], [virtual]

```

请求查询融资融券业务中的现金直接还款报单的响应

参数

<i>cash_repay_info</i>	查询到的某一笔现金直接还款通知的具体信息
<i>error_info</i>	查询现金直接报单发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

5.4.2.15 OnQueryCreditDebtInfo()

```

virtual void OnQueryCreditDebtInfo (
    XTPCrdDebtInfo * debt_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]

```

请求查询信用账户负债信息的响应，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

参数

<i>debt_info</i>	查询到的信用账户合约负债情况
<i>error_info</i>	查询信用账户负债信息发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

5.4.2.16 OnQueryCreditExcessStock()

```
virtual void OnQueryCreditExcessStock (
    XTPClientQueryCrdSurplusStkRspInfo * stock_info,
    XTPRI * error_info,
    int request_id,
    uint64_t session_id ) [inline], [virtual]
```

融资融券业务中请求查询指定余券信息的响应，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

参数

stock_info	查询到的余券信息
error_info	查询信用账户余券信息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
request_id	此消息响应函数对应的请求ID
session_id	资金账户对应的session_id，登录时得到

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

5.4.2.17 OnQueryCreditExtendDebtDateOrders()

```
virtual void OnQueryCreditExtendDebtDateOrders (
    XTPCreditDebtExtendNotice * debt_extend_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

查询融资融券业务中负债合约展期订单响应，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

参数

debt_extend_info	查询到的负债合约展期情况
error_info	查询负债合约展期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。当error_info.error_id=11000350时，表明没有记录，当为其他非0值时，表明合约发生拒单时的错误原因
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
session_id	资金账户对应的session_id，登录时得到

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

5.4.2.18 OnQueryCreditFundExtraInfo()

```
virtual void OnQueryCreditFundExtraInfo (
    XTPCrdFundExtraInfo * fund_info,
    XTPRI * error_info,
    int request_id,
    uint64_t session_id ) [inline], [virtual]
```

查询融资融券业务中信用账户附加信息的响应，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

参数

<i>fund_info</i>	信用账户附加信息
<i>error_info</i>	查询信用账户附加信息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>request_id</i>	此消息响应函数对应的请求ID
<i>session_id</i>	资金账户对应的session_id，登录时得到

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

5.4.2.19 OnQueryCreditFundInfo()

```
virtual void OnQueryCreditFundInfo (
    XTPCrdFundInfo * fund_info,
    XTPRI * error_info,
    int request_id,
    uint64_t session_id ) [inline], [virtual]
```

请求查询信用账户额外信息的响应，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

参数

<i>fund_info</i>	查询到的信用账户额外信息情况
<i>error_info</i>	查询信用账户额外信息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>request_id</i>	此消息响应函数对应的请求ID
<i>session_id</i>	资金账户对应的session_id，登录时得到

制作者 Doxygen

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

5.4.2.20 OnQueryCreditPositionExtraInfo()

```
virtual void OnQueryCreditPositionExtraInfo (
    XTPCrdPositionExtraInfo * fund_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

查询融资融券业务中信用账户指定证券的附加信息的响应，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

参数

<i>fund_info</i>	信用账户指定证券的附加信息
<i>error_info</i>	查询信用账户附加信息发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

5.4.2.21 OnQueryCreditTickerAssignInfo()

```
virtual void OnQueryCreditTickerAssignInfo (
    XTPClientQueryCrdPositionStkInfo * assign_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询信用账户可融券头寸信息的响应，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

参数

<i>assign_info</i>	查询到的信用账户可融券头寸信息
<i>error_info</i>	查询信用账户可融券头寸信息发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

5.4.2.22 OnQueryCreditTickerDebtInfo()

```
virtual void OnQueryCreditTickerDebtInfo (
    XTPCrddbtStockInfo * debt_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询信用账户指定证券负债未还信息响应，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

参数

<i>debt_info</i>	查询到的信用账户指定证券负债未还信息情况
<i>error_info</i>	查询信用账户指定证券负债未还信息发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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

5.4.2.23 OnQueryETF()

```
virtual void OnQueryETF (
    XTPQueryETFBaseRsp * etf_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询ETF清单文件的响应，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

参数

etf_info	查询到的ETF清单文件情况
error_info	查询ETF清单文件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
request_id↔	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
session_id↔	资金账户对应的session_id，登录时得到

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 [DemoTestTraderSpi](#) 重载.

5.4.2.24 OnQueryETFBasket()

```
virtual void OnQueryETFBasket (
    XTPQueryETFComponentRsp * etf_component_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询ETF股票篮的响应，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

参数

etf_component_info	查询到的ETF合约的相关成分股信息
error_info	查询ETF股票篮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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
session_id	资金账户对应的session_id，登录时得到

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 [DemoTestTraderSpi](#) 重载.

5.4.2.25 OnQueryFundTransfer()

```
virtual void OnQueryFundTransfer (
    XTPFundTransferNotice * fund_transfer_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询资金划拨订单响应，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

参数

<i>fund_transfer_info</i>	查询到的资金账户情况
<i>error_info</i>	查询资金账户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>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的session_id，登录时得到

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 [DemoTestTraderSpi](#) 重载.

5.4.2.26 OnQueryIPOInfoList()

```
virtual void OnQueryIPOInfoList (
    XTPQueryIPOTickerRsp * ipo_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询今日新股申购信息列表的响应，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

参数

<i>ipo_info</i>	查询到的今日新股申购的一只股票信息
<i>error_info</i>	查询今日新股申购信息列表发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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 [DemoTestTraderSpi](#) 重载.

5.4.2.27 OnQueryIPOQuotaInfo()

```
virtual void OnQueryIPOQuotaInfo (
    XTPQueryIPOQuotaRsp * quota_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询用户新股申购额度信息的响应，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

参数

<i>quota_info</i>	查询到的用户某个市场的今日新股申购额度信息
<i>error_info</i>	查查询用户新股申购额度信息发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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 [DemoTestTraderSpi](#) 重载.

