

# XTP极速交易系统TraderAPI

制作者 中泰证券股份有限公司

2018年 六月 28日 星期四 17:34:53



# Contents

<b>1</b>	<b>XTP 极速行情交易系统 Trader 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	DemoTestTraderSpi类 参考	9
5.1.1	详细描述	10
5.1.2	成员函数说明	10
5.1.2.1	OnFundTransfer(XTPFundTransferNotice *fund_transfer_info, XTPRI *error_info, uint64_t session_id)	10
5.1.2.2	OnQueryAsset(XTPQueryAssetRsp *asset, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	10
5.1.2.3	OnQueryETF(XTPQueryETFBaseRsp *etf_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	11
5.1.2.4	OnQueryETFBasket(XTPQueryETFComponentRsp *etf_component_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	11
5.1.2.5	OnQueryFundTransfer(XTPFundTransferNotice *fund_transfer_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	12
5.1.2.6	OnQueryIPOInfoList(XTPQueryIPOTickerRsp *ipo_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	13
5.1.2.7	OnQueryIPOQuotaInfo(XTPQueryIPOQuotaRsp *quota_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	13
5.1.2.8	OnQueryOptionAuctionInfo(XTPQueryOptionAuctionInfoRsp *option_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	14
5.1.2.9	OnQueryOrder(XTPQueryOrderRsp *order_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	14
5.1.2.10	OnQueryPosition(XTPQueryStkPositionRsp *position, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	14

5.1.2.11	OnQueryStructuredFund(XTPStructuredFundInfo *fund_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id) . . . . .	15
5.1.2.12	OnQueryTrade(XTPQueryTradeRsp *trade_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id) . . . . .	15
5.2	OrderBookStruct结构体 参考 . . . . .	16
5.2.1	详细描述 . . . . .	16
5.3	TraderApi类 参考 . . . . .	17
5.3.1	详细描述 . . . . .	17
5.3.2	成员函数说明 . . . . .	18
5.3.2.1	CancelOrder(const uint64_t order_xtp_id, uint64_t session_id)=0 . . . . .	18
5.3.2.2	CreateTraderApi(uint8_t client_id, const char *save_file_path, XTP_LOG_LEVEL log_level=XTP_LOG_LEVEL_DEBUG) . . . . .	18
5.3.2.3	FundTransfer(XTPFundTransferReq *fund_transfer, uint64_t session_id)=0 . . . . .	18
5.3.2.4	GetAccountByXTPID(uint64_t order_xtp_id)=0 . . . . .	19
5.3.2.5	GetApiLastError()=0 . . . . .	19
5.3.2.6	GetApiVersion()=0 . . . . .	19
5.3.2.7	GetClientIDByXTPID(uint64_t order_xtp_id)=0 . . . . .	19
5.3.2.8	GetTradingDay()=0 . . . . .	20
5.3.2.9	InsertOrder(XTPOrderInsertInfo *order, uint64_t session_id)=0 . . . . .	20
5.3.2.10	Login(const char *ip, int port, const char *user, const char *password, XTP_PROTOCOL_TYPE sock_type)=0 . . . . .	20
5.3.2.11	Logout(uint64_t session_id)=0 . . . . .	21
5.3.2.12	QueryAsset(uint64_t session_id, int request_id)=0 . . . . .	21
5.3.2.13	QueryETF(XTPQueryETFBaseReq *query_param, uint64_t session_id, int request_id)=0 . . . . .	21
5.3.2.14	QueryETFTickerBasket(XTPQueryETFComponentReq *query_param, uint64_t session_id, int request_id)=0 . . . . .	22
5.3.2.15	QueryFundTransfer(XTPQueryFundTransferLogReq *query_param, uint64_t session_id, int request_id)=0 . . . . .	22
5.3.2.16	QueryIPOInfoList(uint64_t session_id, int request_id)=0 . . . . .	22
5.3.2.17	QueryIPOQuotaInfo(uint64_t session_id, int request_id)=0 . . . . .	22
5.3.2.18	QueryOptionAuctionInfo(XTPQueryOptionAuctionInfoReq *query_param, uint64_t session_id, int request_id)=0 . . . . .	23
5.3.2.19	QueryOrderByXTPID(const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0 . . . . .	23
5.3.2.20	QueryOrders(const XTPQueryOrderReq *query_param, uint64_t session_id, int request_id)=0 . . . . .	23
5.3.2.21	QueryPosition(const char *ticker, uint64_t session_id, int request_id)=0 . . . . .	24
5.3.2.22	QueryStructuredFund(XTPQueryStructuredFundInfoReq *query_param, uint64_t session_id, int request_id)=0 . . . . .	24
5.3.2.23	QueryTrades(XTPQueryTraderReq *query_param, uint64_t session_id, int request_id)=0 . . . . .	24
5.3.2.24	QueryTradesByXTPID(const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0 . . . . .	25

5.3.2.25	RegisterSpi(TraderSpi *spi)=0	25
5.3.2.26	Release()=0	25
5.3.2.27	SetHeartBeatInterval(uint32_t interval)=0	25
5.3.2.28	SetSoftwareKey(const char *key)=0	26
5.3.2.29	SetSoftwareVersion(const char *version)=0	26
5.3.2.30	SubscribePublicTopic(XTP_TE_RESUME_TYPE resume_type)=0	26
5.4	TraderSpi类 参考	27
5.4.1	详细描述	27
5.4.2	成员函数说明	28
5.4.2.1	OnCancelOrderError(XTPOrderCancelInfo *cancel_info, XTPRI *error_info, uint64_t session_id)	28
5.4.2.2	OnDisconnected(uint64_t session_id, int reason)	28
5.4.2.3	OnError(XTPRI *error_info)	28
5.4.2.4	OnFundTransfer(XTPFundTransferNotice *fund_transfer_info, XTPRI *error_info, uint64_t session_id)	28
5.4.2.5	OnOrderEvent(XTPOrderInfo *order_info, XTPRI *error_info, uint64_t session_id)	29
5.4.2.6	OnQueryAsset(XTPQueryAssetResp *asset, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	29
5.4.2.7	OnQueryETF(XTPQueryETFBaseResp *etf_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	30
5.4.2.8	OnQueryETFBasket(XTPQueryETFComponentResp *etf_component_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	30
5.4.2.9	OnQueryFundTransfer(XTPFundTransferNotice *fund_transfer_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	30
5.4.2.10	OnQueryIPOInfoList(XTPQueryIPOTickerResp *ipo_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	31
5.4.2.11	OnQueryIPOQuotaInfo(XTPQueryIPOQuotaResp *quota_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	31
5.4.2.12	OnQueryOptionAuctionInfo(XTPQueryOptionAuctionInfoResp *option_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	32
5.4.2.13	OnQueryOrder(XTPQueryOrderResp *order_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	32
5.4.2.14	OnQueryPosition(XTPQueryStkPositionResp *position, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	32
5.4.2.15	OnQueryStructuredFund(XTPStructuredFundInfo *fund_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	33
5.4.2.16	OnQueryTrade(XTPQueryTradeResp *trade_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	33
5.4.2.17	OnTradeEvent(XTPTradeReport *trade_info, uint64_t session_id)	34
5.5	XTPFundTransferNotice结构体 参考	34
5.5.1	详细描述	35
5.6	XTPFundTransferReq结构体 参考	35
5.6.1	详细描述	35

5.7	XTPMarketDataOptionExData结构体 参考	35
5.7.1	详细描述	36
5.8	XTPMarketDataStockExData结构体 参考	36
5.8.1	详细描述	37
5.9	XTPMarketDataStruct结构体 参考	37
5.9.1	详细描述	39
5.10	XTPOrderCancelInfo结构体 参考	39
5.10.1	详细描述	39
5.11	XTPOrderInfo结构体 参考	39
5.11.1	详细描述	40
5.12	XTPOrderInsertInfo结构体 参考	41
5.12.1	详细描述	41
5.13	XTPQueryAssetRsp结构体 参考	41
5.13.1	详细描述	42
5.14	XTPQueryETFBaseReq结构体 参考	43
5.14.1	详细描述	43
5.15	XTPQueryETFBaseRsp结构体 参考	43
5.15.1	详细描述	44
5.16	XTPQueryETFComponentReq结构体 参考	44
5.16.1	详细描述	44
5.17	XTPQueryETFComponentRsp结构体 参考	44
5.17.1	详细描述	45
5.18	XTPQueryFundTransferLogReq结构体 参考	45
5.18.1	详细描述	45
5.19	XTPQueryIPOQuotaRsp结构体 参考	45
5.19.1	详细描述	45
5.20	XTPQueryIPOTickerRsp结构体 参考	46
5.20.1	详细描述	46
5.21	XTPQueryOptionAuctionInfoReq结构体 参考	46
5.21.1	详细描述	46
5.22	XTPQueryOptionAuctionInfoRsp结构体 参考	47
5.22.1	详细描述	48
5.23	XTPQueryOrderReq结构体 参考	48
5.23.1	详细描述	49
5.24	XTPQueryReportByExecIdReq结构体 参考	49
5.24.1	详细描述	49
5.25	XTPQueryStkPositionRsp结构体 参考	49
5.25.1	详细描述	50
5.26	XTPQueryStructuredFundInfoReq结构体 参考	50
5.26.1	详细描述	51

