XTP极速交易系统TraderAPI

制作者 中泰证券股份有限公司

Contents

Chapter 1

XTP 极速交易系统 Trader API 2.2.33.5

本项目是XTP项目中的交易类+算法单接口

(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

(3) 程序化交易接口响应类Demo demo_test_trade_spi.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:

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:

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:

Chapter 5

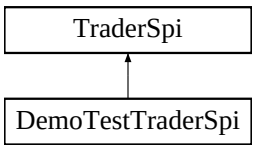
结构体说明

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)
  当客户端个某个连接与交易后台通信连接断开
- virtual void OnAlgoDisconnected (int reason)
- virtual void OnAlgoConnected ()
  当客户端与*AlgoBus*断线后重新连接时，该方法被调用，仅在断线重连成功后会被调用。
- virtual void OnQueryStrategy (XTPStrategyInfoStruct *strategy_info, char *strategy_param, XTPRI *error_info, int32_t request_id, bool is_last, uint64_t session_id)
- virtual void OnStrategyStateReport (XTPStrategyStateReportStruct *strategy_state, uint64_t session_id)
- virtual void OnALGOUserEstablishChannel (char *user, XTPRI *error_info, uint64_t session_id)
- virtual void OnInsertAlgoOrder (XTPStrategyInfoStruct *strategy_info, XTPRI *error_info, uint64_t session_id)
- virtual void OnCancelAlgoOrder (XTPStrategyInfoStruct *strategy_info, XTPRI *error_info, 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 OnAlgoConnected()

```
void OnAlgoConnected ( )  [virtual]
```

当客户端与AlgoBus断线后重新连接时，该方法被调用，仅在断线重连成功后会被调用。

与算法服务器重新建立起连接，仅在断连重连成功后通知

重载 TraderSpi .

5.1.2.2 OnAlgoDisconnected()

```
void OnAlgoDisconnected (
            int reason )  [virtual]
```

当客户端与AlgoBus通信连接断开时，该方法被调用。

参数

reason	错误原因，请与错误代码表对应

备注

请不要堵塞此线程，否则会影响algo的登录，与Algo之间的连接，断线后会自动重连，用户无需做其他操作

与算法服务器断线，此时api会自动与算法服务器重连，无需用户重连

重载 TraderSpi .

5.1.2.3 OnALGOUserEstablishChannel()

```
void OnALGOUserEstablishChannel (
            char * user,
            XTPRI * error_info,
            uint64_t session_id )  [virtual]
```

algo业务中用户建立算法通道的消息响应

参数

user	用户名
error_info	建立算法通道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，即算法通道成功
session_id	资金账户对应的session_id，登录时得到

备注

算法通道建立成功后，才能对用户创建策略等操作，一个用户只能拥有一个算法通道，如果之前已经建立，则无需重复建立

建立算法通道的异步通知

建立算法通道成功后，可以下算法母单

读取算法单的参数

发送算法单

算法单发送成功

算法单发送失败

重载 TraderSpi .

5.1.2.4 OnCancelAlgoOrder()

```
void OnCancelAlgoOrder (
            XTPStrategyInfoStruct * strategy_info,
            XTPRI * error_info,
            uint64_t session_id ) [virtual]
```

algo业务中撤销策略单的响应

参数

strategy_info	用户撤销的策略单的具体信息
error_info	撤销策略单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
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

算法单撤销结果通知

算法单撤销失败

算法单撤销成功

重载 TraderSpi .

5.1.2.5 OnDisconnected()

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

当客户端个某个连接与交易后台通信连接断开

用户与交易服务器断连，需要用户重新建立连接，并更新session id

重载 TraderSpi .

5.1.2.6 OnInsertAlgoOrder()

```
void OnInsertAlgoOrder (
            XTPStrategyInfoStruct * strategy_info,
            XTPRI * error_info,
            uint64_t session_id )  [virtual]
```

algo业务中报送策略单的响应

参数

strategy_info	用户报送的策略单的具体信息
error_info	报送策略单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
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

发送算法单后的异步通知

算法单建立成功

需要撤销算法单的时候，可以发送撤销算法单消息

撤销算法单发送成功

撤销算法单发送失败

重载 TraderSpi .