5.4.2.28 OnQueryMulCreditExcessStock()

```
virtual void OnQueryMulCreditExcessStock (
    XTPClientQueryCrdSurplusStkRspInfo * stock_info,
    XTPRI * error_info,
    int request_id,
    uint64_t session_id,
    bool is_last ) [inline], [virtual]
```

融资融券业务中请求查询余券信息的响应，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

参数

<i>stock_info</i>	查询到的余券信息
<i>error_info</i>	查询信用账户余券信息发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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

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

5.4.2.29 OnQueryOptionAuctionInfo()

```
virtual void OnQueryOptionAuctionInfo (
    XTPQueryOptionAuctionInfoRsp * option_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询期权合约的响应，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

参数

<i>option_info</i>	查询到的期权合约情况
<i>error_info</i>	查询期权合约发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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 [DemoTestTraderSpi](#) 重载.

5.4.2.30 OnQueryOptionCombinedOrders()

```
virtual void OnQueryOptionCombinedOrders (
    XTPQueryOptCombOrderRsp * order_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询期权组合策略报单响应

参数

<i>order_info</i>	查询到的一个报单
<i>error_info</i>	查询报单时发生错误时，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>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的session_id，登录时得到

备注

由于支持分时段查询，一个查询请求可能对应多个响应，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。此对应的请求函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.4.2.31 OnQueryOptionCombinedOrdersByPage()

```
virtual void OnQueryOptionCombinedOrdersByPage (
    XTPQueryOptCombOrderRsp * order_info,
    int64_t req_count,
    int64_t order_sequence,
    int64_t query_reference,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

分页请求查询期权组合策略报单响应

参数

<i>order_info</i>	查询到的一个报单
<i>req_count</i>	分页请求的最大数量
<i>order_sequence</i>	分页请求的当前回报数量
<i>query_reference</i>	当前报单信息所对应的查询索引，需要记录下来，在进行下一次分页查询的时候需要用到
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

当*order_sequence*为0，表明当次查询没有查到任何记录，当*is_last*为*true*时，如果*order_sequence*等于*req_count*，那么表示还有报单，可以进行下一次分页查询，如果不等，表示所有报单已经查询完毕。一个查询请求可能对应多个响应，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。

5.4.2.32 OnQueryOptionCombinedPosition()

```
virtual void OnQueryOptionCombinedPosition (
    XTPQueryOptCombPositionRsp * position_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询期权组合策略持仓响应

参数

<i>position_info</i>	查询到的一个持仓信息
<i>error_info</i>	查询持仓发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

一个查询请求可能对应多个响应，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。

5.4.2.33 OnQueryOptionCombinedStrategyInfo()

```
virtual void OnQueryOptionCombinedStrategyInfo (
    XTPQueryCombineStrategyInfoRsp * strategy_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询期权组合策略信息响应

参数

strategy_info	查询到的一个组合策略信息
error_info	查询成交回报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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
session_id	资金账户对应的session_id，登录时得到

备注

一个查询请求可能对应多个响应，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。

5.4.2.34 OnQueryOptionCombinedTrades()

```
virtual void OnQueryOptionCombinedTrades (
    XTPQueryOptCombTradeRsp * trade_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询期权组合策略成交响应

参数

trade_info	查询到的一个成交回报
error_info	查询成交回报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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
session_id	资金账户对应的session_id，登录时得到

备注

由于支持分时段查询，一个查询请求可能对应多个响应，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。此对应的请求函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.4.2.35 OnQueryOptionCombinedTradesByPage()

```
virtual void OnQueryOptionCombinedTradesByPage (
    XTPQueryOptCombTradeRsp * trade_info,
    int64_t req_count,
    int64_t trade_sequence,
    int64_t query_reference,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

分页请求查询期权组合策略成交响应

参数

trade_info	查询到的一个成交信息
req_count	分页请求的最大数量
trade_sequence	分页请求的当前回报数量
query_reference	当前报单信息所对应的查询索引，需要记录下来，在进行下一次分页查询的时候需要用到
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
session_id	资金账户对应的session_id，登录时得到

备注

当trade_sequence为0，表明当次查询没有查到任何记录，当is_last为true时，如果trade_sequence等于req_count，那么表示还有回报，可以进行下一次分页查询，如果不等，表示所有回报已经查询完毕。一个查询请求可能对应多个响应，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。

5.4.2.36 OnQueryOrder()

```
virtual void OnQueryOrder (
    XTPQueryOrderRsp * order_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询报单响应

参数

<i>order_info</i>	查询到的一个报单
<i>error_info</i>	查询报单时发生错误时，返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

由于支持分时段查询，一个查询请求可能对应多个响应，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 [DemoTestTraderSpi](#) 重载.

5.4.2.37 OnQueryOrderByPage()

```
virtual void OnQueryOrderByPage (  
    XTPQueryOrderRsp * order_info,  
    int64_t req_count,  
    int64_t order_sequence,  
    int64_t query_reference,  
    int request_id,  
    bool is_last,  
    uint64_t session_id ) [inline], [virtual]
```

分页请求查询报单响应

参数

<i>order_info</i>	查询到的一个报单
<i>req_count</i>	分页请求的最大数量
<i>order_sequence</i>	分页请求的当前回报数量
<i>query_reference</i>	当前报单信息所对应的查询索引，需要记录下来，在进行下一次分页查询的时候需要用到
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，当为最后一个的时候为 <i>true</i> ，如果为 <i>false</i> ，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

当order_sequence为0，表明当次查询没有查到任何记录，当is_last为true时，如果order_sequence等于req_count，那么表示还有报单，可以进行下一次分页查询，如果不等，表示所有报单已经查询完毕。一个查询请求可能对应多个响应，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。

5.4.2.38 OnQueryPosition()

```
virtual void OnQueryPosition (
    XTPQueryStkPositionRsp * position,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询投资者持仓响应

参数

<i>position</i>	查询到的一只股票的持仓情况
<i>error_info</i>	查询账户持仓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>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的session_id，登录时得到

备注

由于用户可能持有多个股票，一个查询请求可能对应多个响应，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 [DemoTestTraderSpi](#) 重载.

5.4.2.39 OnQueryStructuredFund()

```
virtual void OnQueryStructuredFund (
    XTPStructuredFundInfo * fund_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询分级基金信息响应，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

参数

<i>fund_info</i>	查询到的分级基金情况
<i>error_info</i>	查询分级基金发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

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 [DemoTestTraderSpi](#) 重载.