5.27 XTPQueryTraderReq结构体 参考	51
5.27.1 详细描述	51
5.28 XTPQuoteStaticInfo结构体 参考	51
5.28.1 详细描述	52
5.29 XTPRspInfoStruct结构体 参考	52
5.29.1 详细描述	52
5.30 XTPSpecificTickerStruct结构体 参考	52
5.30.1 详细描述	53
5.31 XTPStructuredFundInfo结构体 参考	53
5.31.1 详细描述	53
5.32 XTPTickByTickEntrust结构体 参考	53
5.32.1 详细描述	54
5.33 XTPTickByTickStruct结构体 参考	54
5.33.1 详细描述	54
5.34 XTPTickByTickTrade结构体 参考	55
5.34.1 详细描述	55
5.34.2 结构体成员变量说明	55
5.34.2.1 trade_flag	55
5.35 XTPTickerPricInfo结构体 参考	55
5.35.1 详细描述	56
5.36 XTPTradeReport结构体 参考	56
5.36.1 详细描述	57
<b>6 文件说明</b>	<b>59</b>
6.1 demo_test_trade_api.cpp 文件参考	59
6.1.1 详细描述	59
6.1.2 函数说明	59
6.1.2.1 main()	59
6.2 demo_test_trade_spi.h 文件参考	61
6.2.1 详细描述	61
6.3 xoms_api_fund_struct.h 文件参考	61
6.3.1 详细描述	62
6.4 xoms_api_struct.h 文件参考	62
6.4.1 详细描述	63
6.5 xquote_api_struct.h 文件参考	63
6.5.1 详细描述	64
6.5.2 枚举类型说明	64
6.5.2.1 XTP_MARKETDATA_TYPE	64
6.6 xtp_api_data_type.h 文件参考	65
6.6.1 详细描述	69

6.6.2	枚举类型说明	69
6.6.2.1	ETF_REPLACE_TYPE	69
6.6.2.2	XTP_ACCOUNT_TYPE	70
6.6.2.3	XTP_BUSINESS_TYPE	70
6.6.2.4	XTP_EXCHANGE_TYPE	70
6.6.2.5	XTP_FUND_OPER_STATUS	71
6.6.2.6	XTP_FUND_TRANSFER_TYPE	71
6.6.2.7	XTP_LOG_LEVEL	71
6.6.2.8	XTP_MARKET_TYPE	71
6.6.2.9	XTP_OPT_CALL_OR_PUT_TYPE	72
6.6.2.10	XTP_OPT_EXERCISE_TYPE_TYPE	72
6.6.2.11	XTP_ORDER_ACTION_STATUS_TYPE	72
6.6.2.12	XTP_ORDER_STATUS_TYPE	72
6.6.2.13	XTP_ORDER_SUBMIT_STATUS_TYPE	72
6.6.2.14	XTP_POSITION_DIRECTION_TYPE	73
6.6.2.15	XTP_PRICE_TYPE	73
6.6.2.16	XTP_PROTOCOL_TYPE	73
6.6.2.17	XTP_SPLIT_MERGE_STATUS	73
6.6.2.18	XTP_TBT_TYPE	74
6.6.2.19	XTP_TE_RESUME_TYPE	74
6.6.2.20	XTP_TICKER_TYPE	74
6.7	xtp_api_struct.h 文件参考	74
6.7.1	详细描述	74
6.8	xtp_api_struct_common.h 文件参考	75
6.8.1	详细描述	75
6.9	xtp_trader_api.h 文件参考	75
6.9.1	详细描述	75
	索引	77



## Chapter 1

# XTP 极速行情交易系统 Trader API 1.1.18.13

本项目是XTP项目中的交易类接口

(1) XTP的交易接口和响应类 [xtp\\_trader\\_api.h](#)

(2) 程序化交易接口测试Demo [demo\\_test\\_trade\\_api.cpp](#)



## Chapter 2

# 继承关系索引

### 2.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

OrderBookStruct . . . . .	16
TraderApi . . . . .	17
TraderSpi . . . . .	27
DemoTestTraderSpi . . . . .	9
XTPFundTransferNotice . . . . .	34
XTPFundTransferReq . . . . .	35
XTPMarketDataOptionExData . . . . .	35
XTPMarketDataStockExData . . . . .	36
XTPMarketDataStruct . . . . .	37
XTPOrderCancelInfo . . . . .	39
XTPOrderInfo . . . . .	39
XTPOrderInsertInfo . . . . .	41
XTPQueryAssetRsp . . . . .	41
XTPQueryETFBaseReq . . . . .	43
XTPQueryETFBaseRsp . . . . .	43
XTPQueryETFComponentReq . . . . .	44
XTPQueryETFComponentRsp . . . . .	44
XTPQueryFundTransferLogReq . . . . .	45
XTPQueryIPOQuotaRsp . . . . .	45
XTPQueryIPOTickerRsp . . . . .	46
XTPQueryOptionAuctionInfoReq . . . . .	46
XTPQueryOptionAuctionInfoRsp . . . . .	47
XTPQueryOrderReq . . . . .	48
XTPQueryReportByExecIdReq . . . . .	49
XTPQueryStkPositionRsp . . . . .	49
XTPQueryStructuredFundInfoReq . . . . .	50
XTPQueryTraderReq . . . . .	51
XTPQuoteStaticInfo . . . . .	51
XTPRspInfoStruct . . . . .	52
XTPSpecificTickerStruct . . . . .	52
XTPStructuredFundInfo . . . . .	53
XTPTickByTickEntrust . . . . .	53
XTPTickByTickStruct . . . . .	54
XTPTickByTickTrade . . . . .	55
XTPTickerPriceInfo . . . . .	55
XTPTradeReport . . . . .	56



## Chapter 3

# 结构体索引

### 3.1 结构体

这里列出了所有结构体，并附带简要说明：

DemoTestTraderSpi	
Demo自定义交易接口响应类	9
OrderBookStruct	
定单簿	16
TraderApi	
交易接口类	17
TraderSpi	
交易接口响应类	27
XTPFundTransferNotice	
资金内转流水通知	34
XTPFundTransferReq	
用户资金请求	35
XTPMarketDataOptionExData	
期权额外数据	35
XTPMarketDataStockExData	
股票、基金、债券等额外数据	36
XTPMarketDataStruct	
行情	37
XTPOrderCancelInfo	
撤单失败响应消息	39
XTPOrderInfo	
报单响应结构体	39
XTPOrderInsertInfo	
新订单请求	41
XTPQueryAssetRsp	
账户资金查询响应结构体	41
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	43
XTPQueryETFBaseRsp	
查询股票ETF合约基本情况-响应结构体	43
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	44
XTPQueryETFComponentRsp	
查询股票ETF合约成分股信息-响应结构体	44
XTPQueryFundTransferLogReq	
资金内转流水查询请求与响应	45
XTPQueryIPOQuotaRsp	
查询用户申购额度	45

<a href="#">XTPQueryIPOTickerRsp</a>	
查询当日可申购新股信息	46
<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	46
<a href="#">XTPQueryOptionAuctionInfoRsp</a>	
查询期权竞价交易业务参考信息	47
<a href="#">XTPQueryOrderReq</a>	
报单查询 // 报单查询请求-条件查询	48
<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)	49
<a href="#">XTPQueryStkPositionRsp</a>	
查询股票持仓情况	49
<a href="#">XTPQueryStructuredFundInfoReq</a>	
查询分级基金信息结构体	50
<a href="#">XTPQueryTraderReq</a>	
查询成交回报请求-查询条件	51
<a href="#">XTPQuoteStaticInfo</a>	
股票行情静态信息	51
<a href="#">XTPRspInfoStruct</a>	
响应信息	52
<a href="#">XTPSpecificTickerStruct</a>	
指定的合约	52
<a href="#">XTPStructuredFundInfo</a>	
查询分级基金信息响应结构体	53
<a href="#">XTPTickByTickEntrust</a>	
逐笔委托(仅适用深交所)	53
<a href="#">XTPTickByTickStruct</a>	
逐笔数据信息	54
<a href="#">XTPTickByTickTrade</a>	
逐笔成交	55
<a href="#">XTPTickerPriceInfo</a>	
供查询的最新信息	55
<a href="#">XTPTradeReport</a>	
报单成交结构体	56

