

XTP极速交易系统TraderAPI

制作者 中泰证券股份有限公司

2017年 十一月 15日 星期三 14:45:54

Contents

1	XTP 极速行情交易系统 Trader API 1.1.16.9	1
2	继承关系索引	3
2.1	类继承关系	3
3	结构体索引	5
3.1	结构体	5
4	文件索引	7
4.1	文件列表	7
5	结构体说明	9
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)	11
5.1.2.6	OnQueryIPOInfoList(XTPQueryIPOTickerRsp *ipo_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	12
5.1.2.7	OnQueryIPOQuotaInfo(XTPQueryIPOQuotaRsp *quota_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	12
5.1.2.8	OnQueryOrder(XTPQueryOrderRsp *order_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	13
5.1.2.9	OnQueryPosition(XTPQueryStkPositionRsp *position, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	13
5.1.2.10	OnQueryStructuredFund(XTPStructuredFundInfo *fund_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	13

5.1.2.11	OnQueryTrade(XTPQueryTradeRsp *trade_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	14
5.2	OrderBookStruct结构体 参考	14
5.2.1	详细描述	15
5.3	TraderApi类 参考	15
5.3.1	详细描述	16
5.3.2	成员函数说明	16
5.3.2.1	CancelOrder(const uint64_t order_xtp_id, uint64_t session_id)=0	16
5.3.2.2	CreateTraderApi(uint8_t client_id, const char *save_file_path, XTP_LOG_LEVEL log_level=XTP_LOG_LEVEL_DEBUG)	17
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	18
5.3.2.5	GetApiLastError()=0	18
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	19
5.3.2.9	InsertOrder(XTPOrderInsertInfo *order, uint64_t session_id)=0	19
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	20
5.3.2.12	QueryAsset(uint64_t session_id, int request_id)=0	20
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	21
5.3.2.15	QueryFundTransfer(XTPQueryFundTransferLogReq *query_param, uint64_t session_id, int request_id)=0	21
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	QueryOrderByXTPID(const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0	22
5.3.2.19	QueryOrders(const XTPQueryOrderReq *query_param, uint64_t session_id, int request_id)=0	22
5.3.2.20	QueryPosition(const char *ticker, uint64_t session_id, int request_id)=0	23
5.3.2.21	QueryStructuredFund(XTPQueryStructuredFundInfoReq *query_param, uint64_t session_id, int request_id)=0	23
5.3.2.22	QueryTrades(XTPQueryTraderReq *query_param, uint64_t session_id, int request_id)=0	24
5.3.2.23	QueryTradesByXTPID(const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0	24
5.3.2.24	RegisterSpi(TraderSpi *spi)=0	24
5.3.2.25	Release()=0	24
5.3.2.26	SetHeartBeatInterval(uint32_t interval)=0	25

5.3.2.27	SetSoftwareKey(const char *key)=0	26
5.3.2.28	SetSoftwareVersion(const char *version)=0	26
5.3.2.29	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(XTPQueryAssetRsp *asset, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	29
5.4.2.7	OnQueryETF(XTPQueryETFBaseRsp *etf_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	30
5.4.2.8	OnQueryETFBasket(XTPQueryETFComponentRsp *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(XTPQueryIPOTickerRsp *ipo_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	31
5.4.2.11	OnQueryIPOQuotaInfo(XTPQueryIPOQuotaRsp *quota_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	31
5.4.2.12	OnQueryOrder(XTPQueryOrderRsp *order_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	32
5.4.2.13	OnQueryPosition(XTPQueryStkPositionRsp *position, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	32
5.4.2.14	OnQueryStructuredFund(XTPStructuredFundInfo *fund_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	32
5.4.2.15	OnQueryTrade(XTPQueryTradeRsp *trade_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)	33
5.4.2.16	OnTradeEvent(XTPTradeReport *trade_info, uint64_t session_id)	33
5.5	XTPFundTransferNotice结构体 参考	34
5.5.1	详细描述	34
5.6	XTPFundTransferReq结构体 参考	34
5.6.1	详细描述	35
5.7	XTPMarketDataStruct结构体 参考	35
5.7.1	详细描述	37
5.8	XTPOrderCancelInfo结构体 参考	38
5.8.1	详细描述	38
5.9	XTPOrderInfo结构体 参考	38

5.9.1 详细描述	39
5.10 XTPOrderInsertInfo结构体 参考	39
5.10.1 详细描述	40
5.11 XTPQueryAssetRsp结构体 参考	40
5.11.1 详细描述	40
5.12 XTPQueryETFBaseReq结构体 参考	41
5.12.1 详细描述	41
5.13 XTPQueryETFBaseRsp结构体 参考	41
5.13.1 详细描述	42
5.14 XTPQueryETFComponentReq结构体 参考	42
5.14.1 详细描述	42
5.15 XTPQueryETFComponentRsp结构体 参考	42
5.15.1 详细描述	43
5.16 XTPQueryFundTransferLogReq结构体 参考	43
5.16.1 详细描述	43
5.17 XTPQueryIPOQuotaRsp结构体 参考	43
5.17.1 详细描述	43
5.18 XTPQueryIPOTickerRsp结构体 参考	44
5.18.1 详细描述	44
5.19 XTPQueryOrderReq结构体 参考	44
5.19.1 详细描述	44
5.20 XTPQueryReportByExecIdReq结构体 参考	45
5.20.1 详细描述	45
5.21 XTPQueryStkPositionRsp结构体 参考	45
5.21.1 详细描述	46
5.22 XTPQueryStructuredFundInfoReq结构体 参考	46
5.22.1 详细描述	46
5.23 XTPQueryTraderReq结构体 参考	46
5.23.1 详细描述	46
5.24 XTPQuoteStaticInfo结构体 参考	47
5.24.1 详细描述	47
5.25 XTPRspInfoStruct结构体 参考	47
5.25.1 详细描述	48
5.26 XTPSpecificTickerStruct结构体 参考	48
5.26.1 详细描述	48
5.27 XTPStructuredFundInfo结构体 参考	48
5.27.1 详细描述	49
5.28 XTPTickByTickEntrust结构体 参考	49
5.28.1 详细描述	49
5.29 XTPTickByTickStruct结构体 参考	49

