

# XTP极速交易系统QuoteAPI

制作者 中泰证券股份有限公司

2018年 六月 28日 星期四 17:37:59



# Contents

<b>1</b>	<b>XTP 极速行情交易系统 Quote API 1.1.18.13</b>	<b>1</b>
<b>2</b>	<b>继承关系索引</b>	<b>3</b>
2.1	类继承关系	3
<b>3</b>	<b>结构体索引</b>	<b>5</b>
3.1	结构体	5
<b>4</b>	<b>文件索引</b>	<b>7</b>
4.1	文件列表	7
<b>5</b>	<b>结构体说明</b>	<b>9</b>
5.1	DemoTestMdSpi类 参考	9
5.1.1	详细描述	10
5.2	OrderBookStruct结构体 参考	10
5.2.1	详细描述	11
5.3	QuoteApi类 参考	11
5.3.1	详细描述	12
5.3.2	成员函数说明	12
5.3.2.1	CreateQuoteApi(uint8_t client_id, const char *save_file_path, XTP_LOG_LEVEL log_level=XTP_LOG_LEVEL_DEBUG)	12
5.3.2.2	GetApiLastError()=0	13
5.3.2.3	GetApiVersion()=0	13
5.3.2.4	GetTradingDay()=0	13
5.3.2.5	Login(const char *ip, int port, const char *user, const char *password, XTP_PROTOCOL_TYPE sock_type)=0	14
5.3.2.6	Logout()=0	14
5.3.2.7	QueryAllTickers(XTP_EXCHANGE_TYPE exchange_id)=0	14
5.3.2.8	QueryAllTickersPriceInfo()=0	14
5.3.2.9	QueryTickersPriceInfo(char *ticker[], int count, XTP_EXCHANGE_TYPE exchange_id)=0	15
5.3.2.10	RegisterSpi(QuoteSpi *spi)=0	15
5.3.2.11	Release()=0	15
5.3.2.12	SetHeartBeatInterval(uint32_t interval)=0	15

5.3.2.13	SetUDPBufferSize(uint32_t buff_size)=0	15
5.3.2.14	SubscribeAllMarketData(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	16
5.3.2.15	SubscribeAllOptionMarketData(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	16
5.3.2.16	SubscribeAllOptionOrderBook(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	16
5.3.2.17	SubscribeAllOptionTickByTick(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	17
5.3.2.18	SubscribeAllOrderBook(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	17
5.3.2.19	SubscribeAllTickByTick(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	17
5.3.2.20	SubscribeMarketData(char *ticker[], int count, XTP_EXCHANGE_TYPE exchange_id)=0	18
5.3.2.21	SubscribeOrderBook(char *ticker[], int count, XTP_EXCHANGE_TYPE exchange_id)=0	18
5.3.2.22	SubscribeTickByTick(char *ticker[], int count, XTP_EXCHANGE_TYPE exchange_id)=0	18
5.3.2.23	UnSubscribeAllMarketData(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	19
5.3.2.24	UnSubscribeAllOptionMarketData(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	19
5.3.2.25	UnSubscribeAllOptionOrderBook(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	19
5.3.2.26	UnSubscribeAllOptionTickByTick(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	20
5.3.2.27	UnSubscribeAllOrderBook(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	20
5.3.2.28	UnSubscribeAllTickByTick(XTP_EXCHANGE_TYPE exchange_id=XTP_EXCHANGE_UNKNOWN)=0	20
5.3.2.29	UnSubscribeMarketData(char *ticker[], int count, XTP_EXCHANGE_TYPE exchange_id)=0	21
5.3.2.30	UnSubscribeOrderBook(char *ticker[], int count, XTP_EXCHANGE_TYPE exchange_id)=0	21
5.3.2.31	UnSubscribeTickByTick(char *ticker[], int count, XTP_EXCHANGE_TYPE exchange_id)=0	21
5.4	QuoteSpi类 参考	22
5.4.1	详细描述	23
5.4.2	成员函数说明	23
5.4.2.1	OnDepthMarketData(XTPMD *market_data, int64_t bid1_qty[], int32_t bid1_count, int32_t max_bid1_count, int64_t ask1_qty[], int32_t ask1_count, int32_t max_ask1_count)	23
5.4.2.2	OnDisconnected(int reason)	23
5.4.2.3	OnError(XTPRI *error_info)	23
5.4.2.4	OnOrderBook(XTPOB *order_book)	24

5.4.2.5	OnQueryAllTickers(XTPQSI *ticker_info, XTPRI *error_info, bool is_last)	24
5.4.2.6	OnQueryTickersPriceInfo(XTPTPPI *ticker_info, XTPRI *error_info, bool is_last)	24
5.4.2.7	OnSubMarketData(XTPST *ticker, XTPRI *error_info, bool is_last)	24
5.4.2.8	OnSubOrderBook(XTPST *ticker, XTPRI *error_info, bool is_last)	25
5.4.2.9	OnSubscribeAllMarketData(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	25
5.4.2.10	OnSubscribeAllOptionMarketData(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	25
5.4.2.11	OnSubscribeAllOptionOrderBook(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	26
5.4.2.12	OnSubscribeAllOptionTickByTick(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	26
5.4.2.13	OnSubscribeAllOrderBook(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	26
5.4.2.14	OnSubscribeAllTickByTick(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	27
5.4.2.15	OnSubTickByTick(XTPST *ticker, XTPRI *error_info, bool is_last)	27
5.4.2.16	OnTickByTick(XTPBT *tbt_data)	27
5.4.2.17	OnUnSubMarketData(XTPST *ticker, XTPRI *error_info, bool is_last)	28
5.4.2.18	OnUnSubOrderBook(XTPST *ticker, XTPRI *error_info, bool is_last)	28
5.4.2.19	OnUnSubscribeAllMarketData(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	28
5.4.2.20	OnUnSubscribeAllOptionMarketData(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	29
5.4.2.21	OnUnSubscribeAllOptionOrderBook(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	29
5.4.2.22	OnUnSubscribeAllOptionTickByTick(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	29
5.4.2.23	OnUnSubscribeAllOrderBook(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	30
5.4.2.24	OnUnSubscribeAllTickByTick(XTP_EXCHANGE_TYPE exchange_id, XTPRI *error_info)	30
5.4.2.25	OnUnSubTickByTick(XTPST *ticker, XTPRI *error_info, bool is_last)	30
5.5	XTPFundTransferNotice结构体 参考	31
5.5.1	详细描述	31
5.6	XTPFundTransferReq结构体 参考	31
5.6.1	详细描述	32
5.7	XTPMarketDataOptionExData结构体 参考	32
5.7.1	详细描述	32
5.8	XTPMarketDataStockExData结构体 参考	32
5.8.1	详细描述	34
5.9	XTPMarketDataStruct结构体 参考	34
5.9.1	详细描述	35
5.10	XTPOrderCancelInfo结构体 参考	36

5.10.1 详细描述	36
5.11 XTPOrderInfo结构体 参考	36
5.11.1 详细描述	37
5.12 XTPOrderInsertInfo结构体 参考	37
5.12.1 详细描述	38
5.13 XTPQueryAssetRsp结构体 参考	38
5.13.1 详细描述	39
5.14 XTPQueryETFBaseReq结构体 参考	39
5.14.1 详细描述	40
5.15 XTPQueryETFBaseRsp结构体 参考	40
5.15.1 详细描述	40
5.16 XTPQueryETFComponentReq结构体 参考	41
5.16.1 详细描述	41
5.17 XTPQueryETFComponentRsp结构体 参考	41
5.17.1 详细描述	42
5.18 XTPQueryFundTransferLogReq结构体 参考	42
5.18.1 详细描述	42
5.19 XTPQueryIPOQuotaRsp结构体 参考	42
5.19.1 详细描述	42
5.20 XTPQueryIPOTickerRsp结构体 参考	42
5.20.1 详细描述	43
5.21 XTPQueryOptionAuctionInfoReq结构体 参考	43
5.21.1 详细描述	43
5.22 XTPQueryOptionAuctionInfoRsp结构体 参考	43
5.22.1 详细描述	45
5.23 XTPQueryOrderReq结构体 参考	45
5.23.1 详细描述	46
5.24 XTPQueryReportByExecIdReq结构体 参考	46
5.24.1 详细描述	46
5.25 XTPQueryStkPositionRsp结构体 参考	46
5.25.1 详细描述	47
5.26 XTPQueryStructuredFundInfoReq结构体 参考	47
5.26.1 详细描述	47
5.27 XTPQueryTraderReq结构体 参考	48
5.27.1 详细描述	48
5.28 XTPQuoteStaticInfo结构体 参考	48
5.28.1 详细描述	49
5.29 XTPRspInfoStruct结构体 参考	49
5.29.1 详细描述	49
5.30 XTPSpecificTickerStruct结构体 参考	49