## 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_trade_api.cpp</a>	
定义控制台测试应用程序的入口点 . . . . .	59
<a href="#">demo_test_trade_spi.h</a>	
Demo自定义客户端交易响应接口类 . . . . .	61
<a href="#">xoms_api_fund_struct.h</a>	
定义资金划拨相关结构体类型 . . . . .	61
<a href="#">xoms_api_struct.h</a>	
定义交易类相关数据结构 . . . . .	62
<a href="#">xquote_api_struct.h</a>	
定义行情类相关数据结构 . . . . .	63
<a href="#">xtp_api_data_type.h</a>	
定义兼容数据基本类型 . . . . .	65
<a href="#">xtp_api_struct.h</a>	
定义业务数据结构 . . . . .	74
<a href="#">xtp_api_struct_common.h</a>	
定义业务公共数据结构 . . . . .	75
<a href="#">xtp_trader_api.h</a>	
定义客户端交易接口 . . . . .	75





## Chapter 5

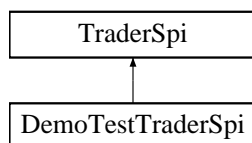
# 结构体说明

### 5.1 DemoTestTraderSpi类 参考

Demo自定义交易接口响应类

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

类 DemoTestTraderSpi 继承关系图:



#### Public 成员函数

- virtual void **OnDisconnected** (uint64\_t session\_id, int reason)  
当客户端某个连接与交易后台通信连接断开
- virtual void **OnError** (XTPRI \*error\_info)  
错误应答
- virtual void **OnOrderEvent** (XTPOrderInfo \*order\_info, XTPRI \*error\_info, uint64\_t session\_id)  
报单通知
- virtual void **OnTradeEvent** (XTPTradeReport \*trade\_info, uint64\_t session\_id)  
成交通知
- virtual void **OnQueryOrder** (XTPQueryOrderRsp \*order\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void **OnQueryTrade** (XTPQueryTradeRsp \*trade\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void **OnQueryPosition** (XTPQueryStkPositionRsp \*position, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void **OnQueryAsset** (XTPQueryAssetRsp \*asset, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void **OnQueryStructuredFund** (XTPStructuredFundInfo \*fund\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void **OnQueryFundTransfer** (XTPFundTransferNotice \*fund\_transfer\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void **OnFundTransfer** (XTPFundTransferNotice \*fund\_transfer\_info, XTPRI \*error\_info, uint64\_t session\_id)

- virtual void [OnQueryETF](#) ([XTPQueryETFBaseRsp](#) \*etf\_info, [XTPRI](#) \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryETFBasket](#) ([XTPQueryETFComponentRsp](#) \*etf\_component\_info, [XTPRI](#) \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryIPOInfoList](#) ([XTPQueryIPOTickerRsp](#) \*ipo\_info, [XTPRI](#) \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryIPOQuotaInfo](#) ([XTPQueryIPOQuotaRsp](#) \*quota\_info, [XTPRI](#) \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryOptionAuctionInfo](#) ([XTPQueryOptionAuctionInfoRsp](#) \*option\_info, [XTPRI](#) \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)

### 5.1.1 详细描述

Demo自定义交易接口响应类

作者

中泰证券股份有限公司

日期

十月 2015

### 5.1.2 成员函数说明

**5.1.2.1** virtual void [OnFundTransfer](#) ( [XTPFundTransferNotice](#) \* fund\_transfer\_info, [XTPRI](#) \* error\_info, uint64\_t session\_id ) [virtual]

资金划拨通知

参数

<i>fund_transfer_info</i>	资金划拨通知的具体信息，用户可以通过fund_transfer_info.serial_id来管理订单，通过GetClientIDByXTPID() == client_id来过滤自己的订单。
<i>error_info</i>	资金划拨订单被拒绝或者发生错误时错误代码和错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
<i>session_id</i>	资金账户对应的session_id，登录时得到

备注

当资金划拨订单有状态变化的时候，会被调用，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。所有登录了此用户的客户端都将收到此用户的资金划拨通知。

重载 [TraderSpi](#) .

**5.1.2.2** virtual void [OnQueryAsset](#) ( [XTPQueryAssetRsp](#) \* asset, [XTPRI](#) \* error\_info, int request\_id, bool is\_last, uint64\_t session\_id ) [virtual]

请求查询资金账户响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询ETF清单文件的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询ETF股票篮的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询资金划拨订单响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询今日新股申购信息列表的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询用户新股申购额度信息的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询期权合约的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询报单响应

参数

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

备注

由于支持分时段查询，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询投资者持仓响应

参数

<i>position</i>	查询到的一只股票的持仓情况
<i>error_info</i>	查询账户持仓发生错误时返回的错误信息，当error_info为空，或者error_info.error_id↔id为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的session_id，登录时得到

备注

由于用户可能持有多个股票，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询分级基金信息响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询成交响应

参数

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

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

#### 备注

由于支持分时段查询，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

该类的文档由以下文件生成:

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

## 5.2 OrderBookStruct结构体 参考

### 定单簿

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

### 成员变量

- [XTP\\_EXCHANGE\\_TYPE](#) `exchange_id`  
交易所代码
- `char` [ticker](#) [[XTP\\_TICKER\\_LEN](#)]  
合约代码（不包含交易所信息），不带空格，以'0'结尾
- `double` [last\\_price](#)  
最新价
- `int64_t` [qty](#)  
数量，为总成交量
- `double` [turnover](#)  
成交金额，为总成交金额
- `int64_t` [trades\\_count](#)  
成交笔数
- `double` [bid](#) [10]  
十档申买价
- `double` [ask](#) [10]  
十档申卖价
- `int64_t` [bid\\_qty](#) [10]  
十档申买量
- `int64_t` [ask\\_qty](#) [10]  
十档申卖量
- `int64_t` [data\\_time](#)  
时间类

### 5.2.1 详细描述

#### 定单簿

该结构体的文档由以下文件生成:

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



## 5.3 TraderApi类 参考

交易接口类

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

### Public 成员函数

- virtual void [Release](#) ()=0
- virtual const char \* [GetTradingDay](#) ()=0
- virtual void [RegisterSpi](#) (TraderSpi \*spi)=0
- virtual XTPRI \* [GetApiLastError](#) ()=0
- virtual const char \* [GetApiVersion](#) ()=0
- virtual uint8\_t [GetClientIDByXTPID](#) (uint64\_t order\_xtp\_id)=0
- virtual const char \* [GetAccountByXTPID](#) (uint64\_t order\_xtp\_id)=0
- virtual void [SubscribePublicTopic](#) (XTP\_TE\_RESUME\_TYPE resume\_type)=0
- virtual void [SetSoftwareVersion](#) (const char \*version)=0
- virtual void [SetSoftwareKey](#) (const char \*key)=0
- virtual void [SetHeartBeatInterval](#) (uint32\_t interval)=0
- virtual uint64\_t [Login](#) (const char \*ip, int port, const char \*user, const char \*password, XTP\_PROTOCOL↵  
\_TYPE sock\_type)=0
- virtual int [Logout](#) (uint64\_t session\_id)=0
- virtual uint64\_t [InsertOrder](#) (XTPOrderInsertInfo \*order, uint64\_t session\_id)=0
- virtual uint64\_t [CancelOrder](#) (const uint64\_t order\_xtp\_id, uint64\_t session\_id)=0
- virtual int [QueryOrderByXTPID](#) (const uint64\_t order\_xtp\_id, uint64\_t session\_id, int request\_id)=0
- virtual int [QueryOrders](#) (const XTPQueryOrderReq \*query\_param, uint64\_t session\_id, int request\_id)=0
- virtual int [QueryTradesByXTPID](#) (const uint64\_t order\_xtp\_id, uint64\_t session\_id, int request\_id)=0
- virtual int [QueryTrades](#) (XTPQueryTraderReq \*query\_param, uint64\_t session\_id, int request\_id)=0
- virtual int [QueryPosition](#) (const char \*ticker, uint64\_t session\_id, int request\_id)=0
- virtual int [QueryAsset](#) (uint64\_t session\_id, int request\_id)=0
- virtual int [QueryStructuredFund](#) (XTPQueryStructuredFundInfoReq \*query\_param, uint64\_t session\_id, int↵  
request\_id)=0
- virtual uint64\_t [FundTransfer](#) (XTPFundTransferReq \*fund\_transfer, uint64\_t session\_id)=0
- virtual int [QueryFundTransfer](#) (XTPQueryFundTransferLogReq \*query\_param, uint64\_t session\_id, int↵  
request\_id)=0
- virtual int [QueryETF](#) (XTPQueryETFBBaseReq \*query\_param, uint64\_t session\_id, int request\_id)=0
- virtual int [QueryETFTickerBasket](#) (XTPQueryETFComponentReq \*query\_param, uint64\_t session\_id, int↵  
request\_id)=0
- virtual int [QueryIPOInfoList](#) (uint64\_t session\_id, int request\_id)=0
- virtual int [QueryIPOQuotaInfo](#) (uint64\_t session\_id, int request\_id)=0
- virtual int [QueryOptionAuctionInfo](#) (XTPQueryOptionAuctionInfoReq \*query\_param, uint64\_t session\_id, int↵  
request\_id)=0