5.29.1 详细描述	50
5.30 XTPTickByTickTrade结构体 参考	50
5.30.1 详细描述	51
5.30.2 结构体成员变量说明	51
5.30.2.1 trade_flag	51
5.31 XTPTickerPricelInfo结构体 参考	51
5.31.1 详细描述	51
5.32 XTPTradeReport结构体 参考	51
5.32.1 详细描述	52
6 文件说明	53
6.1 demo_test_trade_api.cpp 文件参考	53
6.1.1 详细描述	53
6.1.2 函数说明	53
6.1.2.1 main()	53
6.2 demo_test_trade_spi.h 文件参考	55
6.2.1 详细描述	55
6.3 xoms_api_fund_struct.h 文件参考	55
6.3.1 详细描述	56
6.4 xoms_api_struct.h 文件参考	56
6.4.1 详细描述	57
6.5 xquote_api_struct.h 文件参考	57
6.5.1 详细描述	58
6.6 xtp_api_data_type.h 文件参考	58
6.6.1 详细描述	61
6.6.2 枚举类型说明	61
6.6.2.1 ETF_REPLACE_TYPE	61
6.6.2.2 XTP_ACCOUNT_TYPE	62
6.6.2.3 XTP_BUSINESS_TYPE	62
6.6.2.4 XTP_EXCHANGE_TYPE	62
6.6.2.5 XTP_FUND_OPER_STATUS	63
6.6.2.6 XTP_FUND_TRANSFER_TYPE	63
6.6.2.7 XTP_LOG_LEVEL	63
6.6.2.8 XTP_MARKET_TYPE	63
6.6.2.9 XTP_ORDER_ACTION_STATUS_TYPE	64
6.6.2.10 XTP_ORDER_STATUS_TYPE	64
6.6.2.11 XTP_ORDER_SUBMIT_STATUS_TYPE	64
6.6.2.12 XTP_PRICE_TYPE	64
6.6.2.13 XTP_PROTOCOL_TYPE	65
6.6.2.14 XTP_SIDE_TYPE	65

6.6.2.15	XTP_SPLIT_MERGE_STATUS	65
6.6.2.16	XTP_TBT_TYPE	65
6.6.2.17	XTP_TE_RESUME_TYPE	66
6.6.2.18	XTP_TICKER_TYPE	66
6.7	xtp_api_struct.h 文件参考	66
6.7.1	详细描述	66
6.8	xtp_api_struct_common.h 文件参考	66
6.8.1	详细描述	67
6.9	xtp_trader_api.h 文件参考	67
6.9.1	详细描述	67
索引		69

Chapter 1

XTP 极速行情交易系统 Trader API 1.1.16.9

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	14
TraderApi	15
TraderSpi	27
DemoTestTraderSpi	9
XTPFundTransferNotice	34
XTPFundTransferReq	34
XTPMarketDataStruct	35
XTPOrderCancelInfo	38
XTPOrderInfo	38
XTPOrderInsertInfo	39
XTPQueryAssetRsp	40
XTPQueryETFBaseReq	41
XTPQueryETFBaseRsp	41
XTPQueryETFComponentReq	42
XTPQueryETFComponentRsp	42
XTPQueryFundTransferLogReq	43
XTPQueryIPOQuotaRsp	43
XTPQueryIPOTickerRsp	44
XTPQueryOrderReq	44
XTPQueryReportByExecIdReq	45
XTPQueryStkPositionRsp	45
XTPQueryStructuredFundInfoReq	46
XTPQueryTraderReq	46
XTPQuoteStaticInfo	47
XTPRspInfoStruct	47
XTPSpecificTickerStruct	48
XTPStructuredFundInfo	48
XTPTickByTickEntrust	49
XTPTickByTickStruct	49
XTPTickByTickTrade	50
XPTTickerPriceInfo	51
XTPTradeReport	51

结构体索引

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	14
TraderApi		
交易接口类	15
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	34
XTPMarketDataStruct		
行情	35
XTPOrderCancelInfo		
撤单失败响应消息	38
XTPOrderInfo		
报单响应结构体	38
XTPOrderInsertInfo		
新订单请求	39
XTPQueryAssetRsp		
账户资金查询响应结构体	40
XTPQueryETFBaseReq		
查询股票ETF合约基本情况-请求结构体	41
XTPQueryETFBaseRsp		
查询股票ETF合约基本情况-响应结构体	41
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	42
XTPQueryETFComponentRsp		
查询股票ETF合约成分股信息-响应结构体	42
XTPQueryFundTransferLogReq		
资金内转流水查询请求与响应	43
XTPQueryIPOQuotaRsp		
查询用户申购额度	43
XTPQueryIPTickerRsp		
查询当日可申购新股信息	44
XTPQueryOrderReq		
报单查询 //报单查询请求-条件查询	44

XTPQueryReportByExecIdReq	
成交回报查询 // 查询成交报告请求-根据执行编号查询（保留字段）	45
XTPQueryStkPositionRsp	
查询股票持仓情况	45
XTPQueryStructuredFundInfoReq	
查询分级基金信息结构体	46
XTPQueryTraderReq	
查询成交回报请求-查询条件	46
XTPQuoteStaticInfo	
股票行情静态信息	47
XTPRspInfoStruct	
响应信息	47
XTPSpecificTickerStruct	
指定的合约	48
XTPStructuredFundInfo	
查询分级基金信息响应结构体	48
XTPTickByTickEntrust	
逐笔委托(仅适用深交所)	49
XTPTickByTickStruct	
逐笔数据信息	49
XTPTickByTickTrade	
逐笔成交	50
XTPTickerPriceInfo	
供查询的最新信息	51
XTPTradeReport	
报单成交结构体	51

Chapter 4

文件索引

4.1 文件列表

这里列出了所有文档化的文件，并附带简要说明：

demo_test_trade_api.cpp	
定义控制台测试应用程序的入口点	53
demo_test_trade_spi.h	
Demo自定义客户端交易响应接口类	55
xoms_api_fund_struct.h	
定义资金划拨相关结构体类型	55
xoms_api_struct.h	
定义交易类相关数据结构	56
xquote_api_struct.h	
定义行情类相关数据结构	57
xtp_api_data_type.h	
定义兼容数据基本类型	58
xtp_api_struct.h	
定义业务数据结构	66
xtp_api_struct_common.h	
定义业务公共数据结构	66
xtp_trader_api.h	
定义客户端交易接口	67

Chapter 5

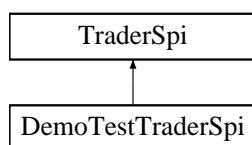
结构体说明

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)

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error_info为空，或者error_info.error_↔ id为0时，表明没有错误
<i>request_id</i>	此消息响应函数对应的请求ID
<i>is_last</i>	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的 时候为true，如果为false，表示还有其他后续消息响应
<i>session_id</i>	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

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

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

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

参数

<i>quota_info</i>	查询到的用户某个市场的今日新股申购额度信息
<i>error_info</i>	查查询用户新股申购额度信息发生错误时返回的错误信息，当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](#) .

5.1.2.8 `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](#) .

5.1.2.9 `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](#) .

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

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

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 [TraderSpi](#) .

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

请求查询成交响应

参数

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

备注

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

重载 [TraderSpi](#) .