5.30.1 详细描述	49
5.31 XTPStructuredFundInfo结构体 参考	50
5.31.1 详细描述	50
5.32 XTPTickByTickEntrust结构体 参考	50
5.32.1 详细描述	51
5.33 XTPTickByTickStruct结构体 参考	51
5.33.1 详细描述	51
5.34 XTPTickByTickTrade结构体 参考	51
5.34.1 详细描述	52
5.34.2 结构体成员变量说明	52
5.34.2.1 trade_flag	52
5.35 XTPTickerPricelInfo结构体 参考	52
5.35.1 详细描述	52
5.36 XTPTradeReport结构体 参考	53
5.36.1 详细描述	54
<b>6 文件说明</b>	<b>55</b>
6.1 demo_test_quote_api.cpp 文件参考	55
6.1.1 详细描述	55
6.1.2 函数说明	56
6.1.2.1 main()	56
6.1.3 变量说明	56
6.1.3.1 ticker_count	56
6.2 demo_test_quote_spi.h 文件参考	56
6.2.1 详细描述	56
6.3 xoms_api_fund_struct.h 文件参考	57
6.3.1 详细描述	57
6.4 xoms_api_struct.h 文件参考	57
6.4.1 详细描述	59
6.5 xquote_api_struct.h 文件参考	59
6.5.1 详细描述	60
6.5.2 枚举类型说明	60
6.5.2.1 XTP_MARKETDATA_TYPE	60
6.6 xtp_api_data_type.h 文件参考	60
6.6.1 详细描述	64
6.6.2 枚举类型说明	65
6.6.2.1 ETF_REPLACE_TYPE	65
6.6.2.2 XTP_ACCOUNT_TYPE	65
6.6.2.3 XTP_BUSINESS_TYPE	65
6.6.2.4 XTP_EXCHANGE_TYPE	66

6.6.2.5	XTP_FUND_OPER_STATUS . . . . .	66
6.6.2.6	XTP_FUND_TRANSFER_TYPE . . . . .	66
6.6.2.7	XTP_LOG_LEVEL . . . . .	66
6.6.2.8	XTP_MARKET_TYPE . . . . .	67
6.6.2.9	XTP_OPT_CALL_OR_PUT_TYPE . . . . .	67
6.6.2.10	XTP_OPT_EXERCISE_TYPE_TYPE . . . . .	67
6.6.2.11	XTP_ORDER_ACTION_STATUS_TYPE . . . . .	67
6.6.2.12	XTP_ORDER_STATUS_TYPE . . . . .	67
6.6.2.13	XTP_ORDER_SUBMIT_STATUS_TYPE . . . . .	68
6.6.2.14	XTP_POSITION_DIRECTION_TYPE . . . . .	68
6.6.2.15	XTP_PRICE_TYPE . . . . .	68
6.6.2.16	XTP_PROTOCOL_TYPE . . . . .	68
6.6.2.17	XTP_SPLIT_MERGE_STATUS . . . . .	69
6.6.2.18	XTP_TBT_TYPE . . . . .	69
6.6.2.19	XTP_TE_RESUME_TYPE . . . . .	69
6.6.2.20	XTP_TICKER_TYPE . . . . .	69
6.7	xtp_api_struct.h 文件参考 . . . . .	69
6.7.1	详细描述 . . . . .	70
6.8	xtp_api_struct_common.h 文件参考 . . . . .	70
6.8.1	详细描述 . . . . .	70
6.9	xtp_quote_api.h 文件参考 . . . . .	70
6.9.1	详细描述 . . . . .	71
索引		73



## Chapter 1

# XTP 极速行情交易系统 Quote API 1.1.18.13

本项目是XTP项目中的行情类接口

(1) XTP的行情订阅接口和响应类 [xtp\\_quote\\_api.h](#)

(2) 行情订阅测试Demo [demo\\_test\\_quote\\_api.cpp](#)



## Chapter 2

# 继承关系索引

## 2.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

OrderBookStruct . . . . .	10
QuoteApi . . . . .	11
QuoteSpi . . . . .	22
DemoTestMdSpi . . . . .	9
XTPFundTransferNotice . . . . .	31
XTPFundTransferReq . . . . .	31
XTPMarketDataOptionExData . . . . .	32
XTPMarketDataStockExData . . . . .	32
XTPMarketDataStruct . . . . .	34
XTPOrderCancelInfo . . . . .	36
XTPOrderInfo . . . . .	36
XTPOrderInsertInfo . . . . .	37
XTPQueryAssetRsp . . . . .	38
XTPQueryETFBaseReq . . . . .	39
XTPQueryETFBaseRsp . . . . .	40
XTPQueryETFComponentReq . . . . .	41
XTPQueryETFComponentRsp . . . . .	41
XTPQueryFundTransferLogReq . . . . .	42
XTPQueryIPOQuotaRsp . . . . .	42
XTPQueryIPOTickerRsp . . . . .	42
XTPQueryOptionAuctionInfoReq . . . . .	43
XTPQueryOptionAuctionInfoRsp . . . . .	43
XTPQueryOrderReq . . . . .	45
XTPQueryReportByExecIdReq . . . . .	46
XTPQueryStkPositionRsp . . . . .	46
XTPQueryStructuredFundInfoReq . . . . .	47
XTPQueryTraderReq . . . . .	48
XTPQuoteStaticInfo . . . . .	48
XTPRspInfoStruct . . . . .	49
XTPSpecificTickerStruct . . . . .	49
XTPStructuredFundInfo . . . . .	50
XTPTickByTickEntrust . . . . .	50
XTPTickByTickStruct . . . . .	51
XTPTickByTickTrade . . . . .	51
XTPTickerPriceInfo . . . . .	52
XTPTradeReport . . . . .	53



## Chapter 3

# 结构体索引

### 3.1 结构体

这里列出了所有结构体，并附带简要说明：

DemoTestMdSpi	
Demo自定义行情订阅接口响应类	9
OrderBookStruct	
定单簿	10
QuoteApi	
行情订阅接口类	11
QuoteSpi	
行情回调类	22
XTPFundTransferNotice	
资金内转流水通知	31
XTPFundTransferReq	
用户资金请求	31
XTPMarketDataOptionExData	
期权额外数据	32
XTPMarketDataStockExData	
股票、基金、债券等额外数据	32
XTPMarketDataStruct	
行情	34
XTPOrderCancelInfo	
撤单失败响应消息	36
XTPOrderInfo	
报单响应结构体	36
XTPOrderInsertInfo	
新订单请求	37
XTPQueryAssetRsp	
账户资金查询响应结构体	38
XTPQueryETFBaseReq	
查询股票ETF合约基本情况-请求结构体,请求参数为:交易市场+ETF买卖代码	39
XTPQueryETFBaseRsp	
查询股票ETF合约基本情况-响应结构体	40
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	41
XTPQueryETFComponentRsp	
查询股票ETF合约成分股信息-响应结构体	41
XTPQueryFundTransferLogReq	
资金内转流水查询请求与响应	42
XTPQueryIPOQuotaRsp	
查询用户申购额度	42

<a href="#">XTPQueryIPOTickerRsp</a>	
查询当日可申购新股信息	42
<a href="#">XTPQueryOptionAuctionInfoReq</a>	
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	43
<a href="#">XTPQueryOptionAuctionInfoRsp</a>	
查询期权竞价交易业务参考信息	43
<a href="#">XTPQueryOrderReq</a>	
报单查询 // 报单查询请求-条件查询	45
<a href="#">XTPQueryReportByExecIdReq</a>	
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)	46
<a href="#">XTPQueryStkPositionRsp</a>	
查询股票持仓情况	46
<a href="#">XTPQueryStructuredFundInfoReq</a>	
查询分级基金信息结构体	47
<a href="#">XTPQueryTraderReq</a>	
查询成交回报请求-查询条件	48
<a href="#">XTPQuoteStaticInfo</a>	
股票行情静态信息	48
<a href="#">XTPRspInfoStruct</a>	
响应信息	49
<a href="#">XTPSpecificTickerStruct</a>	
指定的合约	49
<a href="#">XTPStructuredFundInfo</a>	
查询分级基金信息响应结构体	50
<a href="#">XTPTickByTickEntrust</a>	
逐笔委托(仅适用深交所)	50
<a href="#">XTPTickByTickStruct</a>	
逐笔数据信息	51
<a href="#">XTPTickByTickTrade</a>	
逐笔成交	51
<a href="#">XTPTickerPriceInfo</a>	
供查询的最新信息	52
<a href="#">XTPTradeReport</a>	
报单成交结构体	53

## 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：

<a href="#">demo_test_quote_api.cpp</a>	
定义控制台测试应用程序的入口点 . . . . .	55
<a href="#">demo_test_quote_spi.h</a>	
Demo自定义客户端行情订阅响应接口类 . . . . .	56
<a href="#">xoms_api_fund_struct.h</a>	
定义资金划拨相关结构体类型 . . . . .	57
<a href="#">xoms_api_struct.h</a>	
定义交易类相关数据结构 . . . . .	57
<a href="#">xquote_api_struct.h</a>	
定义行情类相关数据结构 . . . . .	59
<a href="#">xtp_api_data_type.h</a>	
定义兼容数据基本类型 . . . . .	60
<a href="#">xtp_api_struct.h</a>	
定义业务数据结构 . . . . .	69
<a href="#">xtp_api_struct_common.h</a>	
定义业务公共数据结构 . . . . .	70
<a href="#">xtp_quote_api.h</a>	
定义行情订阅客户端接口 . . . . .	70





## Chapter 5

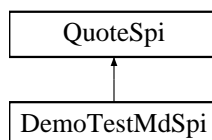
# 结构体说明

### 5.1 DemoTestMdSpi类 参考

Demo自定义行情订阅接口响应类

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

类 DemoTestMdSpi 继承关系图:



#### Public 成员函数

- virtual void **OnDisconnected** (int reason)  
当客户端与行情后台通信连接断开
- virtual void **OnError** (XTPRI \*error\_info)  
错误应答
- virtual void **OnSubMarketData** (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)  
重写订阅行情应答
- virtual void **OnUnSubMarketData** (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)  
重写取消订阅行情应答
- virtual void **OnDepthMarketData** (XTPMD \*market\_data, int64\_t bid1\_qty[], int32\_t bid1\_count, int32\_t max\_bid1\_count, int64\_t ask1\_qty[], int32\_t ask1\_count, int32\_t max\_ask1\_count)  
重写行情通知
- virtual void **OnSubOrderBook** (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)  
重写订阅行情订单簿应答
- virtual void **OnUnSubOrderBook** (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)  
重写退订行情订单簿应答
- virtual void **OnSubTickByTick** (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)  
订阅逐笔行情应答
- virtual void **OnUnSubTickByTick** (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)  
退订逐笔行情应答
- virtual void **OnOrderBook** (XTPOB \*order\_book)  
行情订单簿通知
- virtual void **OnTickByTick** (XTPTBT \*tbt\_data)

- 逐笔行情通知，包括股票、指数和期权
- virtual void **OnQueryAllTickers** (XTPQSI \*ticker\_info, XTPRI \*error\_info, bool is\_last)  
查询可交易合约的应答
  - virtual void **OnQueryTickersPriceInfo** (XTPPTPI \*ticker\_info, XTPRI \*error\_info, bool is\_last)  
查询合约的最新价格信息应答
  - virtual void **OnSubscribeAllMarketData** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
订阅全市场的股票行情应答
  - virtual void **OnUnSubscribeAllMarketData** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
退订全市场的股票行情应答
  - virtual void **OnSubscribeAllOrderBook** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
订阅全市场的股票行情订单簿应答
  - virtual void **OnUnSubscribeAllOrderBook** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
退订全市场的股票行情订单簿应答
  - virtual void **OnSubscribeAllTickByTick** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
订阅全市场的股票逐笔行情应答
  - virtual void **OnUnSubscribeAllTickByTick** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
退订全市场的股票逐笔行情应答
  - virtual void **OnSubscribeAllOptionMarketData** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
订阅全市场的期权行情应答
  - virtual void **OnUnSubscribeAllOptionMarketData** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
退订全市场的期权行情应答
  - virtual void **OnSubscribeAllOptionOrderBook** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
订阅全市场的期权行情订单簿应答
  - virtual void **OnUnSubscribeAllOptionOrderBook** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
退订全市场的期权行情订单簿应答
  - virtual void **OnSubscribeAllOptionTickByTick** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
订阅全市场的期权逐笔行情应答
  - virtual void **OnUnSubscribeAllOptionTickByTick** (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)  
退订全市场的期权逐笔行情应答

### 5.1.1 详细描述

Demo自定义行情订阅接口响应类

作者

中泰证券股份有限公司

日期

十月 2015

该类的文档由以下文件生成:

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

## 5.2 OrderBookStruct结构体 参考

定单簿

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

## 成员变量

- [XTP\\_EXCHANGE\\_TYPE exchange\\_id](#)  
交易所代码
- [char ticker \[XTP\\_TICKER\\_LEN\]](#)  
合约代码（不包含交易所信息），不带空格，以'\0'结尾
- [double last\\_price](#)  
最新价
- [int64\\_t qty](#)  
数量，为总成交量
- [double turnover](#)  
成交金额，为总成交金额
- [int64\\_t trades\\_count](#)  
成交笔数
- [double bid \[10\]](#)  
十档申买价
- [double ask \[10\]](#)  
十档申卖价
- [int64\\_t bid\\_qty \[10\]](#)  
十档申买量
- [int64\\_t ask\\_qty \[10\]](#)  
十档申卖量
- [int64\\_t data\\_time](#)  
时间类

## 5.2.1 详细描述

定单簿

该结构体的文档由以下文件生成:

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

## 5.3 QuoteApi类 参考

行情订阅接口类

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

## Public 成员函数

- [virtual void Release \(\)=0](#)
- [virtual const char \\* GetTradingDay \(\)=0](#)
- [virtual const char \\* GetApiVersion \(\)=0](#)
- [virtual XTPRI \\* GetApiLastError \(\)=0](#)
- [virtual void SetUDPBufferSize \(uint32\\_t buff\\_size\)=0](#)
- [virtual void RegisterSpi \(QuoteSpi \\*spi\)=0](#)
- [virtual void SetHeartBeatInterval \(uint32\\_t interval\)=0](#)
- [virtual int SubscribeMarketData \(char \\*ticker\[\], int count, XTP\\_EXCHANGE\\_TYPE exchange\\_id\)=0](#)
- [virtual int UnSubscribeMarketData \(char \\*ticker\[\], int count, XTP\\_EXCHANGE\\_TYPE exchange\\_id\)=0](#)
- [virtual int SubscribeOrderBook \(char \\*ticker\[\], int count, XTP\\_EXCHANGE\\_TYPE exchange\\_id\)=0](#)
- [virtual int UnSubscribeOrderBook \(char \\*ticker\[\], int count, XTP\\_EXCHANGE\\_TYPE exchange\\_id\)=0](#)

- virtual int [SubscribeTickByTick](#) (char \*ticker[], int count, [XTP\\_EXCHANGE\\_TYPE](#) exchange\_id)=0
- virtual int [UnSubscribeTickByTick](#) (char \*ticker[], int count, [XTP\\_EXCHANGE\\_TYPE](#) exchange\_id)=0
- virtual int [SubscribeAllMarketData](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_UNKNOW↵N](#))=0
- virtual int [UnSubscribeAllMarketData](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_UNKNOW↵N](#))=0
- virtual int [SubscribeAllOrderBook](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_UNKNOW↵N](#))=0
- virtual int [UnSubscribeAllOrderBook](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_UNKNOW↵N](#))=0
- virtual int [SubscribeAllTickByTick](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_UNKNOW↵N](#))=0
- virtual int [UnSubscribeAllTickByTick](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_UNKNOW↵N](#))=0
- virtual int [Login](#) (const char \*ip, int port, const char \*user, const char \*password, [XTP\\_PROTOCOL\\_TYPE](#) sock\_type)=0
- virtual int [Logout](#) ()=0
- virtual int [QueryAllTickers](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id)=0
- virtual int [QueryTickersPriceInfo](#) (char \*ticker[], int count, [XTP\\_EXCHANGE\\_TYPE](#) exchange\_id)=0
- virtual int [QueryAllTickersPriceInfo](#) ()=0
- virtual int [SubscribeAllOptionMarketData](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_UN↵KNOWN](#))=0
- virtual int [UnSubscribeAllOptionMarketData](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_↵UNKNOWN](#))=0
- virtual int [SubscribeAllOptionOrderBook](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_UNK↵NOWN](#))=0
- virtual int [UnSubscribeAllOptionOrderBook](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_U↵NKNOWN](#))=0
- virtual int [SubscribeAllOptionTickByTick](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_UNK↵NOWN](#))=0
- virtual int [UnSubscribeAllOptionTickByTick](#) ([XTP\\_EXCHANGE\\_TYPE](#) exchange\_id=[XTP\\_EXCHANGE\\_U↵NKNOWN](#))=0

## 静态 **Public** 成员函数

- static [QuoteApi](#) \* [CreateQuoteApi](#) (uint8\_t client\_id, const char \*save\_file\_path, [XTP\\_LOG\\_LEVEL](#) log\_↵level=[XTP\\_LOG\\_LEVEL\\_DEBUG](#))

### 5.3.1 详细描述

行情订阅接口类

作者

中泰证券股份有限公司

日期

十月 2015

### 5.3.2 成员函数说明

- #### 5.3.2.1 static [QuoteApi](#)\* [CreateQuoteApi](#) ( uint8\_t client\_id, const char \* save\_file\_path, [XTP\\_LOG\\_LEVEL](#) log\_level = [XTP\\_LOG\\_LEVEL\\_DEBUG](#) ) [static]

创建[QuoteApi](#)

参数

<i>client_id</i>	(必须输入) 用于区分同一用户的不同客户端, 由用户自定义
<i>save_file_path</i>	(必须输入) 存贮订阅信息文件的目录, 请设定一个有可写权限的真实存在的路径
<i>log_level</i>	日志输出级别