### 静态 Public 成员函数

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

#### 5.3.1 详细描述

交易接口类

作者

中泰证券股份有限公司

日期

十月 2015

### 5.3.2 成员函数说明

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

报单操作请求

返回

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

参数

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

备注

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

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

创建TraderApi

参数

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

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

资金划拨请求

返回

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

参数

<i>session_id</i>	资金账户对应的session_id,登录时得到
<i>fund_transfer</i>	用户进行资金划拨提交的具体请求信息

#### 5.3.2.4 virtual const char\* GetAccountByXTPID ( uint64\_t order\_xtp\_id ) [pure virtual]

通过报单在xtp系统中的ID获取相关资金账户名

返回

返回资金账户名

参数

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

备注

只有资金账户登录成功后,才能得到正确的信息

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

获取API的系统错误

返回

返回的错误信息,可以在Login、InsertOrder、CancelOrder返回值为0时调用,获取失败的原因

备注

可以在调用api接口失败时调用,例如login失败时

#### 5.3.2.6 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

#### 5.3.2.7 virtual uint8\_t GetClientIDByXTPID ( uint64\_t order\_xtp\_id ) [pure virtual]

通过报单在xtp系统中的ID获取下单的客户端id

返回

返回客户端id,可以用此方法过滤自己下的订单

参数

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

备注

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

### 5.3.2.8 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

### 5.3.2.9 virtual uint64\_t InsertOrder ( XTPOrderInsertInfo \* order, uint64\_t session\_id ) [pure virtual]

报单录入请求

返回

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

参数

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

备注

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

### 5.3.2.10 virtual uint64\_t 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

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

参数

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

备注

此函数为同步阻塞式，不需要异步等待登录成功，当函数返回即可进行后续操作，此api可支持多个账户连接，但是同一个账户同一个client\_id只能有一个session连接，后面的登录在前一个session存续期间，无法连接

### 5.3.2.11 virtual int Logout ( uint64\_t session\_id ) [pure virtual]

登出请求

返回

登出是否成功，“0”表示登出成功，“-1”表示登出失败

参数

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

### 5.3.2.12 virtual int QueryAsset ( uint64\_t session\_id, int request\_id ) [pure virtual]

请求查询资产

返回

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

参数

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

### 5.3.2.13 virtual int QueryETF ( XTPQueryETFBaseReq \* query\_param, uint64\_t session\_id, int request\_id ) [pure virtual]

请求查询ETF清单文件

返回

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

参数

<i>query_param</i>	需要查询的ETF清单文件的筛选条件，其中合约代码可以为空，则默认所有存在的ETF合约代码，market字段也可以为初始值，则默认所有市场的ETF合约
--------------------	--

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

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

请求查询ETF股票篮

返回

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

参数

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

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

请求查询资金划拨

返回

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

参数

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

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

请求查询今日新股申购信息列表

返回

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

参数

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

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

请求查询用户新股申购额度信息

返回

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

参数

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

**5.3.2.18** `virtual int QueryOptionAuctionInfo ( XTPQueryOptionAuctionInfoReq * query_param, uint64_t session_id, int request_id ) [pure virtual]`

请求查询期权合约

返回

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

参数

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

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

根据报单ID请求查询报单

返回

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

参数

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

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

请求查询报单

返回

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

参数

<i>query_param</i>	需要查询的订单相关筛选条件, 其中合约代码可以为空, 则默认所有存在的合约代码, 如果不为空, 请不带空格, 并以'\0'结尾, 其中起始时间格式为YYYYMMDDHH↵HMMSSsss, 为0则默认当前交易日0点, 结束时间格式为YYYYMMDDHHHMMSSsss, 为0则默认当前时间
--------------------	---

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

备注

该方法支持分时段查询，如果股票代码为空，则默认查询时间段内的所有报单，否则查询时间段内所有跟股票代码相关的报单，此函数查询出的结果可能对应多个查询结果响应

**5.3.2.21** `virtual int QueryPosition ( const char * ticker, uint64_t session_id, int request_id ) [pure virtual]`

请求查询投资者持仓

返回

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

参数

<i>ticker</i>	需要查询的持仓合约代码，可以为空，如果不为空，请不带空格，并以'\0'结尾
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

该方法如果用户提供了合约代码，则会查询此合约的持仓信息，如果合约代码为空，则默认查询所有持仓信息

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

请求查询分级基金

返回

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

参数

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

备注

此函数查询出的结果可能对应多个查询结果响应

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

请求查询已成交

返回

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



参数

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

备注

该方法支持分时段查询，如果股票代码为空，则默认查询时间段内的所有成交回报，否则查询时间段内所有跟股票代码相关的成交回报，此函数查询出的结果可能对应多个查询结果响应

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

根据委托编号请求查询相关成交

返回

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

参数

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

备注

此函数查询出的结果可能对应多个查询结果响应

**5.3.2.25** `virtual void RegisterSpi ( TraderSpi * spi ) [pure virtual]`

注册回调接口

参数

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

**5.3.2.26** `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

**5.3.2.27** `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

### 5.3.2.28 virtual void SetSoftwareKey ( const char \* *key* ) [pure virtual]

设置软件开发Key

参数

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

备注

此函数必须在Login之前调用

### 5.3.2.29 virtual void SetSoftwareVersion ( const char \* *version* ) [pure virtual]

设置软件开发版本号

参数

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

备注

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

### 5.3.2.30 virtual void SubscribePublicTopic ( XTP\_TE\_RESUME\_TYPE *resume\_type* ) [pure virtual]

订阅公共流。

参数

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

备注

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

该类的文档由以下文件生成:

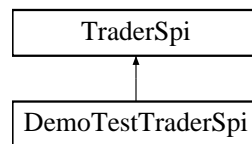
- [xtp\\_trader\\_api.h](#)

## 5.4 TraderSpi类 参考

交易接口响应类

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

类 TraderSpi 继承关系图:



### Public 成员函数

- virtual void [OnDisconnected](#) (uint64\_t session\_id, int reason)
- virtual void [OnError](#) (XTPRI \*error\_info)
- virtual void [OnOrderEvent](#) (XTPOrderInfo \*order\_info, XTPRI \*error\_info, uint64\_t session\_id)
- virtual void [OnTradeEvent](#) (XTPTradeReport \*trade\_info, uint64\_t session\_id)
- virtual void [OnCancelOrderError](#) (XTPOrderCancelInfo \*cancel\_info, XTPRI \*error\_info, uint64\_t session\_id)
- virtual void [OnQueryOrder](#) (XTPQueryOrderRsp \*order\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryTrade](#) (XTPQueryTradeRsp \*trade\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryPosition](#) (XTPQueryStkPositionRsp \*position, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryAsset](#) (XTPQueryAssetRsp \*asset, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryStructuredFund](#) (XTPStructuredFundInfo \*fund\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryFundTransfer](#) (XTPFundTransferNotice \*fund\_transfer\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnFundTransfer](#) (XTPFundTransferNotice \*fund\_transfer\_info, XTPRI \*error\_info, uint64\_t session\_id)
- virtual void [OnQueryETF](#) (XTPQueryETFBaseRsp \*etf\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryETFBasket](#) (XTPQueryETFComponentRsp \*etf\_component\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryIPOInfoList](#) (XTPQueryIPOTickerRsp \*ipo\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryIPOQuotaInfo](#) (XTPQueryIPOQuotaRsp \*quota\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)
- virtual void [OnQueryOptionAuctionInfo](#) (XTPQueryOptionAuctionInfoRsp \*option\_info, XTPRI \*error\_info, int request\_id, bool is\_last, uint64\_t session\_id)