5.1.2.7 OnQueryStrategy()

```
void OnQueryStrategy (
            XTPStrategyInfoStruct * strategy_info,
            char * strategy_param,
            XTPRI * error_info,
            int32_t request_id,
            bool is_last,
            uint64_t session_id ) [virtual]
```

algo业务中查询策略列表的响应

参数

strategy_info	策略具体信息
strategy_param	此策略中包含的参数，如果error_info.error_id为0时，有意义
error_info	查询查询策略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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

算法单查询结果通知

重载 TraderSpi .

5.1.2.8 OnStrategyStateReport()

```
void OnStrategyStateReport (
            XTPStrategyStateReportStruct * strategy_state,
            uint64_t session_id ) [virtual]
```

algo业务中策略运行时策略状态通知

参数

strategy_state	用户策略运行情况的状态通知
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

重载 TraderSpi .

该类的文档由以下文件生成:

- demo_test_trade_spi.h
- demo_test_trade_spi.cpp

5.2 OrderBookStruct结构体 参考

订单薄

`#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

订单薄

该结构体的文档由以下文件生成:

- xquote_api_struct.h

5.3 TraderApi类 参考

交易接口类

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

Public 成员函数

- virtual void Release ()=0
- virtual const char * GetTradingDay ()=0
- virtual void RegisterSpi (TraderSpi *spi)=0
- virtual XTPRI * GetApiLastError ()=0
- virtual const char * GetApiVersion ()=0
- virtual uint8_t GetClientIDByXTPID (uint64_t order_xtp_id)=0
- virtual const char * GetAccountByXTPID (uint64_t order_xtp_id)=0
- virtual void SubscribePublicTopic (XTP_TE_RESUME_TYPE resume_type)=0
- virtual void SetSoftwareVersion (const char *version)=0
- virtual void SetSoftwareKey (const char *key)=0
- virtual void SetHeartBeatInterval (uint32_t interval)=0
- virtual uint64_t Login (const char *ip, int port, const char *user, const char *password, XTP_PROTOCOL_TYPE sock_type, const char *local_ip=NULL)=0
- virtual int Logout (uint64_t session_id)=0
- virtual bool IsServerRestart (uint64_t session_id)=0
- virtual int ModifyUserTerminalInfo (const XTPUserTerminalInfoReq *info, uint64_t session_id)=0
- virtual int QueryAccountTradeMarket (uint64_t session_id, int request_id)=0
- virtual uint64_t GetANewOrderXTPID (uint64_t session_id)=0
- virtual uint64_t InsertOrder (XTPOrderInsertInfo *order, uint64_t session_id)=0
- virtual uint64_t InsertOrderExtra (XTPOrderInsertInfo *order, uint64_t session_id)=0
- virtual uint64_t CancelOrder (const uint64_t order_xtp_id, uint64_t session_id)=0
- virtual int QueryOrderByXTPID (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryOrders (const XTPQueryOrderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryUnfinishedOrders (uint64_t session_id, int request_id)=0
- virtual int QueryOrdersByPage (const XTPQueryOrderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOrderByXTPIDEx (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryOrdersEx (const XTPQueryOrderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryUnfinishedOrdersEx (uint64_t session_id, int request_id)=0
- virtual int QueryOrdersByPageEx (const XTPQueryOrderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryTradesByXTPID (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryTrades (XTPQueryTraderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryTradesByPage (const XTPQueryTraderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryPosition (const char *ticker, uint64_t session_id, int request_id, XTP_MARKET_TYPE market=XTP_MKT_INIT)=0
- virtual int QueryAsset (uint64_t session_id, int request_id)=0
- virtual int QueryStructuredFund (XTPQueryStructuredFundInfoReq *query_param, uint64_t session_id, int request_id)=0
- virtual uint64_t FundTransfer (XTPFundTransferReq *fund_transfer, uint64_t session_id)=0
- virtual int QueryFundTransfer (XTPQueryFundTransferLogReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOtherServerFund (XTPFundQueryReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryETF (XTPQueryETFBaseReq *query_param, uint64_t session_id, int request_id)=0