返回

创建出的UserApi

备注

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

#### 5.3.2.2 virtual XTPRI\* GetApiLastError ( ) [pure virtual]

获取API的系统错误

返回

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

备注

可以在调用api接口失败时调用, 例如login失败时

#### 5.3.2.3 virtual const char\* GetApiVersion ( ) [pure virtual]

获取API的发行版本号

返回

返回api发行版本号

#### 5.3.2.4 virtual const char\* GetTradingDay ( ) [pure virtual]

获取当前交易日

返回

获取到的交易日

备注

只有登录成功后,才能得到正确的交易日

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

用户登录请求

返回

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

参数

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

备注

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

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

登出请求

返回

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

备注

此函数为同步阻塞式，不需要异步等待登出，当函数返回即可进行后续操作

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

获取当前交易日可交易合约

返回

查询是否成功，“0”表示查询成功，非“0”表示查询不成功

参数

<i>exchange_id</i>	交易所代码
--------------------	-------

**5.3.2.8** `virtual int QueryAllTickersPricelInfo ( ) [pure virtual]`

获取所有合约的最新价格信息

返回

查询是否成功，“0”表示查询成功，非“0”表示查询不成功

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

获取合约的最新价格信息

返回

查询是否成功，“0”表示查询成功，非“0”表示查询不成功

参数

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

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

注册回调接口

参数

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

**5.3.2.11** `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

**5.3.2.12** `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

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

设置采用UDP方式连接时的接收缓冲区大小

备注

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

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

订阅全市场的股票行情

返回

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

参数

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

备注

需要与全市场退订行情接口配套使用

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

订阅全市场的期权行情

返回

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

参数

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

备注

需要与全市场退订期权行情接口配套使用

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

订阅全市场的期权行情订单簿

返回

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

参数

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

备注

需要与全市场退订期权行情订单簿接口配套使用



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

订阅全市场的期权逐笔行情

返回

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

参数

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

备注

需要与全市场退订期权逐笔行情接口配套使用

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

订阅全市场的股票行情订单簿

返回

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

参数

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

备注

需要与全市场退订行情订单簿接口配套使用

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

订阅全市场的股票逐笔行情

返回

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

参数

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

备注

需要与全市场退订逐笔行情接口配套使用

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

订阅行情，包括股票、指数和期权。

返回

订阅接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

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

备注

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

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

订阅行情订单簿，包括股票、指数和期权。

返回

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

参数

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

备注

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

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

订阅逐笔行情，包括股票、指数和期权。

返回

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

参数

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

备注

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

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

退订全市场的股票行情

返回

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

参数

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

备注

需要与订阅全市场行情接口配套使用

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

退订全市场的期权行情

返回

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

参数

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

备注

需要与订阅全市场期权行情接口配套使用

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

退订全市场的期权行情订单簿

返回

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

参数

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

备注

需要与订阅全市场期权行情订单簿接口配套使用

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

退订全市场的期权逐笔行情

返回

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

参数

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

备注

需要与订阅全市场期权逐笔行情接口配套使用

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

退订全市场的股票行情订单簿

返回

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

参数

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

备注

需要与订阅全市场行情订单簿接口配套使用

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

退订全市场的股票逐笔行情

返回

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

参数

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

备注

需要与订阅全市场逐笔行情接口配套使用

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

退订行情，包括股票、指数和期权。

返回

取消订阅接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

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

备注

可以一次性取消订阅同一证券交易所的多个合约，需要与订阅行情接口配套使用

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

退订行情订单簿，包括股票、指数和期权。

返回

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

参数

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

备注

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

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

退订逐笔行情，包括股票、指数和期权。

返回

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

参数

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

备注

可以一次性取消订阅同一证券交易所的多个合约，需要与订阅逐笔行情接口配套使用

该类的文档由以下文件生成:

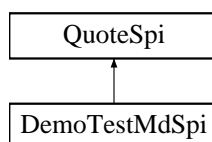
- [xtp\\_quote\\_api.h](#)

## 5.4 QuoteSpi类 参考

行情回调类

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

类 QuoteSpi 继承关系图:



**Public** 成员函数

- virtual void [OnDisconnected](#) (int reason)
- virtual void [OnError](#) (XTPRI \*error\_info)
- virtual void [OnSubMarketData](#) (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)
- virtual void [OnUnSubMarketData](#) (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)
- virtual void [OnDepthMarketData](#) (XTPMD \*market\_data, int64\_t bid1\_qty[], int32\_t bid1\_count, int32\_t max\_bid1\_count, int64\_t ask1\_qty[], int32\_t ask1\_count, int32\_t max\_ask1\_count)
- virtual void [OnSubOrderBook](#) (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)
- virtual void [OnUnSubOrderBook](#) (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)
- virtual void [OnOrderBook](#) (XTPOB \*order\_book)
- virtual void [OnSubTickByTick](#) (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)
- virtual void [OnUnSubTickByTick](#) (XTPST \*ticker, XTPRI \*error\_info, bool is\_last)
- virtual void [OnTickByTick](#) (XTPTBT \*tbt\_data)
- virtual void [OnSubscribeAllMarketData](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnUnSubscribeAllMarketData](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnSubscribeAllOrderBook](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnUnSubscribeAllOrderBook](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnSubscribeAllTickByTick](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnUnSubscribeAllTickByTick](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnQueryAllTickers](#) (XTPQSI \*ticker\_info, XTPRI \*error\_info, bool is\_last)
- virtual void [OnQueryTickersPriceInfo](#) (XTPPTPI \*ticker\_info, XTPRI \*error\_info, bool is\_last)
- virtual void [OnSubscribeAllOptionMarketData](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnUnSubscribeAllOptionMarketData](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnSubscribeAllOptionOrderBook](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnUnSubscribeAllOptionOrderBook](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnSubscribeAllOptionTickByTick](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)
- virtual void [OnUnSubscribeAllOptionTickByTick](#) (XTP\_EXCHANGE\_TYPE exchange\_id, XTPRI \*error\_info)

### 5.4.1 详细描述

行情回调类

作者

中泰证券股份有限公司

日期

十月 2015

### 5.4.2 成员函数说明

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

深度行情通知，包含买一卖一队列

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestMdSpi](#) 重载.

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

当客户端与行情后台通信连接断开时，该方法被调用。

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

错误应答

参数

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

备注

此函数只有在服务器发生错误时才会调用，一般无需用户处理

被 [DemoTestMdSpi](#) 重载.

#### 5.4.2.4 virtual void OnOrderBook ( XTOB \* *order\_book* ) [inline],[virtual]

行情订单簿通知，包括股票、指数和期权

参数

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

被 [DemoTestMdSpi](#) 重载.

#### 5.4.2.5 virtual void OnQueryAllTickers ( XTPQSI \* *ticker\_info*, XTPRI \* *error\_info*, bool *is\_last* ) [inline],[virtual]

查询可交易合约的应答

参数

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

被 [DemoTestMdSpi](#) 重载.

#### 5.4.2.6 virtual void OnQueryTickersPriceInfo ( XTPTPI \* *ticker\_info*, XTPRI \* *error\_info*, bool *is\_last* ) [inline],[virtual]

查询合约的最新价格信息应答

参数

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

被 [DemoTestMdSpi](#) 重载.

#### 5.4.2.7 virtual void OnSubMarketData ( XTPST \* *ticker*, XTPRI \* *error\_info*, bool *is\_last* ) [inline],[virtual]

订阅行情应答，包括股票、指数和期权



参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

订阅行情订单簿应答，包括股票、指数和期权

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

订阅全市场的股票行情应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

订阅全市场的期权行情应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

订阅全市场的期权行情订单簿应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

订阅全市场的期权逐笔行情应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

订阅全市场的股票行情订单簿应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

订阅全市场的股票逐笔行情应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

订阅逐笔行情应答，包括股票、指数和期权

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

逐笔行情通知，包括股票、指数和期权

参数

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

被 [DemoTestMdSpi](#) 重载.

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

退订行情应答，包括股票、指数和期权

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

退订行情订单簿应答，包括股票、指数和期权

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

退订全市场的股票行情应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

退订全市场的期权行情应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

退订全市场的期权行情订单簿应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

退订全市场的期权逐笔行情应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

退订全市场的股票行情订单簿应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

退订全市场的股票逐笔行情应答

参数

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

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

退订逐笔行情应答，包括股票、指数和期权

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载。  
该类的文档由以下文件生成:

- [xtp\\_quote\\_api.h](#)

## 5.5 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

### 5.5.1 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](#)

## 5.6 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

### 5.6.1 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`

## 5.7 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)

### 5.7.1 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`

## 5.8 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

### 5.8.1 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`

## 5.9 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
- 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

### 5.9.1 详细描述

行情

该结构体的文档由以下文件生成:

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

## 5.10 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

### 5.10.1 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](#)

## 5.11 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](#)  
  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
- [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，用户无需关心
- [XTPOrderTypeType order\\_type](#)  
报单类型

### 5.11.1 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](#)

## 5.12 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](#)  
    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

### 5.12.1 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](#)

## 5.13 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(=可用资金 + 证券资产（目前为0） + 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
- double [fund\\_buy\\_fee](#)
  - 累计买入成交交易费用
- double [fund\\_sell\\_amount](#)
  - 累计卖出成交证券所得资金
- double [fund\\_sell\\_fee](#)
  - 累计卖出成交交易费用
- double [withholding\\_amount](#)
  - XTP系统预扣的资金（包括购买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
- double [frozen\\_exec\\_cash](#)
  - 行权冻结资金
- double [frozen\\_exec\\_fee](#)
  - 行权费用
- double [pay\\_later](#)
  - 垫付资金
- double [preadva\\_pay](#)
  - 预垫付资金
- double [orig\\_banlance](#)
  - 昨日余额
- double [banlance](#)
  - 当前余额
- double [deposit\\_withdraw](#)
  - 当天出入金
- double [trade\\_netting](#)
  - 当日交易资金轧差
- double [captial\\_asset](#)
  - 资金资产
- double [force\\_freeze\\_amount](#)
  - 强锁资金
- double [preferred\\_amount](#)
  - 可取资金
- uint64\_t [unknown](#) [43-12]
  - (保留字段)

### 5.13.1 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](#)

## 5.14 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

### 5.14.1 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](#)

## 5.15 XTPQueryETFBaseRsp结构体 参考

查询股票ETF合约基本情况-响应结构体

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

## 成员变量

- [XTP\\_MARKET\\_TYPE market](#)  
交易市场
- `char etf [XTP_TICKER_LEN]`  
*etf*代码,买卖,申赎统一使用该代码
- `char subscribe_redemption_ticker [XTP_TICKER_LEN]`  
*etf*申购赎回代码
- `int32_t unit`  
最小申购赎回单位对应的*ETF*份数,例如上证"50*ETF*"就是900000
- `int32_t subscribe_status`  
是否允许申购,1-允许,0-禁止
- `int32_t redemption_status`  
是否允许赎回,1-允许,0-禁止
- `double max_cash_ratio`  
最大现金替代比例,小于1的数值 *TODO* 是否采用 *double*
- `double estimate_amount`  
7日预估金额
- `double cash_component`  
7-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*