### 5.4.1 详细描述

交易接口响应类

作者

中泰证券股份有限公司

日期

十月 2015

## 5.4.2 成员函数说明

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

撤单出错响应

参数

<i>cancel_info</i>	撤单具体信息，包括撤单的order_cancel_xtp_id和待撤单的order_xtp_id
<i>error_info</i>	撤单被拒绝或者发生错误时错误代码和错误信息，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线，当error_info为空，或者error_info.error_id为0时，表明没有错误
<i>session_id</i>	资金账户对应的session_id，登录时得到

备注

此响应只会在撤单发生错误时被回调

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

当客户端的某个连接与交易后台通信连接断开时，该方法被调用。

参数

<i>reason</i>	错误原因，请与错误代码表对应
<i>session_id</i>	资金账户对应的session_id，登录时得到

备注

用户主动调用logout导致的断线，不会触发此函数。api不会自动重连，当断线发生时，请用户自行选择后续操作，可以在此函数中调用Login重新登录，并更新session\_id，此时用户收到的数据跟断线之前是连续的

被 [DemoTestTraderSpi](#) 重载。

**5.4.2.3** `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 [DemoTestTraderSpi](#) 重载。

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

资金划拨通知

## 参数

<i>fund_transfer_info</i>	资金划拨通知的具体信息，用户可以通过 <i>fund_transfer_info.serial_id</i> 来管理订单，通过 <i>GetClientIDByXTPID() == client_id</i> 来过滤自己的订单。
<i>error_info</i>	资金划拨订单被拒绝或者发生错误时错误代码和错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到

## 备注

当资金划拨订单有状态变化的时候，会被调用，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。所有登录了此用户的客户端都将收到此用户的资金划拨通知。

被 [DemoTestTraderSpi](#) 重载。

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

## 报单通知

## 参数

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

## 备注

每次订单状态更新时，都会被调用，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线，在订单未成交、全部成交、全部撤单、部分撤单、已拒绝这些状态时会有响应，对于部分成交的情况，请由订单的成交回报来自行确认。所有登录了此用户的客户端都将收到此用户的订单响应

被 [DemoTestTraderSpi](#) 重载。

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

请求查询资金账户响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

## 参数

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

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询ETF清单文件的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线  
参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询ETF股票篮的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线  
参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询资金划拨订单响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询今日新股申购信息列表的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询用户新股申购额度信息的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询期权合约的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询报单响应

参数

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

备注

由于支持分时段查询，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询投资者持仓响应



参数

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

备注

由于用户可能持有多个股票，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询分级基金信息响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询成交响应

参数

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

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

## 备注

由于支持分时段查询，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

## 成交通知

## 参数

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

## 备注

订单有成交发生的时候，会被调用，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。所有登录了此用户的客户端都将收到此用户的成交回报。相关订单为部成状态，需要用户通过成交回报的成交数量来确定，OnOrderEvent()不会推送部成状态。

被 [DemoTestTraderSpi](#) 重载.

该类的文档由以下文件生成:

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

## 5.5 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
- [XTPTOrderTypeType](#) [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_balance`  
昨日余额
- `double balance`  
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"50ETF"就是900000
- int32\_t [subscribe\\_status](#)  
是否允许申购,1-允许,0-禁止
- int32\_t [redemption\\_status](#)  
是否允许赎回,1-允许,0-禁止
- double [max\\_cash\\_ratio](#)  
最大现金替代比例,小于1的数值 TODO 是否采用double
- double [estimate\\_amount](#)  
T日预估金额
- double [cash\\_component](#)  
T-X日现金差额
- double [net\\_value](#)  
基金单位净值
- double [total\\_amount](#)  
最小申赎单位净值总金额= $net\_value * unit$

### 5.15.1 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](#)  
成分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YYYYMMDDHHMSSsss，为0则默认当前交易日0点
- [int64\\_t end\\_time](#)  
结束时间，格式为YYYYMMDDHHMSSsss，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'0'结尾
- [char ticker\\_name \[XTP\\_TICKER\\_NAME\\_LEN\]](#)  
合约名称
- [XTP\\_TICKER\\_TYPE ticker\\_type](#)  
合约类型
- [double pre\\_close\\_price](#)  
昨收盘
- [double upper\\_limit\\_price](#)  
涨停板价

- double [lower\\_limit\\_price](#)  
跌停板价
- double [price\\_tick](#)  
最小变动价位
- int32\_t [buy\\_qty\\_unit](#)  
合约最小交易量(买)
- int32\_t [sell\\_qty\\_unit](#)  
合约最小交易量(卖)

### 5.28.1 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](#)  
最小拆分数量
- [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 XTPTickerPriceInfo结构体 参考

供查询的最新信息

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

成员变量

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

### 5.35.1 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\_trade\_api.cpp 文件参考

定义控制台测试应用程序的入口点

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

函数

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

#### 6.1.1 详细描述

定义控制台测试应用程序的入口点

作者

中泰证券股份有限公司

#### 6.1.2 函数说明

##### 6.1.2.1 int main ( )

```
int main() {
```

测试Demo入口函数

```
int client_id = 1;//客户端标识
```

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

```
//初始化UserApi
```

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

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

```

user_api_pointer->SetSoftwareKey("xxxxxxxxxxxxxxxxxxxxxx");//设定用户开发软件Key, 用户申请开户时给予, 以'\0'结尾
user_api_pointer->SetSoftwareVersion("1.1.0");//设定软件的开发版本号, 非api版本号
user_api_pointer->SetHeartBeatInterval(15);//设置心跳超时时间间隔, 单位为秒
DemoTestTraderSpi* user_spi_pointer = new DemoTestTraderSpi();// 创建响应类实例
user_api_pointer->RegisterSpi(user_spi_pointer); // 注册事件类
uint64_t temp_session_ = user_api_pointer->Login(server_ip.c_str(), server_port, username.c_str(), password.c_str(), XTP_PROTOCOL_TCP);//登陆交易服务器
if (temp_session_ != 0)
{
    int order_client_id = 1;//用户自定义用于标识本地订单的编号, 可以任意
    //下单
    XTPOrderInsertInfo orderInsert;
    memset(&orderInsert, 0, sizeof(XTPOrderInsertInfo));
    orderInsert.order_client_id = order_client_id++;//用户自定义, 用来标识订单, 可以不填
    std::string ticker("000002");
    strcpy(orderInsert.ticker, ticker.c_str());
    orderInsert.exchange_id = (XTP_EXCHANGE_TYPE)2; orderInsert.price = 17.5;
    orderInsert.quantity = 200;
    orderInsert.side = (XTP_SIDE_TYPE)1;
    orderInsert.price_type = (XTP_PRICE_TYPE)3;
    orderInsert.business_type = (XTP_BUSINESS_TYPE_CASH)0;
    orderInsert.position_effect = (XTP_POSITION_EFFECT_INIT)0;//期权业务使用的字段, 普通业务请用0
    //返回的xtp_id需要记录下来, 与服务器交互的时候, 所有对订单的操作由xtp_id唯一确定
    uint64_t insert_xtp_id = user_api_pointer->InsertOrder(&orderInsert,temp_session_);
    if (insert_xtp_id == 0)
    {
        //下单失败

        XTPRI* error_info = user_api_pointer->GetApiLastError(); //下单失败时的错误原因代码
    }
    //如果需要撤单 //返回的xtp_id需要记录下来, 与服务器交互的时候, 所有对订单的操作由xtp_id唯一确定
    uint64_t cancel_xtp_id = user_api_pointer->CancelOrder(insert_xtp_id,temp_session_);
    if (cancel_xtp_id == 0)
    {
        //撤单失败

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

```

```
std::cout << "Login to server error, " << error_info->error_id << " : " << error_info->error_msg << std::endl;
}
return 0;
}
```