- virtual int QueryETFTickerBasket (XTPQueryETFComponentReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryIPOInfoList (uint64_t session_id, int request_id)=0
- virtual int QueryIPOQuotaInfo (uint64_t session_id, int request_id)=0
- virtual int QueryOptionAuctionInfo (XTPQueryOptionAuctionInfoReq *query_param, uint64_t session_id, int request_id)=0
- virtual uint64_t CreditCashRepay (double amount, uint64_t session_id)=0
- virtual uint64_t CreditCashRepayDebtInterestFee (const char *debt_id, double amount, uint64_t session_id)=0
- virtual uint64_t CreditSellStockRepayDebtInterestFee (XTPOrderInsertInfo *order, const char *debt_id, uint64_t session_id)=0
- virtual int QueryCreditCashRepayInfo (uint64_t session_id, int request_id)=0
- virtual int QueryCreditFundInfo (uint64_t session_id, int request_id)=0
- virtual int QueryCreditDebtInfo (uint64_t session_id, int request_id)=0
- virtual int QueryCreditTickerDebtInfo (XTPClientQueryCrdDebtStockReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryCreditAssetDebtInfo (uint64_t session_id, int request_id)=0
- virtual int QueryCreditTickerAssignInfo (XTPClientQueryCrdPositionStockReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryCreditExcessStock (XTPClientQueryCrdSurplusStkReqInfo *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryMulCreditExcessStock (XTPClientQueryCrdSurplusStkReqInfo *query_param, uint64_t session_id, int request_id)=0
- virtual uint64_t CreditExtendDebtDate (XTPCreditDebtExtendReq *debt_extend, uint64_t session_id)=0
- virtual int QueryCreditExtendDebtDateOrders (uint64_t xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryCreditFundExtraInfo (uint64_t session_id, int request_id)=0
- virtual int QueryCreditPositionExtraInfo (XTPClientQueryCrdPositionStockReq *query_param, uint64_t session_id, int request_id)=0
- virtual uint64_t InsertOptionCombinedOrder (XTPOptCombOrderInsertInfo *order, uint64_t session_id)=0
- virtual uint64_t InsertOptionCombinedOrderExtra (XTPOptCombOrderInsertInfo *order, uint64_t session_id)=0
- virtual uint64_t CancelOptionCombinedOrder (const uint64_t order_xtp_id, uint64_t session_id)=0
- virtual int QueryOptionCombinedUnfinishedOrders (uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedOrderByXTPID (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedOrders (const XTPQueryOptCombOrderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedOrdersByPage (const XTPQueryOptCombOrderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedUnfinishedOrdersEx (uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedOrderByXTPIDEx (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedOrdersEx (const XTPQueryOptCombOrderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedOrdersByPageEx (const XTPQueryOptCombOrderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedTradesByXTPID (const uint64_t order_xtp_id, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedTrades (const XTPQueryOptCombTraderReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedTradesByPage (const XTPQueryOptCombTraderByPageReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedPosition (const XTPQueryOptCombPositionReq *query_param, uint64_t session_id, int request_id)=0
- virtual int QueryOptionCombinedStrategyInfo (uint64_t session_id, int request_id)=0