### 5.15.1 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](#)



## 5.16 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

### 5.16.1 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](#)

## 5.17 XTPQueryETFComponentRsp结构体 参考

查询股票ETF合约成分股信息-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](#)  
成份股替代标识为必须现金替代时候的总金额

### 5.17.1 详细描述

查询股票ETF合约成分股信息-响应结构体

该结构体的文档由以下文件生成:

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

## 5.18 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

### 5.18.1 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](#)

## 5.19 XTPQueryIPOQuotaRsp结构体 参考

查询用户申购额度

```
#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

### 5.19.1 详细描述

查询用户申购额度

该结构体的文档由以下文件生成:

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

## 5.20 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
- 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

### 5.20.1 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](#)

## 5.21 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

### 5.21.1 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](#)

## 5.22 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)

### 5.22.1 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](#)

## 5.23 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

### 5.23.1 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](#)

## 5.24 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

### 5.24.1 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](#)

## 5.25 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)

### 5.25.1 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](#)

## 5.26 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

### 5.26.1 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](#)

## 5.27 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

### 5.27.1 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`

## 5.28 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10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)



### 5.28.1 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](#)

## 5.29 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

### 5.29.1 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](#)

## 5.30 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

### 5.30.1 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](#)

### 5.31 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

#### 5.31.1 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](#)

### 5.32 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

### 5.32.1 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](#)

## 5.33 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`  
};

### 5.33.1 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](#)

## 5.34 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`

### 5.34.1 详细描述

逐笔成交

### 5.34.2 结构体成员变量说明

#### 5.34.2.1 `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`

## 5.35 XPTickerPriceInfo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

### 5.35.1 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`

## 5.36 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, 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
- `union {`  
  - `uint32_t u32`
  - `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

### 5.36.1 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](#)

## Chapter 6

# 文件说明

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

定义控制台测试应用程序的入口点

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

#### 函数

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

#### 变量

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

#### 6.1.1 详细描述

定义控制台测试应用程序的入口点

作者

中泰证券股份有限公司

## 6.1.2 函数说明

### 6.1.2.1 int main ( )

```
int main() {  
    测试Demo入口函数  
    // 初始化UserApi  
    user_api_pointer = XTP::API::QuoteApi::CreateQuoteApi(client_id,filepath); // 创建UserApi  
    user_spi_pointer->SetHeartBeatInterval(15); //设置心跳超时时间间隔, 单位为秒  
    user_api_pointer->SetUDPBufferSize(128); //设置UDP接收缓冲区大小, 单位为MB  
    DemoTestMdSpi* user_spi_pointer = new DemoTestMdSpi(); //创建响应类实例  
    user_api_pointer->RegisterSpi(user_spi_pointer); // 注册事件类  
    int loginResult = user_api_pointer->Login("127.0.0.1", 6666, username, password, XTP_PROTOCOL_UDP); //采用UDP方式登陆行情订阅服务器  
    if (loginResult == 0)  
    { //登录成功  
        user_api_pointer->SubscribeMarketData(ppTicker, ticker_count, XTP_EXCHANGE_SZ); //订阅股票行情  
    }  
    return 0;  
}
```

## 6.1.3 变量说明

### 6.1.3.1 int ticker\_count = 1

行情订阅数量

## 6.2 demo\_test\_quote\_spi.h 文件参考

Demo自定义客户端行情订阅响应接口类

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

结构体

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

### 6.2.1 详细描述

Demo自定义客户端行情订阅响应接口类



作者

中泰证券股份有限公司

## 6.3 xoms\_api\_fund\_struct.h 文件参考

定义资金划拨相关结构体类型

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

结构体

- struct [XTPFundTransferReq](#)  
用户资金请求

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"
```

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 [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 [XTPQueryAssetRsp](#)  
账户资金查询响应结构体
- struct [XTPQueryStkPositionRsp](#)  
查询股票持仓情况
- 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 [XTPQueryETFComponentRsp](#)  
查询股票ETF合约成分股信息-响应结构体
- struct [XTPQueryIPOTickerRsp](#)  
查询当日可申购新股信息
- struct [XTPQueryIPOQuotaRsp](#)  
查询用户申购额度
- 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

## 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

### 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 [XTPTickerPriceInfo](#)  
供查询的最新信息

类型定义

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

## 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

### 6.5.2 枚举类型说明

#### 6.5.2.1 enum `XTP_MARKETDATA_TYPE`

`XTP_MARKETDATA_TYPE`是行情快照数据类型

枚举值

`XTP_MARKETDATA_ACTUAL` 现货(股票/基金/债券等)

`XTP_MARKETDATA_OPTION` 期权

## 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 `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_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_CASH_REPAY_MARGIN 25`  
现金还款
- `#define XTP_SIDE_STOCK_REPAY_STOCK 26`  
现券还券
- `#define XTP_SIDE_UNKNOWN 27`  
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，现货适用
- `#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_UNKNOWN 8`  
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_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
- `typedef enum XTP_MARKET_TYPE XTP_MARKET_TYPE`  
*XTP\_MARKET\_TYPE*市场类型
- `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\\_SPLIT\\_MERGE\\_STATUS](#) XTP\_SPLIT\_MERGE\_STATUS  
XTP\_SPLIT\_MERGE\_STATUS 是一个基金当天拆分合并状态类型
- typedef enum [XTP\\_TBT\\_TYPE](#) XTP\_TBT\_TYPE  
XTP\_TBT\_TYPE 是一个逐笔回报类型
- typedef enum [XTP\\_OPT\\_CALL\\_OR\\_PUT\\_TYPE](#) XTP\_OPT\_CALL\_OR\_PUT\_TYPE  
XTP\_OPT\_CALL\_OR\_PUT\_TYPE 是一个认沽或认购类型
- typedef enum [XTP\\_OPT\\_EXERCISE\\_TYPE](#) XTP\_OPT\_EXERCISE\_TYPE  
XTP\_OPT\_EXERCISE\_TYPE 是一个行权方式类型
- typedef enum [XTP\\_POSITION\\_DIRECTION\\_TYPE](#) XTP\_POSITION\_DIRECTION\_TYPE  
XTP\_POSITION\_DIRECTION\_TYPE 是一个持仓方向类型
- typedef char [TXTPTradeType](#)  
TXTPTradeType 是成交类型类型
- typedef char [TXTPOrderType](#)  
TXTPOrderType 是报单类型类型

## 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
- 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
- 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\_NOTTRADEQUEUEING, 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

- 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*,  
*E\_INTER\_SZ*,  
*ERT\_CASH\_MUST\_INTER\_SZ*, *ERT\_CASH\_RECOMPUTE\_INTER\_OTHER*, *ERT\_CASH\_MUST\_INTER\_OTHER*, *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\_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\_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\_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\_SPLIT\_MERGE\_STATUS* { *XTP\_SPLIT\_MERGE\_STATUS\_ALLOW* = 0, *XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_SPLIT*, *XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_MERGE*, *XTP\_SPLIT\_MERGE\_STATUS\_FORBIDDEN* }  
*XTP\_SPLIT\_MERGE\_STATUS*是一个基金当天拆分合并状态类型
  - 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* { *XTP\_OPT\_EXERCISE\_TYPE\_EUR* = 1, *XTP\_OPT\_EXERCISE\_TYPE\_AME* = 2 }  
*XTP\_OPT\_EXERCISE\_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

### 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 枚举类型说明

### 6.6.2.1 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  
**ERT\_CASH\_MUST\_INTER\_OTHER** 表示非沪深市场成份证券必须现金替代  
**EPT\_INVALID** 无效值

### 6.6.2.2 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

### 6.6.2.3 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（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（暂未支持）  
**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\_PURCHASE\_REDEMPTION** 分级基金申赎业务  
**XTP\_BUSINESS\_TYPE\_STRUCTURED\_FUND\_SPLIT\_MERGE** 分级基金拆分合并业务  
**XTP\_BUSINESS\_TYPE\_MONEY\_FUND** 货币基金业务（暂未支持）  
**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\_UNKNOWN** 未知类型

#### 6.6.2.4 enum XTP\_EXCHANGE\_TYPE

XTP\_EXCHANGE\_TYPE是交易所类型

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

#### 6.6.2.5 enum XTP\_FUND\_OPER\_STATUS

XTP\_FUND\_OPER\_STATUS柜台资金操作结果

枚举值

**XTP\_FUND\_OPER\_PROCESSING** XOMS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

#### 6.6.2.6 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\_TRANSFER\_UNKNOWN** 未知类型

#### 6.6.2.7 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

### 6.6.2.8 enum XTP\_MARKET\_TYPE

XTP\_MARKET\_TYPE市场类型

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

### 6.6.2.9 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

### 6.6.2.10 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

### 6.6.2.11 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

### 6.6.2.12 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

#### 6.6.2.13 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

#### 6.6.2.14 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）

#### 6.6.2.15 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（除普通股票业务外，其余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  
**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** 最优五档即时成交剩余转撤销，市价单-沪深  
**XTP\_PRICE\_ALL\_OR\_CANCEL** 全部成交或撤销，市价单-深 / 沪期权  
**XTP\_PRICE\_FORWARD\_BEST** 本方最优，市价单-深  
**XTP\_PRICE\_REVERSE\_BEST\_LIMIT** 对方最优剩余转限价，市价单-深 / 沪期权  
**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