该类的文档由以下文件生成:

- [demo_test_trade_spi.h](#)

5.2 OrderBookStruct结构体 参考

定单簿

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

成员变量

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

- 最新价
- `int64_t qty`
数量, 为总成交量
- `double turnover`
成交金额, 为总成交金额
- `int64_t trades_count`
成交笔数
- `double bid [10]`
十档申买价
- `double ask [10]`
十档申卖价
- `int64_t bid_qty [10]`
十档申买量
- `int64_t ask_qty [10]`
十档申卖量
- `int64_t data_time`
时间类

5.2.1 详细描述

定单簿

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.3 TraderApi类 参考

交易接口类

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

Public 成员函数

- `virtual void Release ()=0`
- `virtual const char * GetTradingDay ()=0`
- `virtual void RegisterSpi (TraderSpi *spi)=0`
- `virtual XTPRI * GetApiLastError ()=0`
- `virtual const char * GetApiVersion ()=0`
- `virtual uint8_t GetClientIDByXTPID (uint64_t order_xtp_id)=0`
- `virtual const char * GetAccountByXTPID (uint64_t order_xtp_id)=0`
- `virtual void SubscribePublicTopic (XTP_TE_RESUME_TYPE resume_type)=0`
- `virtual void SetSoftwareVersion (const char *version)=0`
- `virtual void SetSoftwareKey (const char *key)=0`
- `virtual void SetHeartBeatInterval (uint32_t interval)=0`
- `virtual uint64_t Login (const char *ip, int port, const char *user, const char *password, XTP_PROTOCOL↵
_TYPE sock_type)=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](#) (XTPQueryETFBaseReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryETFTickerBasket](#) (XTPQueryETFComponentReq *query_param, uint64_t session_id, int request_id)=0
- virtual int [QueryIPOInfoList](#) (uint64_t session_id, int request_id)=0
- virtual int [QueryIPOQuotalInfo](#) (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>fund_transfer</i>	资金划拨请求具体信息
<i>session_id</i>	资金账户对应的 <i>session_id</i> ,登录时得到

5.3.2.4 `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 <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

5.3.2.13 `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, <i>market</i> 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, <i>market</i> 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 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

5.3.2.19 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YYYYMMDDHHMMSSsss，为0则默认当前时间
<i>session_id</i>	资金账户对应的 <i>session_id</i> ，登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

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

5.3.2.20 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

5.3.2.21 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

5.3.2.22 `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session_id,登录时得到
<i>request_id</i>	用于用户定位查询响应的ID，由用户自定义

备注

该方法支持分时段查询，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

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

根据委托编号请求查询相关成交

返回

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

参数

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

备注

此函数查询出的结果可能对应多个查询结果响应

5.3.2.24 `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
------------	----------------------

5.3.2.25 `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

5.3.2.26 `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

5.3.2.27 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

5.3.2.28 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

5.3.2.29 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:

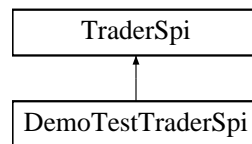
- [xtp_trader_api.h](#)

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)

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 (XTPQueryAssetResp * 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 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.

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

请求查询投资者持仓响应

参数

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

备注

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

被 [DemoTestTraderSpi](#) 重载.

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

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

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestTraderSpi](#) 重载.

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

请求查询成交响应

参数

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

备注

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

被 [DemoTestTraderSpi](#) 重载.

5.4.2.16 `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 <i>trade_info.order_xtp_id</i> 来管理订单，通过 <i>GetClientIDByXTPID() == client_id</i> 来过滤自己的订单。对于上交所， <i>exec_id</i> 可以唯一标识一笔成交。当发现2笔成交回报拥有相同的 <i>exec_id</i> ，则可以认为此笔交易自成交了。对于深交所， <i>exec_id</i> 是唯一的，暂时无此判断机制。 <i>report_index+market</i> 字段可以组成唯一标识表示成交回报。
-------------------	--

<code>session_id</code>	资金账户对应的 <code>session_id</code> ，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 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
- double [pre_open_interest](#)
昨持仓量（目前未填写）
- double [open_interest](#)
持仓量（目前未填写）
- double [pre_settlement_price](#)
上次结算价（目前未填写）
- double [settlement_price](#)
本次结算价（目前未填写）
- double [upper_limit_price](#)
涨停板价（目前未填写）
- double [lower_limit_price](#)

- 跌停板价（目前未填写）
- double `pre_delta`
 - 昨虚实度（目前未填写）
- double `curr_delta`
 - 今虚实度（目前未填写）
- 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
- int64_t `total_bid_qty`
 - 委托买入总量
- int64_t `total_ask_qty`
 - 委托卖出总量
- double `ma_bid_price`
 - 加权平均委买价格
- double `ma_ask_price`
 - 加权平均委卖价格
- double `ma_bond_bid_price`
 - 债券加权平均委买价格
- double `ma_bond_ask_price`
 - 债券加权平均委卖价格
- double `yield_to_maturity`
 - 债券到期收益率
- double `iopv`
 - ETF净值估值
- int32_t `etf_buy_count`
 - ETF申购笔数
- int32_t `etf_sell_count`
 - ETF赎回笔数
- double `etf_buy_qty`
 - ETF申购数量
- double `etf_buy_money`
 - ETF申购金额
- double `etf_sell_qty`
 - ETF赎回数量

- double [etf_sell_money](#)
ETF赎回金额
- double [total_warrant_exec_qty](#)
权证执行的总数量
- double [warrant_lower_price](#)
权证跌停价格 (元)
- double [warrant_upper_price](#)
权证涨停价格 (元)
- int32_t [cancel_buy_count](#)
买入撤单笔数
- int32_t [cancel_sell_count](#)
卖出撤单笔数
- double [cancel_buy_qty](#)
买入撤单数量
- double [cancel_sell_qty](#)
卖出撤单数量
- double [cancel_buy_money](#)
买入撤单金额
- double [cancel_sell_money](#)
卖出撤单金额
- int64_t [total_buy_count](#)
买入总笔数
- int64_t [total_sell_count](#)
卖出总笔数
- int32_t [duration_after_buy](#)
买入委托成交最大等待时间
- int32_t [duration_after_sell](#)
卖出委托成交最大等待时间
- int32_t [num_bid_orders](#)
买方委托价位数
- int32_t [num_ask_orders](#)
卖方委托价位数
- int32_t [exec_time](#)
成交时间 (UA3113)
- char [is_market_closed](#) [4]
闭市标志 (UA103/UA104)
- double [total_position](#)
合约持仓量 (UA103)
- double [pe_ratio1](#)
市盈率1 (UA103)
- double [pe_ratio2](#)
市盈率2 (UA103)

5.7.1 详细描述

行情

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.8 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

5.8.1 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](#)

5.9 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
- `XTP_SIDE_TYPE side`
买卖方向
- `XTP_BUSINESS_TYPE business_type`
业务类型