5.4.2.40 OnQueryTrade()

```
virtual void OnQueryTrade (
    XTPQueryTradeRsp * trade_info,
    XTPRI * error_info,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

请求查询成交响应

参数

<i>trade_info</i>	查询到的一个成交回报
<i>error_info</i>	查询成交回报发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为 <i>request_id</i> 这条请求所对应的最后一个响应，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
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

备注

由于支持分时段查询，一个查询请求可能对应多个响应，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 [DemoTestTraderSpi](#) 重载.

5.4.2.41 OnQueryTradeByPage()

```
virtual void OnQueryTradeByPage (
    XTPQueryTradeRsp * trade_info,
    int64_t req_count,
    int64_t trade_sequence,
    int64_t query_reference,
    int request_id,
    bool is_last,
    uint64_t session_id ) [inline], [virtual]
```

分页请求查询成交响应

参数

trade_info	查询到的一个成交信息
req_count	分页请求的最大数量
trade_sequence	分页请求的当前回报数量
query_reference	当前报单信息所对应的查询索引，需要记录下来，在进行下一次分页查询的时候需要用到
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，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
session_id	资金账户对应的session_id，登录时得到

备注

当trade_sequence为0，表明当次查询没有查到任何记录，当is_last为true时，如果trade_sequence等于req_count，那么表示还有回报，可以进行下一次分页查询，如果不等，表示所有回报已经查询完毕。一个查询请求可能对应多个响应，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。

5.4.2.42 OnTradeEvent()

```
virtual void OnTradeEvent (
    XTPTradeReport * trade_info,
    uint64_t session_id ) [inline], [virtual]
```

成交通知

参数

trade_info	成交回报的具体信息，用户可以通过trade_info.order_xtp_id来管理订单，通过GetClientIDByXTPID() == client_id来过滤自己的订单。对于上交所，exec_id可以唯一标识一笔成交。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了。对于深交所，exec_id是唯一的，暂时无此判断机制。report_index+market字段可以组成唯一标识表示成交回报。
session_id	资金账户对应的session_id，登录时得到

备注

订单有成交发生的时候，会被调用，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。所有登录了此用户的客户端都将收到此用户的成交回报。相关订单为部成状态，需要用户通过成交回报的成交数量来确定，OnOrderEvent()不会推送部成状态。

被 [DemoTestTraderSpi](#) 重载.

该类的文档由以下文件生成:

- [xtp_trader_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
- [int64_t yesterday_qty](#)
昨日日融券数量
- [int64_t left_qty](#)
剩余可融券数量
- [int64_t frozen_qty](#)
冻结融券数量

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](#) `unknown` [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
- 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 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

5.21.1 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](#)

5.22 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

5.22.1 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](#)

5.23 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)

5.23.1 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](#)

5.24 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

5.24.1 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](#)

5.25 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

### 5.25.1 详细描述

行情

该结构体的文档由以下文件生成:

- `xquote_api_struct.h`

## 5.26 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）

### 5.26.1 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`

## 5.27 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
- `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，用户无需关心
- `TXTPOrderTypeType order_type`  
报单类型
- `XTPOptCombPlugin opt_comb_info`  
期权组合策略信息

### 5.27.1 详细描述

期权组合策略报单响应结构体

### 5.27.2 结构体成员变量说明

#### 5.27.2.1 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`

## 5.28 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

### 5.28.1 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`

## 5.29 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。
- `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，填写拟拆分组组合的组合编码。
- `int32_t num_legs`  
成分合约数
- `XTPOptCombLegInfo leg_detail [XTP_STRATEGGE_LEG_NUM]`  
成分合约数组，最多四条腿。

### 5.29.1 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`

## 5.30 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, 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
- `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

### 5.30.1 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`

## 5.31 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

### 5.31.1 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](#)

## 5.32 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

### 5.32.1 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](#)

## 5.33 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

### 5.33.1 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`

## 5.34 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
- uint64\_t [unknown](#) [43 - 12 - 1 - 2]  
(保留字段)

### 5.34.1 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](#)

## 5.35 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_STRATEGE_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

### 5.35.1 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](#)

## 5.36 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

## 5.36.1 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](#)

## 5.37 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"50ETF"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](#)  
7日预估金额
- [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= $net\_value * unit$

### 5.37.1 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](#)

## 5.38 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

### 5.38.1 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](#)

## 5.39 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

## 5.39.1 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](#)

## 5.40 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](#)  
成分股替代标识为必须现金替代时候的总金额

### 5.40.1 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](#)

## 5.41 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

### 5.41.1 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](#)



## 5.42 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

### 5.42.1 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](#)

## 5.43 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

### 5.43.1 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](#)

## 5.44 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

### 5.44.1 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](#)

## 5.45 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

## 5.45.1 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](#)

## 5.46 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

## 5.46.1 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](#)

## 5.47 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

### 5.47.1 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](#)

## 5.48 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

### 5.48.1 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](#)

## 5.49 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

## 5.49.1 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](#)

## 5.50 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

## 5.50.1 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](#)

## 5.51 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

### 5.51.1 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](#)

## 5.52 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

### 5.52.1 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](#)

## 5.53 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)

### 5.53.1 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`

## 5.54 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



### 5.54.1 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](#)

## 5.55 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

### 5.55.1 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](#)

## 5.56 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

### 5.56.1 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](#)

## 5.57 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

### 5.57.1 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](#)

## 5.58 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
- uint32\_t `reserved1`  
保留字段1
- 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
- uint64\_t `unknown` [50 - 6]  
(保留字段)

## 5.58.1 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`

## 5.59 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

### 5.59.1 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](#)

## 5.60 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

### 5.60.1 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](#)

## 5.61 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

## 5.61.1 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](#)

## 5.62 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（仅在有限涨跌幅限制时有效）
- `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`  
市价卖委托数量下限
- `int32_t market_ask_qty_unit`  
市价卖数量单位
- `uint64_t unknown [4]`  
保留字段

### 5.62.1 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`

## 5.63 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)

## 5.63.1 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](#)

## 5.64 XTPRspInfoStruct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

### 5.64.1 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](#)

## 5.65 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

### 5.65.1 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](#)

## 5.66 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
- [uint32\\_t min\\_merge\\_qty](#)  
最小合并数量
- [double net\\_price](#)  
基金净值

## 5.66.1 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](#)

## 5.67 XTPTickByTickEntrust结构体 参考

逐笔委托(仅适用深交所)

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

## 成员变量

- [int32\\_t channel\\_no](#)  
频道代码
- [int64\\_t seq](#)  
委托序号(在同一个channel\_no内唯一, 从1开始连续)
- [double price](#)  
委托价格
- [int64\\_t qty](#)  
委托数量
- [char side](#)  
'1':买; '2':卖; 'G':借入; 'F':出借
- [char ord\\_type](#)  
订单类别: '1': 市价; '2': 限价; 'U': 本方最优

### 5.67.1 详细描述

逐笔委托(仅适用深交所)

该结构体的文档由以下文件生成:

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

## 5.68 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](#)  
预留
- [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](#)  
 };

### 5.68.1 详细描述

逐笔数据信息

该结构体的文档由以下文件生成:

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

## 5.69 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`  
委托序号(在同一个`channel_no`内唯一, 从1开始连续)
- `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`

### 5.69.1 详细描述

#### 逐笔成交

### 5.69.2 结构体成员变量说明

#### 5.69.2.1 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](#)

## 5.70 XTPTickerPricelInfo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

### 5.70.1 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`

## 5.71 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
- [TXTPTradeTypeType](#) [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

### 5.71.1 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](#)

## 5.72 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
- [char unused \[64\]](#)  
预留

### 5.72.1 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\_trade\_api.cpp 文件参考

定义控制台测试应用程序的入口点