- virtual int QueryOptionCombinedExecPosition (const XTPQueryOptCombExecPosReq *query_param, uint64_t session_id, int request_id)=0
- virtual int LoginALGO (const char *ip, int port, const char *user, const char *password, XTP_PROTOCOL_TYPE sock_type, const char *local_ip=NULL)=0
- virtual int QueryStrategy (uint32_t strategy_type, uint64_t client_strategy_id, uint64_t xtp_strategy_id, uint64_t session_id, int32_t request_id)=0
- virtual int ALGOUserEstablishChannel (const char *oms_ip, int oms_port, const char *user, const char *password, uint64_t session_id)=0
- virtual int InsertAlgoOrder (uint32_t strategy_type, uint64_t client_strategy_id, char *strategy_param, uint64_t session_id)=0
- virtual int CancelAlgoOrder (bool cancel_flag, uint64_t xtp_strategy_id, uint64_t session_id)=0
- virtual uint64_t GetAlgorithmIDByOrder (uint64_t order_xtp_id, uint32_t order_client_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 ALGOUserEstablishChannel()

```
virtual int ALGOUserEstablishChannel (
            const char * oms_ip,
            int oms_port,
            const char * user,
            const char * password,
            uint64_t session_id ) [pure virtual]
```

用户请求使用algo服务器建立算法通道

返回

表明此资金账号建立算法通道请求消息发送是否成功，非“0”表示发送失败，可以调用GetApiLastError()来获取错误代码，“0”表示发送成功

参数

oms_ip	oms服务器地址，类似“127.0.0.1”，非algo服务器地址
oms_port	oms服务器端口号，非algo服务器端口号
user	登录用户名
password	登录密码
session_id	资金账户对应的session_id,登录时得到

备注

此函数为异步方式，一个用户只能拥有一个算法通道，如果之前已经建立，则无需重复建立，在使用算法前，请先建立算法通道

5.3.2.2 CancelAlgoOrder()

```
virtual int CancelAlgoOrder (
            bool cancel_flag,
            uint64_t xtp_strategy_id,
            uint64_t session_id ) [pure virtual]
```

algo业务中用户撤销算法单请求

返回

请求发送是否成功，“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

cancel_flag	是否需要算法去处理已下的算法子单标志，true-交给算法自行处理，包括撤单、平仓等，算法处理完成后会通知客户；false-立即停止算法母单的执行，此时算法平台会对已下的子单做撤单操作，其余的平仓等操作需要客户自己处理
xtp_strategy_id	需要撤销的算法单在xtp algobus系统中的id
session_id	资金账户对应的session_id,登录时得到

备注

仅能在用户建立算法通道后调用

5.3.2.3 CancelOptionCombinedOrder()

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

期权组合策略报单撤单请求

返回

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

参数

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

备注

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

5.3.2.4 CancelOrder()

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

报单操作请求

返回

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

参数

order_xtp_id	需要撤销的委托单在XTP系统中的ID
session_id	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

5.3.2.5 CreateTraderApi()

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

创建TraderApi

参数

client_id	（必须输入）客户端id，用于区分同一用户的不同客户端，由用户自定义，普通用户必须使用1-99之间的数值
save_file_path	（必须输入）存贮订阅信息文件的目录，请设定一个真实存在的有可写权限的路径
log_level	日志输出级别

返回

创建出的UserApi

备注

只能创建一次，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

5.3.2.6 CreditCashRepay()

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

融资融券业务中现金直接还款请求

返回

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

参数

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

5.3.2.7 CreditCashRepayDebtInterestFee()

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

融资融券业务中现金还指定负债合约息费请求

返回

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

参数

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

5.3.2.8 CreditExtendDebtDate()

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

融资融券业务中请求负债合约展期

返回

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

参数

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

5.3.2.9 CreditSellStockRepayDebtInterestFee()

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

融资融券业务中卖券还指定负债合约息费请求

返回

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

参数

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

5.3.2.10 FundTransfer()

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

资金划拨请求

返回

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

参数

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

备注

此函数支持一号两中心节点之间的资金划拨，注意资金划拨的方向。

5.3.2.11 GetAccountByXTPID()

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

通过报单在xtp系统中的ID获取相关资金账户名

返回

返回资金账户名

参数

order_xtp_id	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

5.3.2.12 GetAlgorithmIDByOrder()

```
virtual uint64_t GetAlgorithmIDByOrder (
            uint64_t order_xtp_id,
            uint32_t order_client_id )  [pure virtual]
```