#### 6.6.2.16 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)

#### 6.6.2.17 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

#### 6.6.2.18 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

#### 6.6.2.19 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

#### 6.6.2.20 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\_UNKNOWN** 未知类型

## 6.7 xtp\_api\_struct.h 文件参考

定义业务数据结构

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

### 6.7.1 详细描述

定义业务数据结构

作者

中泰证券股份有限公司

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

定义业务公共数据结构

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

结构体

- struct [XTPRspInfoStruct](#)  
响应信息

宏定义

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

类型定义

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

### 6.8.1 详细描述

定义业务公共数据结构

作者

中泰证券股份有限公司

## 6.9 xtp\_quote\_api.h 文件参考

定义行情订阅客户端接口

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

结构体

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

### 6.9.1 详细描述

定义行情订阅客户端接口

作者

中泰证券股份有限公司





# Index

- CreateQuoteApi
  - XTP::API::QuoteApi, [12](#)
- demo\_test\_quote\_api.cpp, [55](#)
  - main, [56](#)
  - ticker\_count, [56](#)
- demo\_test\_quote\_spi.h, [56](#)
- DemoTestMdSpi, [9](#)
- EPT\_INVALID
  - xtp\_api\_data\_type.h, [65](#)
- ERT\_CASH\_FORBIDDEN
  - xtp\_api\_data\_type.h, [65](#)
- ERT\_CASH\_MUST
  - xtp\_api\_data\_type.h, [65](#)
- ERT\_CASH\_MUST\_INTER\_OTHER
  - xtp\_api\_data\_type.h, [65](#)
- ERT\_CASH\_MUST\_INTER\_SZ
  - xtp\_api\_data\_type.h, [65](#)
- ERT\_CASH\_OPTIONAL
  - xtp\_api\_data\_type.h, [65](#)
- ERT\_CASH\_RECOMPUTE\_INTER\_OTHER
  - xtp\_api\_data\_type.h, [65](#)
- ERT\_CASH\_RECOMPUTE\_INTER\_SZ
  - xtp\_api\_data\_type.h, [65](#)
- ETF\_REPLACE\_TYPE
  - xtp\_api\_data\_type.h, [65](#)
- GetApiLastError
  - XTP::API::QuoteApi, [13](#)
- GetApiVersion
  - XTP::API::QuoteApi, [13](#)
- GetTradingDay
  - XTP::API::QuoteApi, [13](#)
- Login
  - XTP::API::QuoteApi, [13](#)
- Logout
  - XTP::API::QuoteApi, [14](#)
- main
  - demo\_test\_quote\_api.cpp, [56](#)
- OnDepthMarketData
  - XTP::API::QuoteSpi, [23](#)
- OnDisconnected
  - XTP::API::QuoteSpi, [23](#)
- OnError
  - XTP::API::QuoteSpi, [23](#)
- OnOrderBook
  - XTP::API::QuoteSpi, [24](#)
- OnQueryAllTickers
  - XTP::API::QuoteSpi, [24](#)
- OnQueryTickersPriceInfo
  - XTP::API::QuoteSpi, [24](#)
- OnSubMarketData
  - XTP::API::QuoteSpi, [24](#)
- OnSubOrderBook
  - XTP::API::QuoteSpi, [25](#)
- OnSubTickByTick
  - XTP::API::QuoteSpi, [27](#)
- OnSubscribeAllMarketData
  - XTP::API::QuoteSpi, [25](#)
- OnSubscribeAllOptionMarketData
  - XTP::API::QuoteSpi, [25](#)
- OnSubscribeAllOptionOrderBook
  - XTP::API::QuoteSpi, [26](#)
- OnSubscribeAllOptionTickByTick
  - XTP::API::QuoteSpi, [26](#)
- OnSubscribeAllOrderBook
  - XTP::API::QuoteSpi, [26](#)
- OnSubscribeAllTickByTick
  - XTP::API::QuoteSpi, [27](#)
- OnTickByTick
  - XTP::API::QuoteSpi, [27](#)
- OnUnSubMarketData
  - XTP::API::QuoteSpi, [28](#)
- OnUnSubOrderBook
  - XTP::API::QuoteSpi, [28](#)
- OnUnSubTickByTick
  - XTP::API::QuoteSpi, [30](#)
- OnUnSubscribeAllMarketData
  - XTP::API::QuoteSpi, [28](#)
- OnUnSubscribeAllOptionMarketData
  - XTP::API::QuoteSpi, [29](#)
- OnUnSubscribeAllOptionOrderBook
  - XTP::API::QuoteSpi, [29](#)
- OnUnSubscribeAllOptionTickByTick
  - XTP::API::QuoteSpi, [29](#)
- OnUnSubscribeAllOrderBook
  - XTP::API::QuoteSpi, [30](#)
- OnUnSubscribeAllTickByTick
  - XTP::API::QuoteSpi, [30](#)
- OrderBookStruct, [10](#)
- QueryAllTickers
  - XTP::API::QuoteApi, [14](#)
- QueryAllTickersPriceInfo
  - XTP::API::QuoteApi, [14](#)
- QueryTickersPriceInfo
  - XTP::API::QuoteApi, [14](#)

- QuoteApi, [11](#)
- QuoteSpi, [22](#)
- RegisterSpi
  - XTP::API::QuoteApi, [15](#)
- Release
  - XTP::API::QuoteApi, [15](#)
- SetHeartBeatInterval
  - XTP::API::QuoteApi, [15](#)
- SetUDPBufferSize
  - XTP::API::QuoteApi, [15](#)
- SubscribeAllMarketData
  - XTP::API::QuoteApi, [15](#)
- SubscribeAllOptionMarketData
  - XTP::API::QuoteApi, [16](#)
- SubscribeAllOptionOrderBook
  - XTP::API::QuoteApi, [16](#)
- SubscribeAllOptionTickByTick
  - XTP::API::QuoteApi, [16](#)
- SubscribeAllOrderBook
  - XTP::API::QuoteApi, [17](#)
- SubscribeAllTickByTick
  - XTP::API::QuoteApi, [17](#)
- SubscribeMarketData
  - XTP::API::QuoteApi, [17](#)
- SubscribeOrderBook
  - XTP::API::QuoteApi, [18](#)
- SubscribeTickByTick
  - XTP::API::QuoteApi, [18](#)
- ticker\_count
  - demo\_test\_quote\_api.cpp, [56](#)
- trade\_flag
  - XTPTickByTickTrade, [52](#)
- UnSubscribeAllMarketData
  - XTP::API::QuoteApi, [19](#)
- UnSubscribeAllOptionMarketData
  - XTP::API::QuoteApi, [19](#)
- UnSubscribeAllOptionOrderBook
  - XTP::API::QuoteApi, [19](#)
- UnSubscribeAllOptionTickByTick
  - XTP::API::QuoteApi, [20](#)
- UnSubscribeAllOrderBook
  - XTP::API::QuoteApi, [20](#)
- UnSubscribeAllTickByTick
  - XTP::API::QuoteApi, [20](#)
- UnSubscribeMarketData
  - XTP::API::QuoteApi, [21](#)
- UnSubscribeOrderBook
  - XTP::API::QuoteApi, [21](#)
- UnSubscribeTickByTick
  - XTP::API::QuoteApi, [21](#)
- XTP::API::QuoteApi
  - CreateQuoteApi, [12](#)
  - GetApiLastError, [13](#)
  - GetApiVersion, [13](#)
  - GetTradingDay, [13](#)
  - Login, [13](#)
  - Logout, [14](#)
  - QueryAllTickers, [14](#)
  - QueryAllTickersPriceInfo, [14](#)
  - QueryTickersPriceInfo, [14](#)
  - RegisterSpi, [15](#)
  - Release, [15](#)
  - SetHeartBeatInterval, [15](#)
  - SetUDPBufferSize, [15](#)
  - SubscribeAllMarketData, [15](#)
  - SubscribeAllOptionMarketData, [16](#)
  - SubscribeAllOptionOrderBook, [16](#)
  - SubscribeAllOptionTickByTick, [16](#)
  - SubscribeAllOrderBook, [17](#)
  - SubscribeAllTickByTick, [17](#)
  - SubscribeMarketData, [17](#)
  - SubscribeOrderBook, [18](#)
  - SubscribeTickByTick, [18](#)
  - UnSubscribeAllMarketData, [19](#)
  - UnSubscribeAllOptionMarketData, [19](#)
  - UnSubscribeAllOptionOrderBook, [19](#)
  - UnSubscribeAllOptionTickByTick, [20](#)
  - UnSubscribeAllOrderBook, [20](#)
  - UnSubscribeAllTickByTick, [20](#)
  - UnSubscribeMarketData, [21](#)
  - UnSubscribeOrderBook, [21](#)
  - UnSubscribeTickByTick, [21](#)
- XTP::API::QuoteSpi
  - OnDepthMarketData, [23](#)
  - OnDisconnected, [23](#)
  - OnError, [23](#)
  - OnOrderBook, [24](#)
  - OnQueryAllTickers, [24](#)
  - OnQueryTickersPriceInfo, [24](#)
  - OnSubMarketData, [24](#)
  - OnSubOrderBook, [25](#)
  - OnSubTickByTick, [27](#)
  - OnSubscribeAllMarketData, [25](#)
  - OnSubscribeAllOptionMarketData, [25](#)
  - OnSubscribeAllOptionOrderBook, [26](#)
  - OnSubscribeAllOptionTickByTick, [26](#)
  - OnSubscribeAllOrderBook, [26](#)
  - OnSubscribeAllTickByTick, [27](#)
  - OnTickByTick, [27](#)
  - OnUnSubMarketData, [28](#)
  - OnUnSubOrderBook, [28](#)
  - OnUnSubTickByTick, [30](#)
  - OnUnSubscribeAllMarketData, [28](#)
  - OnUnSubscribeAllOptionMarketData, [29](#)
  - OnUnSubscribeAllOptionOrderBook, [29](#)
  - OnUnSubscribeAllOptionTickByTick, [29](#)
  - OnUnSubscribeAllOrderBook, [30](#)
  - OnUnSubscribeAllTickByTick, [30](#)
- XTP\_ACCOUNT\_CREDIT
  - xtp\_api\_data\_type.h, [65](#)
- XTP\_ACCOUNT\_DERIVE
  - xtp\_api\_data\_type.h, [65](#)