## 6.2 demo\_test\_trade\_spi.h 文件参考

Demo自定义客户端交易响应接口类

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

结构体

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

### 6.2.1 详细描述

Demo自定义客户端交易响应接口类

作者

中泰证券股份有限公司

## 6.3 xoms\_api\_fund\_struct.h 文件参考

定义资金划拨相关结构体类型

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

结构体

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

宏定义

- #define [XTP\\_ACCOUNT\\_PASSWORD\\_LEN](#) 64  
用户资金账户的密码字符串长度

类型定义

- typedef struct [XTPFundTransferNotice](#) [XTPFundTransferAck](#)  
用户资金划转请求的响应-复用资金通知结构体

### 6.3.1 详细描述

定义资金划拨相关结构体类型

作者

中泰证券股份有限公司

## 6.4 xoms\_api\_struct.h 文件参考

定义交易类相关数据结构

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

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 [XTPQueryReportByExeclIdReq](#)  
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 [XTPQueryETFBBaseReq](#)
- struct [XTPQueryETFBBaseRsp](#)  
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](#)

- 定单簿
- 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  
定单簿
- 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
- enum `XTP_TE_RESUME_TYPE` { `XTP_TERT_RESTART` = 0, `XTP_TERT_RESUME`, `XTP_TERT_QUICK` }  
`XTP_TE_RESUME_TYPE`是公有流（订单响应、成交回报）重传方式
- enum `ETF_REPLACE_TYPE` {  
`ERT_CASH_FORBIDDEN` = 0, `ERT_CASH_OPTIONAL`, `ERT_CASH_MUST`, `ERT_CASH_RECOMPUTE_INTER_SZ`,  
`ERT_CASH_MUST_INTER_SZ`, `ERT_CASH_RECOMPUTE_INTER_OTHER`, `ERT_CASH_MUST_INTER_OTHER`, `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_TYPE` { `XTP_OPT_EXERCISE_TYPE_EUR` = 1, `XTP_OPT_EXERCISE_TYPE_AME` = 2 }  
`XTP_OPT_EXERCISE_TYPE_TYPE` 是一个行权方式类型
- enum `XTP_POSITION_DIRECTION_TYPE` { `XTP_POSITION_DIRECTION_NET` = 0, `XTP_POSITION_DIRECTION_LONG`, `XTP_POSITION_DIRECTION_SHORT`, `XTP_POSITION_DIRECTION_COVERED` }  
`XTP_POSITION_DIRECTION_TYPE` 是一个持仓方向类型

### 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\_trader\_api.h 文件参考

定义客户端交易接口

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

结构体

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

### 6.9.1 详细描述

定义客户端交易接口

作者

中泰证券股份有限公司



# Index

CancelOrder  
    XTP::API::TraderApi, 18

CreateTraderApi  
    XTP::API::TraderApi, 18

demo\_test\_trade\_api.cpp, 59  
    main, 59

demo\_test\_trade\_spi.h, 61

DemoTestTraderSpi, 9  
    OnFundTransfer, 10  
    OnQueryAsset, 10  
    OnQueryETF, 11  
    OnQueryETFBasket, 11  
    OnQueryFundTransfer, 11  
    OnQueryIPOInfoList, 13  
    OnQueryIPOQuotaInfo, 13  
    OnQueryOptionAuctionInfo, 14  
    OnQueryOrder, 14  
    OnQueryPosition, 14  
    OnQueryStructuredFund, 15  
    OnQueryTrade, 15

EPT\_INVALID  
    xtp\_api\_data\_type.h, 70

ERT\_CASH\_FORBIDDEN  
    xtp\_api\_data\_type.h, 69

ERT\_CASH\_MUST  
    xtp\_api\_data\_type.h, 69

ERT\_CASH\_MUST\_INTER\_OTHER  
    xtp\_api\_data\_type.h, 70

ERT\_CASH\_MUST\_INTER\_SZ  
    xtp\_api\_data\_type.h, 70

ERT\_CASH\_OPTIONAL  
    xtp\_api\_data\_type.h, 69

ERT\_CASH\_RECOMPUTE\_INTER\_OTHER  
    xtp\_api\_data\_type.h, 70

ERT\_CASH\_RECOMPUTE\_INTER\_SZ  
    xtp\_api\_data\_type.h, 69

ETF\_REPLACE\_TYPE  
    xtp\_api\_data\_type.h, 69

FundTransfer  
    XTP::API::TraderApi, 18

GetAccountByXTPID  
    XTP::API::TraderApi, 19

GetApiLastError  
    XTP::API::TraderApi, 19

GetApiVersion  
    XTP::API::TraderApi, 19

GetClientIDByXTPID  
    XTP::API::TraderApi, 19

GetTradingDay  
    XTP::API::TraderApi, 20

InsertOrder  
    XTP::API::TraderApi, 20

Login  
    XTP::API::TraderApi, 20

Logout  
    XTP::API::TraderApi, 21

main  
    demo\_test\_trade\_api.cpp, 59

OnCancelOrderError  
    XTP::API::TraderSpi, 28

OnDisconnected  
    XTP::API::TraderSpi, 28

OnError  
    XTP::API::TraderSpi, 28

OnFundTransfer  
    DemoTestTraderSpi, 10  
    XTP::API::TraderSpi, 28

OnOrderEvent  
    XTP::API::TraderSpi, 29

OnQueryAsset  
    DemoTestTraderSpi, 10  
    XTP::API::TraderSpi, 29

OnQueryETF  
    DemoTestTraderSpi, 11  
    XTP::API::TraderSpi, 30

OnQueryETFBasket  
    DemoTestTraderSpi, 11  
    XTP::API::TraderSpi, 30

OnQueryFundTransfer  
    DemoTestTraderSpi, 11  
    XTP::API::TraderSpi, 30

OnQueryIPOInfoList  
    DemoTestTraderSpi, 13  
    XTP::API::TraderSpi, 31

OnQueryIPOQuotaInfo  
    DemoTestTraderSpi, 13  
    XTP::API::TraderSpi, 31

OnQueryOptionAuctionInfo  
    DemoTestTraderSpi, 14  
    XTP::API::TraderSpi, 32

OnQueryOrder  
    DemoTestTraderSpi, 14

- XTP::API::TraderSpi, 32
- OnQueryPosition
  - DemoTestTraderSpi, 14
  - XTP::API::TraderSpi, 32
- OnQueryStructuredFund
  - DemoTestTraderSpi, 15
  - XTP::API::TraderSpi, 33
- OnQueryTrade
  - DemoTestTraderSpi, 15
  - XTP::API::TraderSpi, 33
- OnTradeEvent
  - XTP::API::TraderSpi, 34
- OrderBookStruct, 16
- QueryAsset
  - XTP::API::TraderApi, 21
- QueryETF
  - XTP::API::TraderApi, 21
- QueryETFTickerBasket
  - XTP::API::TraderApi, 22
- QueryFundTransfer
  - XTP::API::TraderApi, 22
- QueryIPOInfoList
  - XTP::API::TraderApi, 22
- QueryIPOQuotaInfo
  - XTP::API::TraderApi, 22
- QueryOptionAuctionInfo
  - XTP::API::TraderApi, 23
- QueryOrderByXTPID
  - XTP::API::TraderApi, 23
- QueryOrders
  - XTP::API::TraderApi, 23
- QueryPosition
  - XTP::API::TraderApi, 24
- QueryStructuredFund
  - XTP::API::TraderApi, 24
- QueryTrades
  - XTP::API::TraderApi, 24
- QueryTradesByXTPID
  - XTP::API::TraderApi, 25
- RegisterSpi
  - XTP::API::TraderApi, 25
- Release
  - XTP::API::TraderApi, 25
- SetHeartBeatInterval
  - XTP::API::TraderApi, 25
- SetSoftwareKey
  - XTP::API::TraderApi, 26
- SetSoftwareVersion
  - XTP::API::TraderApi, 26
- SubscribePublicTopic
  - XTP::API::TraderApi, 26
- trade\_flag
  - XTPTickByTickTrade, 55
- TraderApi, 17
- TraderSpi, 27
- XTP::API::TraderApi
  - CancelOrder, 18
  - CreateTraderApi, 18
  - FundTransfer, 18
  - GetAccountByXTPID, 19
  - GetApiLastError, 19
  - GetApiVersion, 19
  - GetClientIDByXTPID, 19
  - GetTradingDay, 20
  - InsertOrder, 20
  - Login, 20
  - Logout, 21
  - QueryAsset, 21
  - QueryETF, 21
  - QueryETFTickerBasket, 22
  - QueryFundTransfer, 22
  - QueryIPOInfoList, 22
  - QueryIPOQuotaInfo, 22
  - QueryOptionAuctionInfo, 23
  - QueryOrderByXTPID, 23
  - QueryOrders, 23
  - QueryPosition, 24
  - QueryStructuredFund, 24
  - QueryTrades, 24
  - QueryTradesByXTPID, 25
  - RegisterSpi, 25
  - Release, 25
  - SetHeartBeatInterval, 25
  - SetSoftwareKey, 26
  - SetSoftwareVersion, 26
  - SubscribePublicTopic, 26
- XTP::API::TraderSpi
  - OnCancelOrderError, 28
  - OnDisconnected, 28
  - OnError, 28
  - OnFundTransfer, 28
  - OnOrderEvent, 29
  - OnQueryAsset, 29
  - OnQueryETF, 30
  - OnQueryETFBasket, 30
  - OnQueryFundTransfer, 30
  - OnQueryIPOInfoList, 31
  - OnQueryIPOQuotaInfo, 31
  - OnQueryOptionAuctionInfo, 32
  - OnQueryOrder, 32
  - OnQueryPosition, 32
  - OnQueryStructuredFund, 33
  - OnQueryTrade, 33
  - OnTradeEvent, 34
- XTP\_ACCOUNT\_CREDIT
  - xtp\_api\_data\_type.h, 70
- XTP\_ACCOUNT\_DERIVE
  - xtp\_api\_data\_type.h, 70
- XTP\_ACCOUNT\_NORMAL
  - xtp\_api\_data\_type.h, 70
- XTP\_ACCOUNT\_TYPE
  - xtp\_api\_data\_type.h, 70
- XTP\_ACCOUNT\_UNKNOWN