获取算法单的母单ID

返回

返回算法单的母单ID，如果返回为0表示不是算法单

参数

order_xtp_id	算法单对应的xtp id
order_client↩ _id	算法单对应的自定义ID，不可随意填写

备注

返回为0表示，不是算法单，如果传入的参数不对的话，可能会得不到正确结果，此函数调用不依赖于是否登录

5.3.2.13 GetANewOrderXTPID()

```
virtual uint64_t GetANewOrderXTPID (
            uint64_t session_id )  [pure virtual]
```

为用户获取一个新的订单XTPID，用于报单

返回

生成的订单XTPID，非“0”表示获取成功，“0”表示获取失败，此时用户可以调用GetApiLastError()来获取错误代码

参数

session↩ _id	资金账户对应的session_id,登录时得到

备注

此函数必须在Login之后调用，通过这个函数获取的order_xtp_id仅用于对应的用户报单，如果设置错误，将会导致下单失败

5.3.2.14 GetApiLastError()

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

获取API的系统错误

返回

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

备注

可以在调用api接口失败时调用，例如login失败时

5.3.2.15 GetApiVersion()

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

获取API的发行版本号

返回

返回api发行版本号

5.3.2.16 GetClientIDByXTPID()

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

通过报单在xtp系统中的ID获取下单的客户端id

返回

返回客户端id，可以用此方法过滤自己下的订单

参数

order_xtp_id	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

5.3.2.17 GetTradingDay()

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

获取当前交易日

返回

获取到的交易日

备注

只有登录成功后,才能得到正确的交易日

5.3.2.18 InsertAlgoOrder()

```
virtual int InsertAlgoOrder (
            uint32_t strategy_type,
            uint64_t client_strategy_id,
            char * strategy_param,
            uint64_t session_id ) [pure virtual]
```

algo业务中用户报算法单请求

返回

算法报单请求发送是否成功，“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

strategy_type	需要创建的策略类型
client_strategy_id	用户自定义id，帮助用户定位
strategy_param	策略参数
session_id	资金账户对应的session_id,登录时得到

备注

仅能在用户建立算法通道后使用，算法单的异步通知得

5.3.2.19 InsertOptionCombinedOrder()

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

期权组合策略报单录入请求

返回

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

参数

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

备注

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

5.3.2.20 InsertOptionCombinedOrderExtra()

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

已经提前设置order_xtp_id的期权组合策略报单录入请求，与GetANewOrderXTPID()配合使用

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，此时等同与传入的order_xtp_id

参数

order	报单录入信息，其中order.order_client_id字段是用户自定义字段，用户输入什么值，订单响应OnOrderEvent()返回时就会带回什么值，类似于备注，方便用户自己定位订单，也可以什么都不填。order.order_xtp_id字段必须是通过GetANewOrderXTPID()获得的值，order.ticker必须不带空格，以'\0'结尾
session_id	资金账户对应的session_id,登录时得到

备注

使用设置好的order_xtp_id（通过GetANewOrderXTPID()获得）进行报单，注意此处如果order_xtp_id设置不对，将导致报单失败。交易所接收订单后，会在报单响应函数OnOptionCombinedOrderEvent()中返回报单未成交的状态，之后所有的订单状态改变（除了部成状态）都会通过报单响应函数返回