```
#include "xtp_trader_api.h"
#include <string>
#include <map>
#include <iostream>
#include <unistd.h>
#include "demo_test_trade_spi.h"
```

函数

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

#### 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
int client_id = 1;//客户端标识

char filepath[] = "c:\\log\\";//真实存在的可读写路径

//初始化UserApi

XTP::API::TraderApi* user_api_pointer = XTP::API::TraderApi::CreateTraderApi(client_id, filepath, XTP_LOG_LEVEL_DEBUG); // 创建UserApi

user_api_pointer->SubscribePublicTopic(XTP_TERT_QUICK);//设定公共流传输方式

user_api_pointer->SetSoftwareKey("xxxxxxxxxxxxxxxxxxxxxx");//设定用户开发软件Key，用户申请开户时给予，以'\0'结尾

user_api_pointer->SetSoftwareVersion("1.1.0");//设定软件的开发版本号，非api版本号

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

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

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

uint64_t temp_session_ = user_api_pointer->Login(server_ip.c_str(), server_port, username.c_str(), password.c_str(), XTP_PROTOCOL_TCP);//登陆交易服务器

if (temp_session_ != 0)

{ int order_client_id = 1;//用户自定义用于标识本地订单的编号，可以任意

//下单

XTPOrderInsertInfo orderInsert;

memset(&orderInsert, 0, sizeof(XTPOrderInsertInfo));

orderInsert.order_client_id = order_client_id++;//用户自定义，用来标识订单，可以不填

std::string ticker("000002");

strcpy(orderInsert.ticker, ticker.c_str());

orderInsert.exchange_id = (XTP_EXCHANGE_TYPE)2; orderInsert.price = 17.5;

orderInsert.quantity = 200;

orderInsert.side = (XTP_SIDE_TYPE)1;

orderInsert.price_type = (XTP_PRICE_TYPE)3;

orderInsert.business_type = (XTP_BUSINESS_TYPE_CASH)0;

orderInsert.position_effect = (XTP_POSITION_EFFECT_INIT)0;//期权业务使用的字段，普通业务请用0

//返回的xtp_id需要记录下来，与服务器交互的时候，所有对订单的操作由xtp_id唯一确定

uint64_t insert_xtp_id = user_api_pointer->InsertOrder(&orderInsert,temp_session_);

if (insert_xtp_id == 0)

{

```



```

//下单失败

XTPRI* error_info = user_api_pointer->GetApiLastError(); //下单失败时的错误原因代码

}

//如果需要撤单 //返回的xtp_id需要记录下来，与服务器交互的时候，所有对订单的操作由xtp_id唯一确定
uint64_t cancel_xtp_id = user_api_pointer->CancelOrder(insert_xtp_id,temp_session_);

if (cancel_xtp_id == 0)
{

//撤单失败

XTPRI* error_info = user_api_pointer->GetApiLastError(); //撤单失败时的错误原因代码

}
}

else
{

XTPRI* error_info = user_api_pointer->GetApiLastError();

std::cout << "Login to server error, " << error_info->error_id << " : " << error_info->error_msg << std::endl;

}

return 0;

}

```

## 6.2 demo\_test\_trade\_spi.h 文件参考

Demo自定义客户端交易响应接口类

```
#include "xtp_trader_api.h"
```

结构体

- class [DemoTestTraderSpi](#)  
Demo自定义交易接口响应类

### 6.2.1 详细描述

Demo自定义客户端交易响应接口类

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

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 [XTPCrdCashRepayRsp](#)  
融资融券直接还款响应信息
- struct [XTPCrdCashRepayDebtInterestFeeRsp](#)  
融资融券现金还息费响应信息
- struct [XTPCrdCashRepayInfo](#)  
单条融资融券直接还款记录信息
- struct [XTPCrdDebtInfo](#)  
单条融资融券负债记录信息
- struct [XTPCrdFundInfo](#)  
融资融券特有帐户数据
- struct [XTPClientQueryCrdDebtStockReq](#)  
融资融券指定证券上的负债未还数量请求结构体
- struct [XTPCrdDebtStockInfo](#)  
融资融券指定证券的融券负债相关信息
- struct [XTPClientQueryCrdPositionStockReq](#)  
融券头寸证券查询请求结构体
- struct [XTPClientQueryCrdPositionStkInfo](#)  
融券头寸证券信息
- struct [XTPClientQueryCrdSurplusStkReqInfo](#)  
信用业务融券查询请求结构体
- struct [XTPClientQueryCrdSurplusStkRspInfo](#)  
信用业务融券信息
- struct [XTPCreditDebtExtendReq](#)  
用户展期请求
- struct [XTPCrdFundExtraInfo](#)  
融资融券帐户附加信息
- struct [XTPCrdPositionExtraInfo](#)  
融资融券帐户持仓附加信息
- struct [XTPOptCombOrderInsertInfo](#)  
期权组合策略新订单请求
- struct [XTPOptCombOrderInfo](#)  
期权组合策略报单响应结构体

- struct [XTOptCombTradeReport](#)  
期权组合策略报单成交结构体
- struct [XTPQueryOptCombOrderReq](#)  
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
- struct [XTPQueryOptCombOrderByPageReq](#)  
查询期权组合策略订单请求-分页查询
- struct [XTPQueryOptCombReportByExecIdReq](#)  
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）
- struct [XTPQueryOptCombTraderReq](#)  
查询期权组合策略成交回报请求-查询条件
- struct [XTPQueryOptCombTraderByPageReq](#)  
查询期权组合策略成交回报请求-分页查询

## 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  
信用业务余券查询请求结构体
- typedef struct [XTPClientQueryCrdSurplusStkRspInfo](#) XTPClientQueryCrdSurplusStkRspInfo  
信用业务余券信息
- 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 [XTPOptCombOrderInfo](#) XTPQueryOptCombOrderRsp  
期权组合策略报单查询响应结构体
- typedef struct [XTPOptCombTradeReport](#) 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(仅适用深交所)
- struct [XTPTickByTickTrade](#)

逐笔成交

- struct [XTPTickByTickStruct](#)  
逐笔数据信息
- struct [XPTTickerPriceInfo](#)  
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 [XPTTickerPriceInfo](#) 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_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_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\\_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\\_TYPE](#) *XTP\_OPT\_EXERCISE\_TYPE\_TYPE*  
*XTP\_OPT\_EXERCISE\_TYPE\_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 [TXTPTradeTypeType](#)  
*TXTPTradeTypeType*是成交类型类型
- typedef char [TXTPOrderTypeType](#)  
*TXTPOrderTypeType*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