[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_ALLOTMENT  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_CASH  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_DESIGNATION  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE ETF  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_EXECUTE  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_FREEZE  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_IPOS  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_MARGIN  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_MONEY\_FUND  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_OPTION  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_REPO  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_STRUCTURED\_FUND\_PURCHASE\_REDEMPTION  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_STRUCTURED\_FUND\_SPLIT\_MERGE  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_BUSINESS\_TYPE\_UNKNOWN  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_EXCHANGE\_SH  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_EXCHANGE\_SZ  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_EXCHANGE\_TYPE  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_EXCHANGE\_UNKNOWN  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_FUND\_OPER\_FAILED  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_FUND\_OPER\_PROCESSING  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_FUND\_OPER\_STATUS  
[xtp\\_api\\_data\\_type.h, 70](#)  
 XTP\_FUND\_OPER\_SUBMITTED  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_FUND\_OPER\_SUCCESS  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_FUND\_OPER\_UNKNOWN  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_FUND\_TRANSFER\_IN  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_FUND\_TRANSFER\_OUT  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_FUND\_TRANSFER\_TYPE  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_FUND\_TRANSFER\_UNKNOWN  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_LOG\_LEVEL  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_LOG\_LEVEL\_DEBUG  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_LOG\_LEVEL\_ERROR  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_LOG\_LEVEL\_FATAL  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_LOG\_LEVEL\_INFO  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_LOG\_LEVEL\_TRACE  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_LOG\_LEVEL\_WARNING  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_MARKET\_TYPE  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_MARKETDATA\_ACTUAL  
[xquote\\_api\\_struct.h, 64](#)  
 XTP\_MARKETDATA\_OPTION  
[xquote\\_api\\_struct.h, 64](#)  
 XTP\_MARKETDATA\_TYPE  
[xquote\\_api\\_struct.h, 64](#)  
 XTP\_MKT\_INIT  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_MKT\_SH\_A  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_MKT\_SZ\_A  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_MKT\_UNKNOWN  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_OPT\_CALL  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_OPT\_CALL\_OR\_PUT\_TYPE  
[xtp\\_api\\_data\\_type.h, 71](#)  
 XTP\_OPT\_EXERCISE\_TYPE\_AME  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_OPT\_EXERCISE\_TYPE\_EUR  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_OPT\_EXERCISE\_TYPE\_TYPE  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_OPT\_PUT  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_ACTION\_STATUS\_ACCEPTED  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_ACTION\_STATUS\_REJECTED  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_ACTION\_STATUS\_SUBMITTED  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_ACTION\_STATUS\_TYPE  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_STATUS\_ALLTRADED  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_STATUS\_CANCELED  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_STATUS\_INIT

[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_STATUS\_NOTRADEQUEUEING  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_STATUS\_PARTTRADEDNOTQUEUEING  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_STATUS\_PARTTRADEDQUEUEING  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_STATUS\_REJECTED  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_STATUS\_TYPE  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_STATUS\_UNKNOWN  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_ACCEPTED  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_REJECTED  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_SUBMITTED  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_ACCEPTED  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_REJECTED  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_SUBMITTED  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_ORDER\_SUBMIT\_STATUS\_TYPE  
[xtp\\_api\\_data\\_type.h, 72](#)  
 XTP\_POSITION\_DIRECTION\_COVERED  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_POSITION\_DIRECTION\_LONG  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_POSITION\_DIRECTION\_NET  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_POSITION\_DIRECTION\_SHORT  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_POSITION\_DIRECTION\_TYPE  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_ALL\_OR\_CANCEL  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_BEST5\_OR\_CANCEL  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_BEST5\_OR\_LIMIT  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_BEST\_OR\_CANCEL  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_FORWARD\_BEST  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_LIMIT  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_LIMIT\_OR\_CANCEL  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_REVERSE\_BEST\_LIMIT  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_TYPE  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PRICE\_TYPE\_UNKNOWN  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PROTOCOL\_TCP  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PROTOCOL\_TYPE  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_PROTOCOL\_UDP  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_SPLIT\_MERGE\_STATUS  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_SPLIT\_MERGE\_STATUS\_ALLOW  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_SPLIT\_MERGE\_STATUS\_FORBIDDEN  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_MERGE  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_SPLIT  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_TBT\_ENTRUST  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TBT\_TRADE  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TBT\_TYPE  
[xtp\\_api\\_data\\_type.h, 73](#)  
 XTP\_TE\_RESUME\_TYPE  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TERT\_QUICK  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TERT\_RESTART  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TERT\_RESUME  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TICKER\_TYPE  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TICKER\_TYPE\_BOND  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TICKER\_TYPE\_FUND  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TICKER\_TYPE\_INDEX  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TICKER\_TYPE\_OPTION  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TICKER\_TYPE\_STOCK  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTP\_TICKER\_TYPE\_UNKNOWN  
[xtp\\_api\\_data\\_type.h, 74](#)  
 XTPFundTransferNotice, [34](#)  
 XTPFundTransferReq, [35](#)  
 XTPMarketDataOptionExData, [35](#)  
 XTPMarketDataStockExData, [36](#)  
 XTPMarketDataStruct, [37](#)  
 XTPOrderCancelInfo, [39](#)  
 XTPOrderInfo, [39](#)  
 XTPOrderInsertInfo, [41](#)