- `int64_t qty_traded`
今成交数量，为此订单累计成交数量
- `int64_t qty_left`
剩余数量，当撤单成功时，表示撤单数量
- `int64_t insert_time`
委托时间，格式为YYYYMMDDHHMMSSsss
- `int64_t update_time`
最后修改时间，格式为YYYYMMDDHHMMSSsss
- `int64_t cancel_time`
撤销时间，格式为YYYYMMDDHHMMSSsss
- `double trade_amount`
成交金额，为此订单的成交总金额
- `char order_local_id [XTP_LOCAL_ORDER_LEN]`
本地报单编号 OMS生成的单号，不等同于`order_xtp_id`，为服务器传到报盘的单号
- `XTP_ORDER_STATUS_TYPE order_status`
报单状态，订单响应中没有部分成交状态的推送，在查询订单结果中，会有部分成交状态
- `XTP_ORDER_SUBMIT_STATUS_TYPE order_submit_status`
报单提交状态，OMS内部使用，用户无需关心
- `XTPOrderTypeType order_type`
报单类型

5.9.1 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`

5.10 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
- [XTP_SIDE_TYPE side](#)
买卖方向
- [XTP_BUSINESS_TYPE business_type](#)
业务类型

5.10.1 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](#)

5.11 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
- [uint64_t unknown \[43\]](#)
(保留字段)

5.11.1 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](#)

5.12 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

5.12.1 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](#)

5.13 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= $net_value * unit$

5.13.1 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](#)

5.14 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

5.14.1 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](#)

5.15 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

5.15.1 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](#)

5.16 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

5.16.1 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](#)

5.17 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

5.17.1 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](#)

5.18 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

5.18.1 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](#)

5.19 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

5.19.1 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](#)

5.20 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

5.20.1 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](#)

5.21 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）
- uint64_t [unknown](#) [50]
(保留字段)

5.21.1 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](#)

5.22 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

5.22.1 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](#)

5.23 XTPQueryTraderReq结构体 参考

查询成交回报请求-查询条件

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

成员变量

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

5.23.1 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](#)

5.24 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)

5.24.1 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](#)

5.25 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

5.25.1 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](#)

5.26 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

5.26.1 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](#)

5.27 XTPStructuredFundInfo结构体 参考

查询分级基金信息响应结构体

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

成员变量

- [XTP_EXCHANGE_TYPE exchange_id](#)
交易所代码
- [char sf_ticker \[XTP_TICKER_LEN\]](#)
分级基金母基金代码
- [char sf_ticker_name \[XTP_TICKER_NAME_LEN\]](#)
分级基金母基金名称
- [char ticker \[XTP_TICKER_LEN\]](#)
分级基金子基金代码
- [char ticker_name \[XTP_TICKER_NAME_LEN\]](#)
分级基金子基金名称
- [XTP_SPLIT_MERGE_STATUS split_merge_status](#)
基金允许拆分合并状态
- [uint32_t ratio](#)
拆分合并比例

- uint32_t [min_split_qty](#)
最小拆分数
- uint32_t [min_merge_qty](#)
最小合并数量
- double [net_price](#)
基金净值

5.27.1 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](#)

5.28 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; '3': 本方最优

5.28.1 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](#)

5.29 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`
};

5.29.1 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](#)

5.30 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](#)

5.30.1 详细描述

逐笔成交

5.30.2 结构体成员变量说明

5.30.2.1 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](#)

5.31 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

5.31.1 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](#)

5.32 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
- [XTPTTradeType trade_type](#)
成交类型 - 成交回报中的执行类型
- [XTP_SIDE_TYPE side](#)
买卖方向
- [XTP_BUSINESS_TYPE business_type](#)
业务类型
- [char branch_pbu \[XTP_BRANCH_PBU_LEN\]](#)
交易所交易员代码

5.32.1 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;
    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;
    //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
用户资金账户密码字符串数组长度