## 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_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\_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\_S*, *XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_MERGE*, *XTP\_SPLIT\_MERGE\_STATUS\_FORBIDDEN* }
- XTP\_SPLIT\_MERGE\_STATUS* 是一个基金当天拆分合并状态类型
- 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\_COMPANY\_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\_SINGLE\_MARKET\_BOND* = 17, *XTP\_SECURITY\_ETF\_GOLD* = 19, *XTP\_SECURITY\_STRUCTURED* = 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

### 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_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\_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                                            |

#### 6.6.3.11 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 |

### 6.6.3.12 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 |

### 6.6.3.13 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 |

### 6.6.3.14 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  |

### 6.6.3.15 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 |

#### 6.6.3.16 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 |

#### 6.6.3.17 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 |

#### 6.6.3.18 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 |

### 6.6.3.19 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    |
| 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 |

### 6.6.3.20 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 |

### 6.6.3.21 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） |

#### 6.6.3.22 XTP\_PRICE\_TYPE

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

XTP\_PRICE\_TYPE是价格类型

枚举值

|                              |                                                 |
|------------------------------|-------------------------------------------------|
| XTP_PRICE_LIMIT              | 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） |
| XTP_PRICE_BEST_OR_CANCEL     | 即时成交剩余转撤销，市价单-深 / 沪期权 / 深期权                     |
| XTP_PRICE_BEST5_OR_LIMIT     | 最优五档即时成交剩余转限价，市价单-沪                             |
| XTP_PRICE_BEST5_OR_CANCEL    | 最优5档即时成交剩余转撤销，市价单-沪深 / 深期权                      |
| XTP_PRICE_ALL_OR_CANCEL      | 全部成交或撤销，市价单-深 / 沪期权 / 深期权                       |
| XTP_PRICE_FORWARD_BEST       | 本方最优，市价单-深 / 深期权                                |
| XTP_PRICE_REVERSE_BEST_LIMIT | 对方最优剩余转限价，市价单-深 / 沪期权 / 深期权                     |
| XTP_PRICE_LIMIT_OR_CANCEL    | 期权限价申报FOK                                       |
| XTP_PRICE_TYPE_UNKNOWN       | 未知或者无效价格类型                                      |

#### 6.6.3.23 XTP\_PROTOCOL\_TYPE

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

XTP\_PROTOCOL\_TYPE是通讯传输协议方式

枚举值

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

## 6.6.3.24 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       |

## 6.6.3.25 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          |
| 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_COMPANEY_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          |

### 6.6.3.26 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     |

### 6.6.3.27 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 |

### 6.6.3.28 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 |



## 6.6.3.29 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      |

## 6.6.3.30 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 |

## 6.6.3.31 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 [XTPRsplInfoStruct](#)  
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 [XTPRsplInfoStruct](#) [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\_trader\_api.h 文件参考

定义客户端交易接口

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

结构体

- class [TraderSpi](#)  
交易接口响应类
- class [TraderApi](#)  
交易接口类

### 6.9.1 详细描述

定义客户端交易接口

作者

中泰证券股份有限公司



# Index

CancelOptionCombinedOrder  
    XTP::API::TraderApi, [22](#)

CancelOrder  
    XTP::API::TraderApi, [23](#)

CreateTraderApi  
    XTP::API::TraderApi, [24](#)

CreditCashRepay  
    XTP::API::TraderApi, [24](#)

CreditCashRepayDebtInterestFee  
    XTP::API::TraderApi, [25](#)

CreditExtendDebtDate  
    XTP::API::TraderApi, [25](#)

CreditSellStockRepayDebtInterestFee  
    XTP::API::TraderApi, [26](#)

demo\_test\_trade\_api.cpp, [129](#)  
    main, [129](#)

demo\_test\_trade\_spi.h, [131](#)

DemoTestTraderSpi, [11](#)  
    OnFundTransfer, [12](#)  
    OnQueryAsset, [13](#)  
    OnQueryETFBasket, [14](#)  
    OnQueryETF, [13](#)  
    OnQueryFundTransfer, [14](#)  
    OnQueryIPOInfoList, [15](#)  
    OnQueryIPOQuotaInfo, [16](#)  
    OnQueryOptionAuctionInfo, [16](#)  
    OnQueryOrder, [17](#)  
    OnQueryPosition, [17](#)  
    OnQueryStructuredFund, [18](#)  
    OnQueryTrade, [19](#)

ETF\_REPLACE\_TYPE  
    xtp\_api\_data\_type.h, [144](#)

FundTransfer  
    XTP::API::TraderApi, [26](#)

GetAccountByXTPID  
    XTP::API::TraderApi, [27](#)

GetApiLastError  
    XTP::API::TraderApi, [27](#)

GetApiVersion  
    XTP::API::TraderApi, [27](#)

GetClientIDByXTPID  
    XTP::API::TraderApi, [28](#)

GetTradingDay  
    XTP::API::TraderApi, [28](#)

InsertOptionCombinedOrder  
    XTP::API::TraderApi, [28](#)

InsertOrder  
    XTP::API::TraderApi, [29](#)

IsServerRestart  
    XTP::API::TraderApi, [30](#)

Login  
    XTP::API::TraderApi, [30](#)

Logout  
    XTP::API::TraderApi, [31](#)

main  
    demo\_test\_trade\_api.cpp, [129](#)

market  
    XTPOptCombOrderInfo, [95](#)

ModifyUserTerminalInfo  
    XTP::API::TraderApi, [31](#)

OnCancelOptionCombinedOrderError  
    XTP::API::TraderSpi, [55](#)

OnCancelOrderError  
    XTP::API::TraderSpi, [55](#)

OnCreditCashRepay  
    XTP::API::TraderSpi, [56](#)

OnCreditCashRepayDebtInterestFee  
    XTP::API::TraderSpi, [56](#)

OnCreditExtendDebtDate  
    XTP::API::TraderSpi, [57](#)

OnDisconnected  
    XTP::API::TraderSpi, [57](#)

OnError  
    XTP::API::TraderSpi, [58](#)

OnFundTransfer  
    DemoTestTraderSpi, [12](#)  
    XTP::API::TraderSpi, [58](#)

OnOptionCombinedOrderEvent  
    XTP::API::TraderSpi, [59](#)

OnOptionCombinedTradeEvent  
    XTP::API::TraderSpi, [59](#)

OnOrderEvent  
    XTP::API::TraderSpi, [60](#)

OnQueryAsset  
    DemoTestTraderSpi, [13](#)  
    XTP::API::TraderSpi, [60](#)

OnQueryCreditAssetDebtInfo  
    XTP::API::TraderSpi, [61](#)

OnQueryCreditCashRepayInfo  
    XTP::API::TraderSpi, [61](#)