5.3.2.21 InsertOrder()

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

报单录入请求

返回

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

参数

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

备注

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

5.3.2.22 InsertOrderExtra()

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

已经提前设置order_xtp_id的报单录入请求，与GetANewOrderXTPID()配合使用

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，此时等同与传入的order_xtp_id

参数

order	报单录入信息，其中order.order_client_id字段是用户自定义字段，用户输入什么值，订单响应OnOrderEvent()返回时就会带回什么值，类似于备注，方便用户自己定位订单，也可以什么都不填。order.order_xtp_id字段必须是通过GetANewOrderXTPID()获得的值，order.ticker必须不带空格，以'\0'结尾
session↩ _id	资金账户对应的session_id,登录时得到

备注

使用设置好的order_xtp_id（通过GetANewOrderXTPID()获得）进行报单，注意此处如果order_xtp_↩ id设置不对，将导致报单失败。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

5.3.2.23 IsServerRestart()

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

服务器是否重启过

返回

“true”表示重启过，“false”表示没有重启过

参数

session_id	资金账户对应的session_id,登录时得到

备注

此函数必须在Login之后调用

5.3.2.24 Login()

```
virtual uint64_t Login (
            const char * ip,
            int port,
            const char * user,
            const char * password,
            XTP_PROTOCOL_TYPE sock_type,
            const char * local_ip = NULL )  [pure virtual]
```

用户登录请求

返回

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

参数

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

备注

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

5.3.2.25 LoginALGO()

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

用户登录algo服务器请求

返回

表明此资金账号登录是否成功，非“0”表示登录失败，可以调用GetApiLastError()来获取错误代码，“0”表示登录成功

参数

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

备注

此函数为同步阻塞式，不需要异步等待登录成功，当函数返回即可进行后续操作，此api只需调用一次，所有用户共用即可

5.3.2.26 Logout()

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

登出请求

返回

登出是否成功，“0”表示登出成功，“-1”表示登出失败

参数

session↩ _id	资金账户对应的session_id,登录时得到

5.3.2.27 ModifyUserTerminalInfo()

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

修改已登录用户的硬件信息，仅限授权系统使用

返回

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

参数

info	需要修改成的用户硬件信息
session↩ _id	资金账户对应的session_id,登录时得到

备注

此函数必须在Login之后调用，且仅限授权系统使用，一般客户无需使用

5.3.2.28 QueryAccountTradeMarket()

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

查询用户在本节点上的可交易市场类型

返回

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

参数

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

备注

此函数必须在Login之后调用，对应的响应函数是OnQueryAccountTradeMarket()