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"
```

结构体

- struct [XTPOrderInsertInfo](#)
新订单请求
- struct [XTPOrderCancelInfo](#)
撤单失败响应消息
- struct [XTPOrderInfo](#)
报单响应结构体
- struct [XTPTradeReport](#)
报单成交结构体
- struct [XTPQueryOrderReq](#)
报单查询 // 报单查询请求-条件查询
- struct [XTPQueryReportByExecIdReq](#)
成交回报查询 // 查询成交报告请求-根据执行编号查询（保留字段）
- struct [XTPQueryTraderReq](#)
查询成交回报请求-查询条件
- struct [XTPQueryAssetRsp](#)
账户资金查询响应结构体
- struct [XTPQueryStkPositionRsp](#)
查询股票持仓情况
- struct [XTPFundTransferNotice](#)
资金内转流水通知
- struct [XTPQueryFundTransferLogReq](#)
资金内转流水查询请求与响应
- struct [XTPQueryStructuredFundInfoReq](#)
查询分级基金信息结构体
- struct [XTPStructuredFundInfo](#)
查询分级基金信息响应结构体
- struct [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

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 [XTPMarketDataStruct](#)
行情
- struct [XTPQuoteStaticInfo](#)
股票行情静态信息
- struct [OrderBookStruct](#)
订单簿
- struct [XTPTickByTickEntrust](#)
逐笔委托(仅适用深交所)
- struct [XTPTickByTickTrade](#)
逐笔成交
- struct [XTPTickByTickStruct](#)
逐笔数据信息
- struct [XTPTickerPriceInfo](#)
供查询的最新信息

类型定义

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

6.5.1 详细描述

定义行情类相关数据结构

作者

中泰证券股份有限公司

6.6 xtp_api_data_type.h 文件参考

定义兼容数据基本类型

宏定义

- #define [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_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 enum XTP_SIDE_TYPE XTP_SIDE_TYPE`
*XTP_SIDE_TYPE*是买卖方向类型
- `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 char `TXTPTradeTypeType`
`TXTPTradeTypeType`是成交类型类型
- typedef char `TXTPOrderTypeType`
`TXTPOrderTypeType`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_TYPE_UNKNOWN` }
`XTP_PRICE_TYPE`是价格类型
- enum `XTP_SIDE_TYPE` {
`XTP_SIDE_BUY` = 1, `XTP_SIDE_SELL`, `XTP_SIDE_BUY_OPEN`, `XTP_SIDE_SELL_OPEN`,
`XTP_SIDE_BUY_CLOSE`, `XTP_SIDE_SELL_CLOSE`, `XTP_SIDE_PURCHASE`, `XTP_SIDE_REDEMPTION`,
`XTP_SIDE_SPLIT`, `XTP_SIDE_MERGE`, `XTP_SIDE_UNKNOWN` }
`XTP_SIDE_TYPE`是买卖方向类型
- enum `XTP_ORDER_ACTION_STATUS_TYPE` { `XTP_ORDER_ACTION_STATUS_SUBMITTED` = 1, `XTP_ORDER_ACTION_STATUS_ACCEPTED`, `XTP_ORDER_ACTION_STATUS_REJECTED` }
`XTP_ORDER_ACTION_STATUS_TYPE`是报单操作状态类型
- enum `XTP_ORDER_STATUS_TYPE` {
`XTP_ORDER_STATUS_INIT` = 0, `XTP_ORDER_STATUS_ALLTRADED` = 1, `XTP_ORDER_STATUS_PARTTRADEDQUEUEING`,
`XTP_ORDER_STATUS_PARTTRADEDNOTQUEUEING`,
`XTP_ORDER_STATUS_NOTTRADEQUEUEING`, `XTP_ORDER_STATUS_CANCELED`, `XTP_ORDER_STATUS_REJECTED`, `XTP_ORDER_STATUS_UNKNOWN` }
`XTP_ORDER_STATUS_TYPE`是报单状态类型
- enum `XTP_ORDER_SUBMIT_STATUS_TYPE` {
`XTP_ORDER_SUBMIT_STATUS_INSERT_SUBMITTED` = 1, `XTP_ORDER_SUBMIT_STATUS_INSERT_ACCEPTED`, `XTP_ORDER_SUBMIT_STATUS_INSERT_REJECTED`, `XTP_ORDER_SUBMIT_STATUS_CANCEL_SUBMITTED`,
`XTP_ORDER_SUBMIT_STATUS_CANCEL_REJECTED`, `XTP_ORDER_SUBMIT_STATUS_CANCEL_ACCEPTED` }
`XTP_ORDER_SUBMIT_STATUS_TYPE`是报单提交状态类型

- XTP_ORDER_SUBMIT_STATUS_TYPE*是报单提交状态类型
- enum *XTP_TE_RESUME_TYPE* { *XTP_TERT_RESTART* = 0, *XTP_TERT_RESUME*, *XTP_TERT_QUICK* }
*XTP_TE_RESUME_TYPE*是公有流（订单响应、成交回报）重传方式
- enum *ETF_REPLACE_TYPE* { *ERT_CASH_FORBIDDEN* = 0, *ERT_CASH_OPTIONAL*, *ERT_CASH_MUST*, *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_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_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

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

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（暂未支持）

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_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_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

6.6.2.10 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

6.6.2.11 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

6.6.2.12 enum XTP_PRICE_TYPE

XTP_PRICE_TYPE是价格类型

枚举值

XTP_PRICE_LIMIT 限价单-沪深（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
XTP_PRICE_BEST5_OR_LIMIT 最优五档即时成交剩余转限价，市价单-沪
XTP_PRICE_BEST5_OR_CANCEL 最优5档即时成交剩余转撤销，市价单-沪深
XTP_PRICE_ALL_OR_CANCEL 全部成交或撤销，市价单-深
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
XTP_PRICE_TYPE_UNKNOWN 未知或者无效价格类型

6.6.2.13 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（目前暂未支持）

6.6.2.14 enum XTP_SIDE_TYPE

XTP_SIDE_TYPE是买卖方向类型

枚举值

XTP_SIDE_BUY 买（新股申购、ETF买等）

XTP_SIDE_SELL 卖（逆回购）

XTP_SIDE_BUY_OPEN 买开（暂未支持）

XTP_SIDE_SELL_OPEN 卖开（暂未支持）

XTP_SIDE_BUY_CLOSE 买平（暂未支持）

XTP_SIDE_SELL_CLOSE 卖平（暂未支持）

XTP_SIDE_PURCHASE 申购

XTP_SIDE_REDEMPTION 赎回

XTP_SIDE_SPLIT 拆分