OnQueryCreditDebtInfo  
    XTP::API::TraderSpi, [62](#)

OnQueryCreditExcessStock

- XTP::API::TraderSpi, [62](#)
- OnQueryCreditExtendDebtDateOrders
  - XTP::API::TraderSpi, [63](#)
- OnQueryCreditFundExtraInfo
  - XTP::API::TraderSpi, [64](#)
- OnQueryCreditFundInfo
  - XTP::API::TraderSpi, [64](#)
- OnQueryCreditPositionExtraInfo
  - XTP::API::TraderSpi, [65](#)
- OnQueryCreditTickerAssignInfo
  - XTP::API::TraderSpi, [65](#)
- OnQueryCreditTickerDebtInfo
  - XTP::API::TraderSpi, [66](#)
- OnQueryETFBasket
  - DemoTestTraderSpi, [14](#)
  - XTP::API::TraderSpi, [67](#)
- OnQueryETF
  - DemoTestTraderSpi, [13](#)
  - XTP::API::TraderSpi, [66](#)
- OnQueryFundTransfer
  - DemoTestTraderSpi, [14](#)
  - XTP::API::TraderSpi, [68](#)
- OnQueryIPOInfoList
  - DemoTestTraderSpi, [15](#)
  - XTP::API::TraderSpi, [68](#)
- OnQueryIPOQuotaInfo
  - DemoTestTraderSpi, [16](#)
  - XTP::API::TraderSpi, [69](#)
- OnQueryMulCreditExcessStock
  - XTP::API::TraderSpi, [69](#)
- OnQueryOptionAuctionInfo
  - DemoTestTraderSpi, [16](#)
  - XTP::API::TraderSpi, [70](#)
- OnQueryOptionCombinedOrders
  - XTP::API::TraderSpi, [71](#)
- OnQueryOptionCombinedOrdersByPage
  - XTP::API::TraderSpi, [71](#)
- OnQueryOptionCombinedPosition
  - XTP::API::TraderSpi, [72](#)
- OnQueryOptionCombinedStrategyInfo
  - XTP::API::TraderSpi, [72](#)
- OnQueryOptionCombinedTrades
  - XTP::API::TraderSpi, [73](#)
- OnQueryOptionCombinedTradesByPage
  - XTP::API::TraderSpi, [74](#)
- OnQueryOrder
  - DemoTestTraderSpi, [17](#)
  - XTP::API::TraderSpi, [74](#)
- OnQueryOrderByPage
  - XTP::API::TraderSpi, [75](#)
- OnQueryPosition
  - DemoTestTraderSpi, [17](#)
  - XTP::API::TraderSpi, [76](#)
- OnQueryStructuredFund
  - DemoTestTraderSpi, [18](#)
  - XTP::API::TraderSpi, [76](#)
- OnQueryTrade
  - DemoTestTraderSpi, [19](#)
- XTP::API::TraderSpi, [77](#)
- OnQueryTradeByPage
  - XTP::API::TraderSpi, [77](#)
- OnTradeEvent
  - XTP::API::TraderSpi, [78](#)
- OrderBookStruct, [19](#)
- QueryAsset
  - XTP::API::TraderApi, [32](#)
- QueryCreditAssetDebtInfo
  - XTP::API::TraderApi, [32](#)
- QueryCreditCashRepayInfo
  - XTP::API::TraderApi, [33](#)
- QueryCreditDebtInfo
  - XTP::API::TraderApi, [33](#)
- QueryCreditExcessStock
  - XTP::API::TraderApi, [34](#)
- QueryCreditExtendDebtDateOrders
  - XTP::API::TraderApi, [34](#)
- QueryCreditFundExtraInfo
  - XTP::API::TraderApi, [35](#)
- QueryCreditFundInfo
  - XTP::API::TraderApi, [35](#)
- QueryCreditPositionExtraInfo
  - XTP::API::TraderApi, [35](#)
- QueryCreditTickerAssignInfo
  - XTP::API::TraderApi, [36](#)
- QueryCreditTickerDebtInfo
  - XTP::API::TraderApi, [36](#)
- QueryETFTickerBasket
  - XTP::API::TraderApi, [38](#)
- QueryETF
  - XTP::API::TraderApi, [38](#)
- QueryFundTransfer
  - XTP::API::TraderApi, [39](#)
- QueryIPOInfoList
  - XTP::API::TraderApi, [39](#)
- QueryIPOQuotaInfo
  - XTP::API::TraderApi, [39](#)
- QueryMulCreditExcessStock
  - XTP::API::TraderApi, [40](#)
- QueryOptionAuctionInfo
  - XTP::API::TraderApi, [40](#)
- QueryOptionCombinedOrderByXTPID
  - XTP::API::TraderApi, [41](#)
- QueryOptionCombinedOrders
  - XTP::API::TraderApi, [41](#)
- QueryOptionCombinedOrdersByPage
  - XTP::API::TraderApi, [42](#)
- QueryOptionCombinedPosition
  - XTP::API::TraderApi, [42](#)
- QueryOptionCombinedStrategyInfo
  - XTP::API::TraderApi, [43](#)
- QueryOptionCombinedTrades
  - XTP::API::TraderApi, [44](#)
- QueryOptionCombinedTradesByPage
  - XTP::API::TraderApi, [44](#)
- QueryOptionCombinedTradesByXTPID
  - XTP::API::TraderApi, [45](#)