XTP\_ACCOUNT\_NORMAL  
     xtp\_api\_data\_type.h, 65  
 XTP\_ACCOUNT\_TYPE  
     xtp\_api\_data\_type.h, 65  
 XTP\_ACCOUNT\_UNKNOWN  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_ALLOTMENT  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_CASH  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_DESIGNATION  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE ETF  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_EXECUTE  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_FREEZE  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_IPOS  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_MARGIN  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_MONEY\_FUND  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_OPTION  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_REPO  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_STRUCTURED\_FUND\_PURCHASE\_REDEMPTION  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_STRUCTURED\_FUND\_SPLIT\_MERGE  
     xtp\_api\_data\_type.h, 65  
 XTP\_BUSINESS\_TYPE\_UNKNOWN  
     xtp\_api\_data\_type.h, 65  
 XTP\_EXCHANGE\_SH  
     xtp\_api\_data\_type.h, 66  
 XTP\_EXCHANGE\_SZ  
     xtp\_api\_data\_type.h, 66  
 XTP\_EXCHANGE\_TYPE  
     xtp\_api\_data\_type.h, 65  
 XTP\_EXCHANGE\_UNKNOWN  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_OPER\_FAILED  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_OPER\_PROCESSING  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_OPER\_STATUS  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_OPER\_SUBMITTED  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_OPER\_SUCCESS  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_OPER\_UNKNOWN  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_TRANSFER\_IN  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_TRANSFER\_OUT  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_TRANSFER\_TYPE  
     xtp\_api\_data\_type.h, 66  
 XTP\_FUND\_TRANSFER\_UNKNOWN  
     xtp\_api\_data\_type.h, 66  
 XTP\_LOG\_LEVEL  
     xtp\_api\_data\_type.h, 66  
 XTP\_LOG\_LEVEL\_DEBUG  
     xtp\_api\_data\_type.h, 66  
 XTP\_LOG\_LEVEL\_ERROR  
     xtp\_api\_data\_type.h, 66  
 XTP\_LOG\_LEVEL\_FATAL  
     xtp\_api\_data\_type.h, 66  
 XTP\_LOG\_LEVEL\_INFO  
     xtp\_api\_data\_type.h, 66  
 XTP\_LOG\_LEVEL\_TRACE  
     xtp\_api\_data\_type.h, 66  
 XTP\_LOG\_LEVEL\_WARNING  
     xtp\_api\_data\_type.h, 66  
 XTP\_MARKET\_TYPE  
     xtp\_api\_data\_type.h, 66  
 XTP\_MARKETDATA\_ACTUAL  
     xquote\_api\_struct.h, 60  
 XTP\_MARKETDATA\_OPTION  
     xquote\_api\_struct.h, 60  
 XTP\_MARKETDATA\_TYPE  
     xquote\_api\_struct.h, 60  
 XTP\_MKT\_INIT  
     xtp\_api\_data\_type.h, 67  
 XTP\_MKT\_SH\_A  
     xtp\_api\_data\_type.h, 67  
 XTP\_MKT\_SZ\_A  
     xtp\_api\_data\_type.h, 67  
 XTP\_MKT\_UNKNOWN  
     xtp\_api\_data\_type.h, 67  
 XTP\_OPT\_CALL  
     xtp\_api\_data\_type.h, 67  
 XTP\_OPT\_CALL\_OR\_PUT\_TYPE  
     xtp\_api\_data\_type.h, 67  
 XTP\_OPT\_EXERCISE\_TYPE\_AME  
     xtp\_api\_data\_type.h, 67  
 XTP\_OPT\_EXERCISE\_TYPE\_EUR  
     xtp\_api\_data\_type.h, 67  
 XTP\_OPT\_EXERCISE\_TYPE\_TYPE  
     xtp\_api\_data\_type.h, 67  
 XTP\_OPT\_PUT  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_ACTION\_STATUS\_ACCEPTED  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_ACTION\_STATUS\_REJECTED  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_ACTION\_STATUS\_SUBMITTED  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_ACTION\_STATUS\_TYPE  
     xtp\_api\_data\_type.h, 67

XTP\_ORDER\_STATUS\_ALLTRADED  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_STATUS\_CANCELED  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_STATUS\_INIT  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_STATUS\_NOTRADEQUEUEING  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_STATUS\_PARTTRADEDNOTQUEUEING  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_STATUS\_PARTTRADEDQUEUEING  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_STATUS\_REJECTED  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_STATUS\_TYPE  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_STATUS\_UNKNOWN  
     xtp\_api\_data\_type.h, 67  
 XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_ACCEPTED  
     xtp\_api\_data\_type.h, 68  
 XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_REJECTED  
     xtp\_api\_data\_type.h, 68  
 XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_SUBMITTED  
     xtp\_api\_data\_type.h, 68  
 XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_ACCEPTED  
     xtp\_api\_data\_type.h, 68  
 XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_REJECTED  
     xtp\_api\_data\_type.h, 68  
 XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_SUBMITTED  
     xtp\_api\_data\_type.h, 68  
 XTP\_ORDER\_SUBMIT\_STATUS\_TYPE  
     xtp\_api\_data\_type.h, 67  
 XTP\_POSITION\_DIRECTION\_COVERED  
     xtp\_api\_data\_type.h, 68  
 XTP\_POSITION\_DIRECTION\_LONG  
     xtp\_api\_data\_type.h, 68  
 XTP\_POSITION\_DIRECTION\_NET  
     xtp\_api\_data\_type.h, 68  
 XTP\_POSITION\_DIRECTION\_SHORT  
     xtp\_api\_data\_type.h, 68  
 XTP\_POSITION\_DIRECTION\_TYPE  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_ALL\_OR\_CANCEL  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_BEST5\_OR\_CANCEL  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_BEST5\_OR\_LIMIT  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_BEST\_OR\_CANCEL  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_FORWARD\_BEST  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_LIMIT  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_LIMIT\_OR\_CANCEL  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_REVERSE\_BEST\_LIMIT  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_TYPE  
     xtp\_api\_data\_type.h, 68  
 XTP\_PRICE\_TYPE\_UNKNOWN  
     xtp\_api\_data\_type.h, 68  
 XTP\_PROTOCOL\_TCP  
     xtp\_api\_data\_type.h, 68  
 XTP\_PROTOCOL\_TYPE  
     xtp\_api\_data\_type.h, 68  
 XTP\_PROTOCOL\_UDP  
     xtp\_api\_data\_type.h, 68  
 XTP\_SPLIT\_MERGE\_STATUS  
     xtp\_api\_data\_type.h, 68  
 XTP\_SPLIT\_MERGE\_STATUS\_ALLOW  
     xtp\_api\_data\_type.h, 69  
 XTP\_SPLIT\_MERGE\_STATUS\_FORBIDDEN  
     xtp\_api\_data\_type.h, 69  
 XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_MERGE  
     xtp\_api\_data\_type.h, 69  
 XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_SPLIT  
     xtp\_api\_data\_type.h, 69  
 XTP\_TBT\_ENTRUST  
     xtp\_api\_data\_type.h, 69  
 XTP\_TBT\_TRADE  
     xtp\_api\_data\_type.h, 69  
 XTP\_TBT\_TYPE  
     xtp\_api\_data\_type.h, 69  
 XTP\_TE\_RESUME\_TYPE  
     xtp\_api\_data\_type.h, 69  
 XTP\_TERT\_QUICK  
     xtp\_api\_data\_type.h, 69  
 XTP\_TERT\_RESTART  
     xtp\_api\_data\_type.h, 69  
 XTP\_TERT\_RESUME  
     xtp\_api\_data\_type.h, 69  
 XTP\_TICKER\_TYPE  
     xtp\_api\_data\_type.h, 69  
 XTP\_TICKER\_TYPE\_BOND  
     xtp\_api\_data\_type.h, 69  
 XTP\_TICKER\_TYPE\_FUND  
     xtp\_api\_data\_type.h, 69  
 XTP\_TICKER\_TYPE\_INDEX  
     xtp\_api\_data\_type.h, 69  
 XTP\_TICKER\_TYPE\_OPTION  
     xtp\_api\_data\_type.h, 69  
 XTP\_TICKER\_TYPE\_STOCK  
     xtp\_api\_data\_type.h, 69  
 XTP\_TICKER\_TYPE\_UNKNOWN  
     xtp\_api\_data\_type.h, 69  
 XTPFundTransferNotice, 31  
 XTPFundTransferReq, 31  
 XTPMarketDataOptionExData, 32