XTP_SIDE_MERGE 合并

XTP_SIDE_UNKNOWN 未知或者无效买卖方向

6.6.2.15 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

6.6.2.16 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

6.6.2.17 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

6.6.2.18 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_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 [XTPRsplInfoStruct](#)
响应信息

宏定义

- `#define XTP_ERR_MSG_LEN 124`
错误信息的字符串长度

类型定义

- `typedef struct 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, [16](#)

CreateTraderApi
 XTP::API::TraderApi, [16](#)

demo_test_trade_api.cpp, [53](#)
 main, [53](#)

demo_test_trade_spi.h, [55](#)

DemoTestTraderSpi, [9](#)
 OnFundTransfer, [10](#)
 OnQueryAsset, [10](#)
 OnQueryETF, [11](#)
 OnQueryETFBasket, [11](#)
 OnQueryFundTransfer, [11](#)
 OnQueryIPOInfoList, [12](#)
 OnQueryIPOQuotaInfo, [12](#)
 OnQueryOrder, [13](#)
 OnQueryPosition, [13](#)
 OnQueryStructuredFund, [13](#)
 OnQueryTrade, [14](#)

EPT_INVALID
 xtp_api_data_type.h, [62](#)

ERT_CASH_FORBIDDEN
 xtp_api_data_type.h, [61](#)

ERT_CASH_MUST
 xtp_api_data_type.h, [62](#)

ERT_CASH_OPTIONAL
 xtp_api_data_type.h, [61](#)

ETF_REPLACE_TYPE
 xtp_api_data_type.h, [61](#)

FundTransfer
 XTP::API::TraderApi, [18](#)

GetAccountByXTPID
 XTP::API::TraderApi, [18](#)

GetApiLastError
 XTP::API::TraderApi, [18](#)

GetApiVersion
 XTP::API::TraderApi, [19](#)

GetClientIDByXTPID
 XTP::API::TraderApi, [19](#)

GetTradingDay
 XTP::API::TraderApi, [19](#)

InsertOrder
 XTP::API::TraderApi, [19](#)

Login
 XTP::API::TraderApi, [20](#)

Logout
 XTP::API::TraderApi, [20](#)

main
 demo_test_trade_api.cpp, [53](#)

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, [12](#)
 XTP::API::TraderSpi, [31](#)

OnQueryIPOQuotaInfo
 DemoTestTraderSpi, [12](#)
 XTP::API::TraderSpi, [31](#)

OnQueryOrder
 DemoTestTraderSpi, [13](#)
 XTP::API::TraderSpi, [32](#)

OnQueryPosition
 DemoTestTraderSpi, [13](#)
 XTP::API::TraderSpi, [32](#)

OnQueryStructuredFund
 DemoTestTraderSpi, [13](#)
 XTP::API::TraderSpi, [32](#)

OnQueryTrade
 DemoTestTraderSpi, [14](#)
 XTP::API::TraderSpi, [33](#)

OnTradeEvent
 XTP::API::TraderSpi, [33](#)

OrderBookStruct, [14](#)

- QueryAsset
 - XTP::API::TraderApi, 20
- QueryETF
 - XTP::API::TraderApi, 21
- QueryETFTickerBasket
 - XTP::API::TraderApi, 21
- QueryFundTransfer
 - XTP::API::TraderApi, 21
- QueryIPOInfoList
 - XTP::API::TraderApi, 22
- QueryIPOQuotaInfo
 - XTP::API::TraderApi, 22
- QueryOrderByXTPID
 - XTP::API::TraderApi, 22
- QueryOrders
 - XTP::API::TraderApi, 22
- QueryPosition
 - XTP::API::TraderApi, 23
- QueryStructuredFund
 - XTP::API::TraderApi, 23
- QueryTrades
 - XTP::API::TraderApi, 23
- QueryTradesByXTPID
 - XTP::API::TraderApi, 24
- RegisterSpi
 - XTP::API::TraderApi, 24
- Release
 - XTP::API::TraderApi, 24
- SetHeartBeatInterval
 - XTP::API::TraderApi, 24
- SetSoftwareKey
 - XTP::API::TraderApi, 26
- SetSoftwareVersion
 - XTP::API::TraderApi, 26
- SubscribePublicTopic
 - XTP::API::TraderApi, 26
- trade_flag
 - XTPTickByTickTrade, 51
- TraderApi, 15
- TraderSpi, 27
- XTP::API::TraderApi
 - CancelOrder, 16
 - CreateTraderApi, 16
 - FundTransfer, 18
 - GetAccountByXTPID, 18
 - GetApiLastError, 18
 - GetApiVersion, 19
 - GetClientIDByXTPID, 19
 - GetTradingDay, 19
 - InsertOrder, 19
 - Login, 20
 - Logout, 20
 - QueryAsset, 20
 - QueryETF, 21
 - QueryETFTickerBasket, 21
 - QueryFundTransfer, 21
 - QueryIPOInfoList, 22
 - QueryIPOQuotaInfo, 22
 - QueryOrderByXTPID, 22
 - QueryOrders, 22
 - QueryPosition, 23
 - QueryStructuredFund, 23
 - QueryTrades, 23
 - QueryTradesByXTPID, 24
 - RegisterSpi, 24
 - Release, 24
 - SetHeartBeatInterval, 24
 - 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
 - OnQueryOrder, 32
 - OnQueryPosition, 32
 - OnQueryStructuredFund, 32
 - OnQueryTrade, 33
 - OnTradeEvent, 33
- XTP_ACCOUNT_CREDIT
 - xtp_api_data_type.h, 62
- XTP_ACCOUNT_DERIVE
 - xtp_api_data_type.h, 62
- XTP_ACCOUNT_NORMAL
 - xtp_api_data_type.h, 62
- XTP_ACCOUNT_TYPE
 - xtp_api_data_type.h, 62
- XTP_ACCOUNT_UNKNOWN
 - xtp_api_data_type.h, 62
- XTP_BUSINESS_TYPE
 - xtp_api_data_type.h, 62
- XTP_BUSINESS_TYPE_ALLOTMENT
 - xtp_api_data_type.h, 62
- XTP_BUSINESS_TYPE_CASH
 - xtp_api_data_type.h, 62
- XTP_BUSINESS_TYPE_DESIGNATION
 - xtp_api_data_type.h, 62
- XTP_BUSINESS_TYPE_ETF
 - xtp_api_data_type.h, 62
- XTP_BUSINESS_TYPE_IPOS
 - xtp_api_data_type.h, 62