- XTPQueryAssetRsp, 41
- XTPQueryETFBaseReq, 43
- XTPQueryETFBaseRsp, 43
- XTPQueryETFComponentReq, 44
- XTPQueryETFComponentRsp, 44
- XTPQueryFundTransferLogReq, 45
- XTPQueryIPOQuotaRsp, 45
- XTPQueryIPOTickerRsp, 46
- XTPQueryOptionAuctionInfoReq, 46
- XTPQueryOptionAuctionInfoRsp, 47
- XTPQueryOrderReq, 48
- XTPQueryReportByExecIdReq, 49
- XTPQueryStkPositionRsp, 49
- XTPQueryStructuredFundInfoReq, 50
- XTPQueryTraderReq, 51
- XTPQuoteStaticInfo, 51
- XTPRspInfoStruct, 52
- XTPSpecificTickerStruct, 52
- XTPStructuredFundInfo, 53
- XTPTickByTickEntrust, 53
- XTPTickByTickStruct, 54
- XTPTickByTickTrade, 55
  - trade\_flag, 55
- XTPTickerPriceInfo, 55
- XTPTTradeReport, 56
- xoms\_api\_fund\_struct.h, 61
- xoms\_api\_struct.h, 62
- xquote\_api\_struct.h, 63
  - XTP\_MARKETDATA\_ACTUAL, 64
  - XTP\_MARKETDATA\_OPTION, 64
  - XTP\_MARKETDATA\_TYPE, 64
- xtp\_api\_data\_type.h, 65
  - EPT\_INVALID, 70
  - ERT\_CASH\_FORBIDDEN, 69
  - ERT\_CASH\_MUST, 69
  - ERT\_CASH\_MUST\_INTER\_OTHER, 70
  - ERT\_CASH\_MUST\_INTER\_SZ, 70
  - ERT\_CASH\_OPTIONAL, 69
  - ERT\_CASH\_RECOMPUTE\_INTER\_OTHER, 70
  - ERT\_CASH\_RECOMPUTE\_INTER\_SZ, 69
  - ETF\_REPLACE\_TYPE, 69
  - XTP\_ACCOUNT\_CREDIT, 70
  - XTP\_ACCOUNT\_DERIVE, 70
  - XTP\_ACCOUNT\_NORMAL, 70
  - XTP\_ACCOUNT\_TYPE, 70
  - XTP\_ACCOUNT\_UNKNOWN, 70
  - XTP\_BUSINESS\_TYPE, 70
  - XTP\_BUSINESS\_TYPE\_ALLOTMENT, 70
  - XTP\_BUSINESS\_TYPE\_CASH, 70
  - XTP\_BUSINESS\_TYPE\_DESIGNATION, 70
  - XTP\_BUSINESS\_TYPE ETF, 70
  - XTP\_BUSINESS\_TYPE\_EXECUTE, 70
  - XTP\_BUSINESS\_TYPE\_FREEZE, 70
  - XTP\_BUSINESS\_TYPE\_IPOS, 70
  - XTP\_BUSINESS\_TYPE\_MARGIN, 70
  - XTP\_BUSINESS\_TYPE\_MONEY\_FUND, 70
  - XTP\_BUSINESS\_TYPE\_OPTION, 70
  - XTP\_BUSINESS\_TYPE\_REPO, 70
  - XTP\_BUSINESS\_TYPE\_STRUCTURED\_FUND↵  
PURCHASE\_REDEMPTION, 70
  - XTP\_BUSINESS\_TYPE\_STRUCTURED\_FUND↵  
SPLIT\_MERGE, 70
  - XTP\_BUSINESS\_TYPE\_UNKNOWN, 70
  - XTP\_EXCHANGE\_SH, 70
  - XTP\_EXCHANGE\_SZ, 70
  - XTP\_EXCHANGE\_TYPE, 70
  - XTP\_EXCHANGE\_UNKNOWN, 70
  - XTP\_FUND\_OPER\_FAILED, 71
  - XTP\_FUND\_OPER\_PROCESSING, 71
  - XTP\_FUND\_OPER\_STATUS, 70
  - XTP\_FUND\_OPER\_SUBMITTED, 71
  - XTP\_FUND\_OPER\_SUCCESS, 71
  - XTP\_FUND\_OPER\_UNKNOWN, 71
  - XTP\_FUND\_TRANSFER\_IN, 71
  - XTP\_FUND\_TRANSFER\_OUT, 71
  - XTP\_FUND\_TRANSFER\_TYPE, 71
  - XTP\_FUND\_TRANSFER\_UNKNOWN, 71
  - XTP\_LOG\_LEVEL, 71
  - XTP\_LOG\_LEVEL\_DEBUG, 71
  - XTP\_LOG\_LEVEL\_ERROR, 71
  - XTP\_LOG\_LEVEL\_FATAL, 71
  - XTP\_LOG\_LEVEL\_INFO, 71
  - XTP\_LOG\_LEVEL\_TRACE, 71
  - XTP\_LOG\_LEVEL\_WARNING, 71
  - XTP\_MARKET\_TYPE, 71
  - XTP\_MKT\_INIT, 71
  - XTP\_MKT\_SH\_A, 71
  - XTP\_MKT\_SZ\_A, 71
  - XTP\_MKT\_UNKNOWN, 71
  - XTP\_OPT\_CALL, 72
  - XTP\_OPT\_CALL\_OR\_PUT\_TYPE, 71
  - XTP\_OPT\_EXERCISE\_TYPE\_AME, 72
  - XTP\_OPT\_EXERCISE\_TYPE\_EUR, 72
  - XTP\_OPT\_EXERCISE\_TYPE\_TYPE, 72
  - XTP\_OPT\_PUT, 72
  - XTP\_ORDER\_ACTION\_STATUS\_ACCEPTED, 72
  - XTP\_ORDER\_ACTION\_STATUS\_REJECTED, 72
  - XTP\_ORDER\_ACTION\_STATUS\_SUBMITTED, 72
  - XTP\_ORDER\_ACTION\_STATUS\_TYPE, 72
  - XTP\_ORDER\_STATUS\_ALLTRADED, 72
  - XTP\_ORDER\_STATUS\_CANCELED, 72
  - XTP\_ORDER\_STATUS\_INIT, 72
  - XTP\_ORDER\_STATUS\_NOTRADEQUEUEING, 72
  - XTP\_ORDER\_STATUS\_PARTTRADEDNOTQ↵  
UEUEING, 72
  - XTP\_ORDER\_STATUS\_PARTTRADEDQUEUE↵  
ING, 72
  - XTP\_ORDER\_STATUS\_REJECTED, 72
  - XTP\_ORDER\_STATUS\_TYPE, 72
  - XTP\_ORDER\_STATUS\_UNKNOWN, 72
  - XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_AC↵  
CEPTED, 73

XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_RE↵  
JECTED, [73](#)  
XTP\_ORDER\_SUBMIT\_STATUS\_CANCEL\_SU↵  
BMITTED, [73](#)  
XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_AC↵  
CEPTED, [72](#)  
XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_RE↵  
JECTED, [73](#)  
XTP\_ORDER\_SUBMIT\_STATUS\_INSERT\_SU↵  
BMITTED, [72](#)  
XTP\_ORDER\_SUBMIT\_STATUS\_TYPE, [72](#)  
XTP\_POSITION\_DIRECTION\_COVERED, [73](#)  
XTP\_POSITION\_DIRECTION\_LONG, [73](#)  
XTP\_POSITION\_DIRECTION\_NET, [73](#)  
XTP\_POSITION\_DIRECTION\_SHORT, [73](#)  
XTP\_POSITION\_DIRECTION\_TYPE, [73](#)  
XTP\_PRICE\_ALL\_OR\_CANCEL, [73](#)  
XTP\_PRICE\_BEST5\_OR\_CANCEL, [73](#)  
XTP\_PRICE\_BEST5\_OR\_LIMIT, [73](#)  
XTP\_PRICE\_BEST\_OR\_CANCEL, [73](#)  
XTP\_PRICE\_FORWARD\_BEST, [73](#)  
XTP\_PRICE\_LIMIT, [73](#)  
XTP\_PRICE\_LIMIT\_OR\_CANCEL, [73](#)  
XTP\_PRICE\_REVERSE\_BEST\_LIMIT, [73](#)  
XTP\_PRICE\_TYPE, [73](#)  
XTP\_PRICE\_TYPE\_UNKNOWN, [73](#)  
XTP\_PROTOCOL\_TCP, [73](#)  
XTP\_PROTOCOL\_TYPE, [73](#)  
XTP\_PROTOCOL\_UDP, [73](#)  
XTP\_SPLIT\_MERGE\_STATUS, [73](#)  
XTP\_SPLIT\_MERGE\_STATUS\_ALLOW, [73](#)  
XTP\_SPLIT\_MERGE\_STATUS\_FORBIDDEN, [73](#)  
XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_MERGE,  
[73](#)  
XTP\_SPLIT\_MERGE\_STATUS\_ONLY\_SPLIT, [73](#)  
XTP\_TBT\_ENTRUST, [74](#)  
XTP\_TBT\_TRADE, [74](#)  
XTP\_TBT\_TYPE, [73](#)  
XTP\_TE\_RESUME\_TYPE, [74](#)  
XTP\_TERT\_QUICK, [74](#)  
XTP\_TERT\_RESTART, [74](#)  
XTP\_TERT\_RESUME, [74](#)  
XTP\_TICKER\_TYPE, [74](#)  
XTP\_TICKER\_TYPE\_BOND, [74](#)  
XTP\_TICKER\_TYPE\_FUND, [74](#)  
XTP\_TICKER\_TYPE\_INDEX, [74](#)  
XTP\_TICKER\_TYPE\_OPTION, [74](#)  
XTP\_TICKER\_TYPE\_STOCK, [74](#)  
XTP\_TICKER\_TYPE\_UNKNOWN, [74](#)  
xtp\_api\_struct.h, [74](#)  
xtp\_api\_struct\_common.h, [75](#)  
xtp\_trader\_api.h, [75](#)