- QueryOptionCombinedUnfinishedOrders
  - XTP::API::TraderApi, 45
- QueryOrderByXTPID
  - XTP::API::TraderApi, 46
- QueryOrders
  - XTP::API::TraderApi, 46
- QueryOrdersByPage
  - XTP::API::TraderApi, 47
- QueryPosition
  - XTP::API::TraderApi, 47
- QueryStructuredFund
  - XTP::API::TraderApi, 48
- QueryTrades
  - XTP::API::TraderApi, 48
- QueryTradesByPage
  - XTP::API::TraderApi, 49
- QueryTradesByXTPID
  - XTP::API::TraderApi, 50
- QueryUnfinishedOrders
  - XTP::API::TraderApi, 50
- RegisterSpi
  - XTP::API::TraderApi, 51
- Release
  - XTP::API::TraderApi, 51
- SetHeartBeatInterval
  - XTP::API::TraderApi, 51
- SetSoftwareKey
  - XTP::API::TraderApi, 51
- SetSoftwareVersion
  - XTP::API::TraderApi, 52
- SubscribePublicTopic
  - XTP::API::TraderApi, 52
- trade\_flag
  - XTPTickByTickTrade, 125
- TraderApi, 20
- TraderSpi, 53
- XTP::API::TraderApi
  - CancelOptionCombinedOrder, 22
  - CancelOrder, 23
  - CreateTraderApi, 24
  - CreditCashRepay, 24
  - CreditCashRepayDebtInterestFee, 25
  - CreditExtendDebtDate, 25
  - CreditSellStockRepayDebtInterestFee, 26
  - FundTransfer, 26
  - GetAccountByXTPID, 27
  - GetApiLastError, 27
  - GetApiVersion, 27
  - GetClientIDByXTPID, 28
  - GetTradingDay, 28
  - InsertOptionCombinedOrder, 28
  - InsertOrder, 29
  - IsServerRestart, 30
  - Login, 30
  - Logout, 31
  - ModifyUserTerminalInfo, 31
  - QueryAsset, 32
  - QueryCreditAssetDebtInfo, 32
  - QueryCreditCashRepayInfo, 33
  - QueryCreditDebtInfo, 33
  - QueryCreditExcessStock, 34
  - QueryCreditExtendDebtDateOrders, 34
  - QueryCreditFundExtraInfo, 35
  - QueryCreditFundInfo, 35
  - QueryCreditPositionExtraInfo, 35
  - QueryCreditTickerAssignInfo, 36
  - QueryCreditTickerDebtInfo, 36
  - QueryETFTickerBasket, 38
  - QueryETF, 38
  - QueryFundTransfer, 39
  - QueryIPOInfoList, 39
  - QueryIPOQuotaInfo, 39
  - QueryMulCreditExcessStock, 40
  - QueryOptionAuctionInfo, 40
  - QueryOptionCombinedOrderByXTPID, 41
  - QueryOptionCombinedOrders, 41
  - QueryOptionCombinedOrdersByPage, 42
  - QueryOptionCombinedPosition, 42
  - QueryOptionCombinedStrategyInfo, 43
  - QueryOptionCombinedTrades, 44
  - QueryOptionCombinedTradesByPage, 44
  - QueryOptionCombinedTradesByXTPID, 45
  - QueryOptionCombinedUnfinishedOrders, 45
  - QueryOrderByXTPID, 46
  - QueryOrders, 46
  - QueryOrdersByPage, 47
  - QueryPosition, 47
  - QueryStructuredFund, 48
  - QueryTrades, 48
  - QueryTradesByPage, 49
  - QueryTradesByXTPID, 50
  - QueryUnfinishedOrders, 50
  - RegisterSpi, 51
  - Release, 51
  - SetHeartBeatInterval, 51
  - SetSoftwareKey, 51
  - SetSoftwareVersion, 52
  - SubscribePublicTopic, 52
- XTP::API::TraderSpi
  - OnCancelOptionCombinedOrderError, 55
  - OnCancelOrderError, 55
  - OnCreditCashRepay, 56
  - OnCreditCashRepayDebtInterestFee, 56
  - OnCreditExtendDebtDate, 57
  - OnDisconnected, 57
  - OnError, 58
  - OnFundTransfer, 58
  - OnOptionCombinedOrderEvent, 59
  - OnOptionCombinedTradeEvent, 59
  - OnOrderEvent, 60
  - OnQueryAsset, 60
  - OnQueryCreditAssetDebtInfo, 61
  - OnQueryCreditCashRepayInfo, 61

- OnQueryCreditDebtInfo, 62
- OnQueryCreditExcessStock, 62
- OnQueryCreditExtendDebtDateOrders, 63
- OnQueryCreditFundExtraInfo, 64
- OnQueryCreditFundInfo, 64
- OnQueryCreditPositionExtraInfo, 65
- OnQueryCreditTickerAssignInfo, 65
- OnQueryCreditTickerDebtInfo, 66
- OnQueryETFBasket, 67
- OnQueryETF, 66
- OnQueryFundTransfer, 68
- OnQueryIPOInfoList, 68
- OnQueryIPOQuotaInfo, 69
- OnQueryMulCreditExcessStock, 69
- OnQueryOptionAuctionInfo, 70
- OnQueryOptionCombinedOrders, 71
- OnQueryOptionCombinedOrdersByPage, 71
- OnQueryOptionCombinedPosition, 72
- OnQueryOptionCombinedStrategyInfo, 72
- OnQueryOptionCombinedTrades, 73
- OnQueryOptionCombinedTradesByPage, 74
- OnQueryOrder, 74
- OnQueryOrderByPage, 75
- OnQueryPosition, 76
- OnQueryStructuredFund, 76
- OnQueryTrade, 77
- OnQueryTradeByPage, 77
- OnTradeEvent, 78
- XTP\_ACCOUNT\_TYPE
  - xtp\_api\_data\_type.h, 145
- XTP\_AUTO\_SPLIT\_TYPE
  - xtp\_api\_data\_type.h, 145
- XTP\_BUSINESS\_TYPE
  - xtp\_api\_data\_type.h, 145
- XTP\_CRD\_CR\_STATUS
  - xtp\_api\_data\_type.h, 146
- XTP\_DEBT\_EXTEND\_OPER\_STATUS
  - xtp\_api\_data\_type.h, 146
- XTP\_EXCHANGE\_TYPE
  - xtp\_api\_data\_type.h, 147
- XTP\_EXPIRE\_DATE\_TYPE
  - xtp\_api\_data\_type.h, 147
- XTP\_FUND\_OPER\_STATUS
  - xtp\_api\_data\_type.h, 147
- XTP\_FUND\_TRANSFER\_TYPE
  - xtp\_api\_data\_type.h, 148
- XTP\_LOG\_LEVEL
  - xtp\_api\_data\_type.h, 148
- XTP\_MARKET\_TYPE
  - xtp\_api\_data\_type.h, 148
- XTP\_OPT\_CALL\_OR\_PUT\_TYPE
  - xtp\_api\_data\_type.h, 149
- XTP\_OPT\_COVERED\_OR\_UNCOVERED
  - xtp\_api\_data\_type.h, 149
- XTP\_OPT\_EXERCISE\_TYPE\_TYPE
  - xtp\_api\_data\_type.h, 149
- XTP\_OPT\_POSITION\_TYPE
  - xtp\_api\_data\_type.h, 150
- XTP\_ORDER\_ACTION\_STATUS\_TYPE
  - xtp\_api\_data\_type.h, 150
- XTP\_ORDER\_DETAIL\_TYPE
  - xtp\_api\_data\_type.h, 150
- XTP\_ORDER\_STATUS\_TYPE
  - xtp\_api\_data\_type.h, 151
- XTP\_ORDER\_SUBMIT\_STATUS\_TYPE
  - xtp\_api\_data\_type.h, 151
- XTP\_POSITION\_DIRECTION\_TYPE
  - xtp\_api\_data\_type.h, 151
- XTP\_PRICE\_TYPE
  - xtp\_api\_data\_type.h, 152
- XTP\_PROTOCOL\_TYPE
  - xtp\_api\_data\_type.h, 152
- XTP\_QUALIFICATION\_TYPE
  - xtp\_api\_data\_type.h, 152
- XTP\_SECURITY\_TYPE
  - xtp\_api\_data\_type.h, 153
- XTP\_SIDE\_STOCK\_REPAY\_STOCK
  - xtp\_api\_data\_type.h, 144
- XTP\_SPLIT\_MERGE\_STATUS
  - xtp\_api\_data\_type.h, 154
- XTP\_TBT\_TYPE
  - xtp\_api\_data\_type.h, 154
- XTP\_TE\_RESUME\_TYPE
  - xtp\_api\_data\_type.h, 154
- XTP\_TICKER\_TYPE
  - xtp\_api\_data\_type.h, 154
- XTP\_UNDERLYING\_TYPE
  - xtp\_api\_data\_type.h, 155
- XTPClientQueryCrdDebtStockReq, 79
- XTPClientQueryCrdPositionStkInfo, 79
- XTPClientQueryCrdPositionStockReq, 80
- XTPClientQueryCrdSurplusStkReqInfo, 81
- XTPClientQueryCrdSurplusStkRspInfo, 81
- XTPCombLegStrategy, 82
- XTPCrdCashRepayDebtInterestFeeRsp, 82
- XTPCrdCashRepayInfo, 83
- XTPCrdCashRepayRsp, 83
- XTPCrdDebtInfo, 84
- XTPCrdDebtStockInfo, 85
- XTPCrdFundExtraInfo, 86
- XTPCrdFundInfo, 86
- XTPCrdPositionExtraInfo, 87
- XTPCreditDebtExtendNotice, 87
- XTPCreditDebtExtendReq, 88
- XTPFundTransferNotice, 88
- XTPFundTransferReq, 89
- XTPMarketDataOptionExData, 90
- XTPMarketDataStockExData, 90
- XTPMarketDataStruct, 92
- XTPOptCombLegInfo, 93
- XTPOptCombOrderInfo, 94
  - market, 95
- XTPOptCombOrderInsertInfo, 95
- XTPOptCombPlugin, 96
- XTPOptCombTradeReport, 97
- XTPOrderCancelInfo, 98