- XTP_BUSINESS_TYPE_MARGIN
 - xtp_api_data_type.h, 62
- XTP_BUSINESS_TYPE_MONEY_FUND
 - xtp_api_data_type.h, 62

XTP_BUSINESS_TYPE_REPO
 xtp_api_data_type.h, 62
 XTP_BUSINESS_TYPE_STRUCTURED_FUND_PURCHASE_REDEMPTION
 xtp_api_data_type.h, 62
 XTP_BUSINESS_TYPE_STRUCTURED_FUND_SPLIT_MERGE
 xtp_api_data_type.h, 62
 XTP_BUSINESS_TYPE_UNKNOWN
 xtp_api_data_type.h, 62
 XTP_EXCHANGE_SH
 xtp_api_data_type.h, 62
 XTP_EXCHANGE_SZ
 xtp_api_data_type.h, 62
 XTP_EXCHANGE_TYPE
 xtp_api_data_type.h, 62
 XTP_EXCHANGE_UNKNOWN
 xtp_api_data_type.h, 62
 XTP_FUND_OPER_FAILED
 xtp_api_data_type.h, 63
 XTP_FUND_OPER_PROCESSING
 xtp_api_data_type.h, 63
 XTP_FUND_OPER_STATUS
 xtp_api_data_type.h, 62
 XTP_FUND_OPER_SUBMITTED
 xtp_api_data_type.h, 63
 XTP_FUND_OPER_SUCCESS
 xtp_api_data_type.h, 63
 XTP_FUND_OPER_UNKNOWN
 xtp_api_data_type.h, 63
 XTP_FUND_TRANSFER_IN
 xtp_api_data_type.h, 63
 XTP_FUND_TRANSFER_OUT
 xtp_api_data_type.h, 63
 XTP_FUND_TRANSFER_TYPE
 xtp_api_data_type.h, 63
 XTP_FUND_TRANSFER_UNKNOWN
 xtp_api_data_type.h, 63
 XTP_LOG_LEVEL
 xtp_api_data_type.h, 63
 XTP_LOG_LEVEL_DEBUG
 xtp_api_data_type.h, 63
 XTP_LOG_LEVEL_ERROR
 xtp_api_data_type.h, 63
 XTP_LOG_LEVEL_FATAL
 xtp_api_data_type.h, 63
 XTP_LOG_LEVEL_INFO
 xtp_api_data_type.h, 63
 XTP_LOG_LEVEL_TRACE
 xtp_api_data_type.h, 63
 XTP_LOG_LEVEL_WARNING
 xtp_api_data_type.h, 63
 XTP_MARKET_TYPE
 xtp_api_data_type.h, 63
 XTP_MKT_INIT
 xtp_api_data_type.h, 63
 XTP_MKT_SH_A
 xtp_api_data_type.h, 63
 XTP_MKT_SZ_A
 xtp_api_data_type.h, 63
 XTP_MKT_UNKNOWN
 xtp_api_data_type.h, 63
 XTP_ORDER_ACTION_STATUS_ACCEPTED
 xtp_api_data_type.h, 64
 XTP_ORDER_ACTION_STATUS_REJECTED
 xtp_api_data_type.h, 64
 XTP_ORDER_ACTION_STATUS_SUBMITTED
 xtp_api_data_type.h, 64
 XTP_ORDER_ACTION_STATUS_TYPE
 xtp_api_data_type.h, 63
 XTP_ORDER_STATUS_ALLTRADED
 xtp_api_data_type.h, 64
 XTP_ORDER_STATUS_CANCELED
 xtp_api_data_type.h, 64
 XTP_ORDER_STATUS_INIT
 xtp_api_data_type.h, 64
 XTP_ORDER_STATUS_NOTTRADEQUEUEING
 xtp_api_data_type.h, 64
 XTP_ORDER_STATUS_PARTTRADEDNOTQUEUEING
 xtp_api_data_type.h, 64
 XTP_ORDER_STATUS_PARTTRADEDQUEUEING
 xtp_api_data_type.h, 64
 XTP_ORDER_STATUS_REJECTED
 xtp_api_data_type.h, 64
 XTP_ORDER_STATUS_TYPE
 xtp_api_data_type.h, 64
 XTP_ORDER_STATUS_UNKNOWN
 xtp_api_data_type.h, 64
 XTP_ORDER_SUBMIT_STATUS_CANCEL_ACCEPTED
 xtp_api_data_type.h, 64
 XTP_ORDER_SUBMIT_STATUS_CANCEL_REJECTED
 xtp_api_data_type.h, 64
 XTP_ORDER_SUBMIT_STATUS_CANCEL_SUBMITTED
 xtp_api_data_type.h, 64
 XTP_ORDER_SUBMIT_STATUS_INSERT_ACCEPTED
 xtp_api_data_type.h, 64
 XTP_ORDER_SUBMIT_STATUS_INSERT_REJECTED
 xtp_api_data_type.h, 64
 XTP_ORDER_SUBMIT_STATUS_INSERT_SUBMITTED
 xtp_api_data_type.h, 64
 XTP_ORDER_SUBMIT_STATUS_TYPE
 xtp_api_data_type.h, 64
 XTP_PRICE_ALL_OR_CANCEL
 xtp_api_data_type.h, 64
 XTP_PRICE_BEST5_OR_CANCEL
 xtp_api_data_type.h, 64
 XTP_PRICE_BEST5_OR_LIMIT
 xtp_api_data_type.h, 64
 XTP_PRICE_BEST_OR_CANCEL

xtp_api_data_type.h, 64
 XTP_PRICE_FORWARD_BEST
 xtp_api_data_type.h, 64
 XTP_PRICE_LIMIT
 xtp_api_data_type.h, 64
 XTP_PRICE_REVERSE_BEST_LIMIT
 xtp_api_data_type.h, 64
 XTP_PRICE_TYPE
 xtp_api_data_type.h, 64
 XTP_PRICE_TYPE_UNKNOWN
 xtp_api_data_type.h, 64
 XTP_PROTOCOL_TCP
 xtp_api_data_type.h, 65
 XTP_PROTOCOL_TYPE
 xtp_api_data_type.h, 64
 XTP_PROTOCOL_UDP
 xtp_api_data_type.h, 65
 XTP_SIDE_BUY
 xtp_api_data_type.h, 65
 XTP_SIDE_BUY_CLOSE
 xtp_api_data_type.h, 65
 XTP_SIDE_BUY_OPEN
 xtp_api_data_type.h, 65
 XTP_SIDE_MERGE
 xtp_api_data_type.h, 65
 XTP_SIDE_PURCHASE
 xtp_api_data_type.h, 65
 XTP_SIDE_REDEMPTION
 xtp_api_data_type.h, 65
 XTP_SIDE_SELL
 xtp_api_data_type.h, 65
 XTP_SIDE_SELL_CLOSE
 xtp_api_data_type.h, 65
 XTP_SIDE_SELL_OPEN
 xtp_api_data_type.h, 65
 XTP_SIDE_SPLIT
 xtp_api_data_type.h, 65
 XTP_SIDE_TYPE
 xtp_api_data_type.h, 65
 XTP_SIDE_UNKNOWN
 xtp_api_data_type.h, 65
 XTP_SPLIT_MERGE_STATUS
 xtp_api_data_type.h, 65
 XTP_SPLIT_MERGE_STATUS_ALLOW
 xtp_api_data_type.h, 65
 XTP_SPLIT_MERGE_STATUS_FORBIDDEN
 xtp_api_data_type.h, 65
 XTP_SPLIT_MERGE_STATUS_ONLY_MERGE
 xtp_api_data_type.h, 65
 XTP_SPLIT_MERGE_STATUS_ONLY_SPLIT
 xtp_api_data_type.h, 65
 XTP_TBT_ENTRUST
 xtp_api_data_type.h, 65
 XTP_TBT_TRADE
 xtp_api_data_type.h, 65
 XTP_TBT_TYPE
 xtp_api_data_type.h, 65
 XTP_TE_RESUME_TYPE
 xtp_api_data_type.h, 65
 XTP_TERT_QUICK
 xtp_api_data_type.h, 66
 XTP_TERT_RESTART
 xtp_api_data_type.h, 66