- XTPMarketDataStockExData, 32
- XTPMarketDataStruct, 34
- XTPOrderCancelInfo, 36
- XTPOrderInfo, 36
- XTPOrderInsertInfo, 37
- XTPQueryAssetRsp, 38
- XTPQueryETFBaseReq, 39
- XTPQueryETFBaseRsp, 40
- XTPQueryETFComponentReq, 41
- XTPQueryETFComponentRsp, 41
- XTPQueryFundTransferLogReq, 42
- XTPQueryIPOQuotaRsp, 42
- XTPQueryIPTickerRsp, 42
- XTPQueryOptionAuctionInfoReq, 43
- XTPQueryOptionAuctionInfoRsp, 43
- XTPQueryOrderReq, 45
- XTPQueryReportByExecIdReq, 46
- XTPQueryStkPositionRsp, 46
- XTPQueryStructuredFundInfoReq, 47
- XTPQueryTraderReq, 48
- XTPQuoteStaticInfo, 48
- XTPRspInfoStruct, 49
- XTPSpecificTickerStruct, 49
- XTPStructuredFundInfo, 50
- XTPTickByTickEntrust, 50
- XTPTickByTickStruct, 51
- XTPTickByTickTrade, 51
  - trade\_flag, 52
- XTPTickerPriceInfo, 52
- XTPTradeReport, 53
- xoms\_api\_fund\_struct.h, 57
- xoms\_api\_struct.h, 57
- xquote\_api\_struct.h, 59
  - XTP\_MARKETDATA\_ACTUAL, 60
  - XTP\_MARKETDATA\_OPTION, 60
  - XTP\_MARKETDATA\_TYPE, 60
- xtp\_api\_data\_type.h, 60
  - EPT\_INVALID, 65
  - ERT\_CASH\_FORBIDDEN, 65
  - ERT\_CASH\_MUST, 65
  - ERT\_CASH\_MUST\_INTER\_OTHER, 65
  - ERT\_CASH\_MUST\_INTER\_SZ, 65
  - ERT\_CASH\_OPTIONAL, 65
  - ERT\_CASH\_RECOMPUTE\_INTER\_OTHER, 65
  - ERT\_CASH\_RECOMPUTE\_INTER\_SZ, 65
  - ETF\_REPLACE\_TYPE, 65
  - XTP\_ACCOUNT\_CREDIT, 65
  - XTP\_ACCOUNT\_DERIVE, 65
  - XTP\_ACCOUNT\_NORMAL, 65
  - XTP\_ACCOUNT\_TYPE, 65
  - XTP\_ACCOUNT\_UNKNOWN, 65
  - XTP\_BUSINESS\_TYPE, 65
  - XTP\_BUSINESS\_TYPE\_ALLOTMENT, 65
  - XTP\_BUSINESS\_TYPE\_CASH, 65
  - XTP\_BUSINESS\_TYPE\_DESIGNATION, 65
  - XTP\_BUSINESS\_TYPE ETF, 65
  - XTP\_BUSINESS\_TYPE\_EXECUTE, 65
  - XTP\_BUSINESS\_TYPE\_FREEZE, 65
  - XTP\_BUSINESS\_TYPE\_IPOS, 65
  - XTP\_BUSINESS\_TYPE\_MARGIN, 65
  - XTP\_BUSINESS\_TYPE\_MONEY\_FUND, 65
  - XTP\_BUSINESS\_TYPE\_OPTION, 65
  - XTP\_BUSINESS\_TYPE\_REPO, 65
  - XTP\_BUSINESS\_TYPE\_STRUCTURED\_FUND←  
\_PURCHASE\_REDEMPTION, 65
  - XTP\_BUSINESS\_TYPE\_STRUCTURED\_FUND←  
\_SPLIT\_MERGE, 65
  - XTP\_BUSINESS\_TYPE\_UNKNOWN, 65
  - XTP\_EXCHANGE\_SH, 66
  - XTP\_EXCHANGE\_SZ, 66
  - XTP\_EXCHANGE\_TYPE, 65
  - XTP\_EXCHANGE\_UNKNOWN, 66
  - XTP\_FUND\_OPER\_FAILED, 66
  - XTP\_FUND\_OPER\_PROCESSING, 66
  - XTP\_FUND\_OPER\_STATUS, 66
  - XTP\_FUND\_OPER\_SUBMITTED, 66
  - XTP\_FUND\_OPER\_SUCCESS, 66
  - XTP\_FUND\_OPER\_UNKNOWN, 66
  - XTP\_FUND\_TRANSFER\_IN, 66
  - XTP\_FUND\_TRANSFER\_OUT, 66
  - XTP\_FUND\_TRANSFER\_TYPE, 66
  - XTP\_FUND\_TRANSFER\_UNKNOWN, 66
  - XTP\_LOG\_LEVEL, 66
  - XTP\_LOG\_LEVEL\_DEBUG, 66
  - XTP\_LOG\_LEVEL\_ERROR, 66
  - XTP\_LOG\_LEVEL\_FATAL, 66
  - XTP\_LOG\_LEVEL\_INFO, 66
  - XTP\_LOG\_LEVEL\_TRACE, 66
  - XTP\_LOG\_LEVEL\_WARNING, 66
  - XTP\_MARKET\_TYPE, 66
  - XTP\_MKT\_INIT, 67
  - XTP\_MKT\_SH\_A, 67
  - XTP\_MKT\_SZ\_A, 67
  - XTP\_MKT\_UNKNOWN, 67
  - XTP\_OPT\_CALL, 67
  - XTP\_OPT\_CALL\_OR\_PUT\_TYPE, 67
  - XTP\_OPT\_EXERCISE\_TYPE\_AME, 67
  - XTP\_OPT\_EXERCISE\_TYPE\_EUR, 67
  - XTP\_OPT\_EXERCISE\_TYPE\_TYPE, 67
  - XTP\_OPT\_PUT, 67
  - XTP\_ORDER\_ACTION\_STATUS\_ACCEPTED, 67
  - XTP\_ORDER\_ACTION\_STATUS\_REJECTED, 67
  - XTP\_ORDER\_ACTION\_STATUS\_SUBMITTED, 67
  - XTP\_ORDER\_ACTION\_STATUS\_TYPE, 67
  - XTP\_ORDER\_STATUS\_ALLTRADED, 67
  - XTP\_ORDER\_STATUS\_CANCELED, 67
  - XTP\_ORDER\_STATUS\_INIT, 67
  - XTP\_ORDER\_STATUS\_NOTRADEQUEUEING, 67
  - XTP\_ORDER\_STATUS\_PARTTRADEDNOTQ←  
UEUEING, 67
  - XTP\_ORDER\_STATUS\_PARTTRADEDQUEUE←  
ING, 67
  - XTP\_ORDER\_STATUS\_REJECTED, 67

XTP\_ORDER\_STATUS\_TYPE, [67](#)  
 XTP\_ORDER\_STATUS\_UNKNOWN, [67](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_ACCEPTED, [68](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_REJECTED, [68](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_SUBMITTED, [68](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_ACCEPTED, [68](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_REJECTED, [68](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_SUBMITTED, [68](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_TYPE, [67](#)  
 XTP\_POSITION\_DIRECTION\_COVERED, [68](#)  
 XTP\_POSITION\_DIRECTION\_LONG, [68](#)  
 XTP\_POSITION\_DIRECTION\_NET, [68](#)  
 XTP\_POSITION\_DIRECTION\_SHORT, [68](#)  
 XTP\_POSITION\_DIRECTION\_TYPE, [68](#)  
 XTP\_PRICE\_ALL\_OR\_CANCEL, [68](#)  
 XTP\_PRICE\_BEST5\_OR\_CANCEL, [68](#)  
 XTP\_PRICE\_BEST5\_OR\_LIMIT, [68](#)  
 XTP\_PRICE\_BEST\_OR\_CANCEL, [68](#)  
 XTP\_PRICE\_FORWARD\_BEST, [68](#)  
 XTP\_PRICE\_LIMIT, [68](#)  
 XTP\_PRICE\_LIMIT\_OR\_CANCEL, [68](#)  
 XTP\_PRICE\_REVERSE\_BEST\_LIMIT, [68](#)  
 XTP\_PRICE\_TYPE, [68](#)  
 XTP\_PRICE\_TYPE\_UNKNOWN, [68](#)  
 XTP\_PROTOCOL\_TCP, [68](#)  
 XTP\_PROTOCOL\_TYPE, [68](#)  
 XTP\_PROTOCOL\_UDP, [68](#)  
 XTP\_SPLIT\_MERGE\_STATUS, [68](#)  
 XTP\_SPLIT\_MERGE\_STATUS\_ALLOW, [69](#)  
 XTP\_SPLIT\_MERGE\_STATUS\_FORBIDDEN, [69](#)  
 XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_MERGE, [69](#)  
 XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_SPLIT, [69](#)  
 XTP\_TBT\_ENTRUST, [69](#)  
 XTP\_TBT\_TRADE, [69](#)  
 XTP\_TBT\_TYPE, [69](#)  
 XTP\_TE\_RESUME\_TYPE, [69](#)  
 XTP\_TERT\_QUICK, [69](#)  
 XTP\_TERT\_RESTART, [69](#)  
 XTP\_TERT\_RESUME, [69](#)  
 XTP\_TICKER\_TYPE, [69](#)  
 XTP\_TICKER\_TYPE\_BOND, [69](#)  
 XTP\_TICKER\_TYPE\_FUND, [69](#)  
 XTP\_TICKER\_TYPE\_INDEX, [69](#)  
 XTP\_TICKER\_TYPE\_OPTION, [69](#)  
 XTP\_TICKER\_TYPE\_STOCK, [69](#)  
 XTP\_TICKER\_TYPE\_UNKNOWN, [69](#)  
 xtp\_api\_struct.h, [69](#)  
 xtp\_api\_struct\_common.h, [70](#)  
 xtp\_quote\_api.h, [70](#)