XTPOrderInfo, 98  
XTPOrderInsertInfo, 99  
XTPQueryAssetRsp, 100  
XTPQueryCombineStrategyInfoRsp, 102  
XTPQueryETFBaseReq, 102  
XTPQueryETFBaseRsp, 103  
XTPQueryETFComponentReq, 104  
XTPQueryETFComponentRsp, 104  
XTPQueryETFComponentRspV1, 105  
XTPQueryFundTransferLogReq, 106  
XTPQueryIPOQuotaRsp, 107  
XTPQueryIPOQuotaRspV1, 107  
XTPQueryIPOTickerRsp, 108  
XTPQueryOptCombOrderByPageReq, 108  
XTPQueryOptCombOrderReq, 109  
XTPQueryOptCombPositionReq, 109  
XTPQueryOptCombPositionRsp, 110  
XTPQueryOptCombReportByExeclIdReq, 110  
XTPQueryOptCombTraderByPageReq, 111  
XTPQueryOptCombTraderReq, 111  
XTPQueryOptionAuctionInfoReq, 112  
XTPQueryOptionAuctionInfoRsp, 112  
XTPQueryOrderByPageReq, 114  
XTPQueryOrderReq, 115  
XTPQueryReportByExeclIdReq, 115  
XTPQueryStkPositionReq, 116  
XTPQueryStkPositionRsp, 116  
XTPQueryStructuredFundInfoReq, 117  
XTPQueryTraderByPageReq, 118  
XTPQueryTraderReq, 118  
XTPQuoteFullInfo, 119  
XTPQuoteStaticInfo, 120  
XTPRspInfoStruct, 121  
XTPSpecificTickerStruct, 122  
XTPStructuredFundInfo, 122  
XTPTerminalType  
    xtp\_api\_data\_type.h, 155  
XTPTickByTickEntrust, 123  
XTPTickByTickStruct, 124  
XTPTickByTickTrade, 124  
    trade\_flag, 125  
XTPTickerPriceInfo, 125  
XTPTradeReport, 126  
XTPUserTerminalInfoReq, 127  
xoms\_api\_fund\_struct.h, 132  
xoms\_api\_struct.h, 132  
xquote\_api\_struct.h, 136  
xtp\_api\_data\_type.h, 137  
    ETF\_REPLACE\_TYPE, 144  
    XTP\_ACCOUNT\_TYPE, 145  
    XTP\_AUTO\_SPLIT\_TYPE, 145  
    XTP\_BUSINESS\_TYPE, 145  
    XTP\_CRD\_CR\_STATUS, 146  
    XTP\_DEBT\_EXTEND\_OPER\_STATUS, 146  
    XTP\_EXCHANGE\_TYPE, 147  
    XTP\_EXPIRE\_DATE\_TYPE, 147  
    XTP\_FUND\_OPER\_STATUS, 147  
    XTP\_FUND\_TRANSFER\_TYPE, 148  
    XTP\_LOG\_LEVEL, 148  
    XTP\_MARKET\_TYPE, 148  
    XTP\_OPT\_CALL\_OR\_PUT\_TYPE, 149  
    XTP\_OPT\_COVERED\_OR\_UNCOVERED, 149  
    XTP\_OPT\_EXERCISE\_TYPE\_TYPE, 149  
    XTP\_OPT\_POSITION\_TYPE, 150  
    XTP\_ORDER\_ACTION\_STATUS\_TYPE, 150  
    XTP\_ORDER\_DETAIL\_TYPE, 150  
    XTP\_ORDER\_STATUS\_TYPE, 151  
    XTP\_ORDER\_SUBMIT\_STATUS\_TYPE, 151  
    XTP\_POSITION\_DIRECTION\_TYPE, 151  
    XTP\_PRICE\_TYPE, 152  
    XTP\_PROTOCOL\_TYPE, 152  
    XTP\_QUALIFICATION\_TYPE, 152  
    XTP\_SECURITY\_TYPE, 153  
    XTP\_SIDE\_STOCK\_REPAY\_STOCK, 144  
    XTP\_SPLIT\_MERGE\_STATUS, 154  
    XTP\_TBT\_TYPE, 154  
    XTP\_TE\_RESUME\_TYPE, 154  
    XTP\_TICKER\_TYPE, 154  
    XTP\_UNDERLYING\_TYPE, 155  
XTPTerminalType, 155  
xtp\_api\_struct.h, 156  
xtp\_api\_struct\_common.h, 156  
xtp\_trader\_api.h, 157