 XTP_TERT_RESUME
 xtp_api_data_type.h, 66
 XTP_TICKER_TYPE
 xtp_api_data_type.h, 66
 XTP_TICKER_TYPE_BOND
 xtp_api_data_type.h, 66
 XTP_TICKER_TYPE_FUND
 xtp_api_data_type.h, 66
 XTP_TICKER_TYPE_INDEX
 xtp_api_data_type.h, 66
 XTP_TICKER_TYPE_STOCK
 xtp_api_data_type.h, 66
 XTP_TICKER_TYPE_UNKNOWN
 xtp_api_data_type.h, 66
 XTPFundTransferNotice, 34
 XTPFundTransferReq, 34
 XTPMarketDataStruct, 35
 XTPOrderCancelInfo, 38
 XTPOrderInfo, 38
 XTPOrderInsertInfo, 39
 XTPQueryAssetRsp, 40
 XTPQueryETFBaseReq, 41
 XTPQueryETFBaseRsp, 41
 XTPQueryETFComponentReq, 42
 XTPQueryETFComponentRsp, 42
 XTPQueryFundTransferLogReq, 43
 XTPQueryIPOQuotaRsp, 43
 XTPQueryIPOTickerRsp, 44
 XTPQueryOrderReq, 44
 XTPQueryReportByExecIdReq, 45
 XTPQueryStkPositionRsp, 45
 XTPQueryStructuredFundInfoReq, 46
 XTPQueryTraderReq, 46
 XTPQuoteStaticInfo, 47
 XTPRspInfoStruct, 47
 XTPSpecificTickerStruct, 48
 XTPStructuredFundInfo, 48
 XTPTickByTickEntrust, 49
 XTPTickByTickStruct, 49
 XTPTickByTickTrade, 50
 trade_flag, 51
 XTPTickerPriceInfo, 51
 XTPTradeReport, 51
 xoms_api_fund_struct.h, 55
 xoms_api_struct.h, 56
 xquote_api_struct.h, 57
 xtp_api_data_type.h, 58
 EPT_INVALID, 62
 ERT_CASH_FORBIDDEN, 61
 ERT_CASH_MUST, 62
 ERT_CASH_OPTIONAL, 61
 ETF_REPLACE_TYPE, 61
 XTP_ACCOUNT_CREDIT, 62

XTP_ACCOUNT_DERIVE, 62
 XTP_ACCOUNT_NORMAL, 62
 XTP_ACCOUNT_TYPE, 62
 XTP_ACCOUNT_UNKNOWN, 62
 XTP_BUSINESS_TYPE, 62
 XTP_BUSINESS_TYPE_ALLOTMENT, 62
 XTP_BUSINESS_TYPE_CASH, 62
 XTP_BUSINESS_TYPE_DESIGNATION, 62
 XTP_BUSINESS_TYPE ETF, 62
 XTP_BUSINESS_TYPE_IPOS, 62
 XTP_BUSINESS_TYPE_MARGIN, 62
 XTP_BUSINESS_TYPE_MONEY_FUND, 62
 XTP_BUSINESS_TYPE_REPO, 62
 XTP_BUSINESS_TYPE_STRUCTURED_FUND←
 PURCHASE_REDEMPTION, 62
 XTP_BUSINESS_TYPE_STRUCTURED_FUND←
 SPLIT_MERGE, 62
 XTP_BUSINESS_TYPE_UNKNOWN, 62
 XTP_EXCHANGE_SH, 62
 XTP_EXCHANGE_SZ, 62
 XTP_EXCHANGE_TYPE, 62
 XTP_EXCHANGE_UNKNOWN, 62
 XTP_FUND_OPER_FAILED, 63
 XTP_FUND_OPER_PROCESSING, 63
 XTP_FUND_OPER_STATUS, 62
 XTP_FUND_OPER_SUBMITTED, 63
 XTP_FUND_OPER_SUCCESS, 63
 XTP_FUND_OPER_UNKNOWN, 63
 XTP_FUND_TRANSFER_IN, 63
 XTP_FUND_TRANSFER_OUT, 63
 XTP_FUND_TRANSFER_TYPE, 63
 XTP_FUND_TRANSFER_UNKNOWN, 63
 XTP_LOG_LEVEL, 63
 XTP_LOG_LEVEL_DEBUG, 63
 XTP_LOG_LEVEL_ERROR, 63
 XTP_LOG_LEVEL_FATAL, 63
 XTP_LOG_LEVEL_INFO, 63
 XTP_LOG_LEVEL_TRACE, 63
 XTP_LOG_LEVEL_WARNING, 63
 XTP_MARKET_TYPE, 63
 XTP_MKT_INIT, 63
 XTP_MKT_SH_A, 63
 XTP_MKT_SZ_A, 63
 XTP_MKT_UNKNOWN, 63
 XTP_ORDER_ACTION_STATUS_ACCEPTED,
 64
 XTP_ORDER_ACTION_STATUS_REJECTED, 64
 XTP_ORDER_ACTION_STATUS_SUBMITTED,
 64
 XTP_ORDER_ACTION_STATUS_TYPE, 63
 XTP_ORDER_STATUS_ALLTRADED, 64
 XTP_ORDER_STATUS_CANCELED, 64
 XTP_ORDER_STATUS_INIT, 64
 XTP_ORDER_STATUS_NOTRADEQUEUEING,
 64
 XTP_ORDER_STATUS_PARTTRADEDNOTQ←
 UEUEING, 64
 XTP_ORDER_STATUS_PARTTRADEDQUEUE←
 ING, 64
 XTP_ORDER_STATUS_REJECTED, 64
 XTP_ORDER_STATUS_TYPE, 64
 XTP_ORDER_STATUS_UNKNOWN, 64
 XTP_ORDER_SUBMIT_STATUS_CANCEL_AC←
 CEPTED, 64
 XTP_ORDER_SUBMIT_STATUS_CANCEL_RE←
 JECTED, 64
 XTP_ORDER_SUBMIT_STATUS_CANCEL_SU←
 BMITTED, 64
 XTP_ORDER_SUBMIT_STATUS_INSERT_AC←
 CEPTED, 64
 XTP_ORDER_SUBMIT_STATUS_INSERT_RE←
 JECTED, 64
 XTP_ORDER_SUBMIT_STATUS_INSERT_SU←
 BMITTED, 64
 XTP_ORDER_SUBMIT_STATUS_TYPE, 64
 XTP_PRICE_ALL_OR_CANCEL, 64
 XTP_PRICE_BEST5_OR_CANCEL, 64
 XTP_PRICE_BEST5_OR_LIMIT, 64
 XTP_PRICE_BEST_OR_CANCEL, 64
 XTP_PRICE_FORWARD_BEST, 64
 XTP_PRICE_LIMIT, 64
 XTP_PRICE_REVERSE_BEST_LIMIT, 64
 XTP_PRICE_TYPE, 64
 XTP_PRICE_TYPE_UNKNOWN, 64
 XTP_PROTOCOL_TCP, 65
 XTP_PROTOCOL_TYPE, 64
 XTP_PROTOCOL_UDP, 65
 XTP_SIDE_BUY, 65
 XTP_SIDE_BUY_CLOSE, 65
 XTP_SIDE_BUY_OPEN, 65
 XTP_SIDE_MERGE, 65
 XTP_SIDE_PURCHASE, 65
 XTP_SIDE_REDEMPTION, 65
 XTP_SIDE_SELL, 65
 XTP_SIDE_SELL_CLOSE, 65
 XTP_SIDE_SELL_OPEN, 65
 XTP_SIDE_SPLIT, 65
 XTP_SIDE_TYPE, 65
 XTP_SIDE_UNKNOWN, 65
 XTP_SPLIT_MERGE_STATUS, 65
 XTP_SPLIT_MERGE_STATUS_ALLOW, 65
 XTP_SPLIT_MERGE_STATUS_FORBIDDEN, 65
 XTP_SPLIT_MERGE_STATUS_ONLY_MERGE,
 65
 XTP_SPLIT_MERGE_STATUS_ONLY_SPLIT, 65
 XTP_TBT_ENTRUST, 65
 XTP_TBT_TRADE, 65
 XTP_TBT_TYPE, 65
 XTP_TE_RESUME_TYPE, 65
 XTP_TERT_QUICK, 66
 XTP_TERT_RESTART, 66
 XTP_TERT_RESUME, 66
 XTP_TICKER_TYPE, 66
 XTP_TICKER_TYPE_BOND, 66
 XTP_TICKER_TYPE_FUND, 66

XTP_TICKER_TYPE_INDEX, [66](#)
XTP_TICKER_TYPE_STOCK, [66](#)
XTP_TICKER_TYPE_UNKNOWN, [66](#)
xtp_api_struct.h, [66](#)
xtp_api_struct_common.h, [66](#)
xtp_trader_api.h, [67](#)