5.3.2.29 QueryAsset()

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

请求查询资产

返回

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

参数

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

5.3.2.30 QueryCreditAssetDebtInfo()

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

请求查询信用账户待还资金信息

返回

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

参数

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

5.3.2.31 QueryCreditCashRepayInfo()

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

请求查询融资融券业务中的现金直接还款报单

返回

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

参数

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

5.3.2.32 QueryCreditDebtInfo()

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

请求查询信用账户负债合约信息

返回

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

参数

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

5.3.2.33 QueryCreditExcessStock()

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

融资融券业务中请求查询指定证券的余券

返回

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

参数

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

备注

该方法中用户必须提供了证券代码和所在市场

5.3.2.34 QueryCreditExtendDebtDateOrders()

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

融资融券业务中请求查询负债合约展期

返回

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

参数

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

5.3.2.35 QueryCreditFundExtraInfo()

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

请求查询融资融券业务中账的附加信息

返回

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

参数

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

5.3.2.36 QueryCreditFundInfo()

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

请求查询信用账户特有信息，除资金账户以外的信息

返回

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

参数

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

5.3.2.37 QueryCreditPositionExtraInfo()

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

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

请求查询融资融券业务中账指定证券的附加信息

返回

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

参数

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

5.3.2.38 QueryCreditTickerAssignInfo()

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

请求查询信用账户可融券头寸信息

返回

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

参数

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

5.3.2.39 QueryCreditTickerDebtInfo()

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

请求查询指定证券负债未还信息

返回

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

参数

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

5.3.2.40 QueryETF()

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

请求查询ETF清单文件

返回

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

参数

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

5.3.2.41 QueryETFTickerBasket()

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

请求查询ETF股票篮

返回

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

参数

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

5.3.2.42 QueryFundTransfer()

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

请求查询资金划拨

返回

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

参数

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

5.3.2.43 QueryIPOInfoList()

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

请求查询今日新股申购信息列表

返回

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

参数

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

5.3.2.44 QueryIPOQuotaInfo()

```
virtual int QueryIPOQuotaInfo (
```

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

请求查询用户新股申购额度信息

返回

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

参数

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

5.3.2.45 QueryMulCreditExcessStock()

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

融资融券业务中请求查询余券

返回

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

参数

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

5.3.2.46 QueryOptionAuctionInfo()

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

请求查询期权合约

返回

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

参数

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

5.3.2.47 QueryOptionCombinedExecPosition()

```
virtual int QueryOptionCombinedExecPosition (
            const XTPQueryOptCombExecPosReq * query_param,
            uint64_t session_id,
            int request_id ) [pure virtual]
```

请求查询期权行权合并头寸

返回

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

参数

query_param	需要查询的行权合并的筛选条件，其中market为0会默认查询全市场，成分合约代码可以初始化为空，如果不为空，请不带空格，并以'\0'结尾，注意所有填写的条件都会进行匹配
session_id	资金账户对应的session_id,登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

备注

该方法可能对应多条响应消息

5.3.2.48 QueryOptionCombinedOrderByXTPID()

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

根据报单ID请求查询期权组合策略报单-旧版本接口

返回

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

参数

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

5.3.2.49 QueryOptionCombinedOrderByXTPIDEx()

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

根据报单ID请求查询期权组合策略报单-新版本接口

返回

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

参数

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

5.3.2.50 QueryOptionCombinedOrders()

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

请求查询期权组合策略报单-旧版本接口

返回

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

参数

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

备注

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

5.3.2.51 QueryOptionCombinedOrdersByPage()

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

分页请求查询期权组合策略报单-旧版本接口

返回

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

参数

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

备注

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

5.3.2.52 QueryOptionCombinedOrdersByPageEx()

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

分页请求查询期权组合策略报单-新版本接口

返回

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

参数

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

备注

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

5.3.2.53 QueryOptionCombinedOrdersEx()

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

请求查询期权组合策略报单-新版本接口

返回

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

参数

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

备注

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

5.3.2.54 QueryOptionCombinedPosition()

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

请求查询投资者期权组合策略持仓

返回

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

参数

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

备注

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

5.3.2.55 QueryOptionCombinedStrategyInfo()

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

请求查询期权组合策略信息

返回

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

参数

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

备注

该方法仅支持精确查询，不支持模糊查询

5.3.2.56 QueryOptionCombinedTrades()

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

请求查询期权组合策略的成交回报

返回

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

参数

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

备注

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

5.3.2.57 QueryOptionCombinedTradesByPage()

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

分页请求查询期权组合策略成交回报

返回

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

参数

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

备注

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

5.3.2.58 QueryOptionCombinedTradesByXTPID()

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

根据期权组合策略委托编号请求查询相关成交

返回

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

参数

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

备注

此函数查询出的结果可能对应多个查询结果响应

5.3.2.59 QueryOptionCombinedUnfinishedOrders()

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

请求查询期权组合策略未完结报单-旧版本接口

返回

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

参数

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

5.3.2.60 QueryOptionCombinedUnfinishedOrdersEx()

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

请求查询期权组合策略未完结报单-新版本接口

返回

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

参数

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

5.3.2.61 QueryOrderByXTPID()

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

根据报单ID请求查询报单-旧版本接口

返回

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

参数

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

5.3.2.62 QueryOrderByXTPIDEx()

```
virtual int QueryOrderByXTPIDEx (
            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-新版本接口

返回

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

参数

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

5.3.2.63 QueryOrders()

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

请求查询报单-旧版本接口

返回

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

参数

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

备注

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

5.3.2.64 QueryOrdersByPage()

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

分页请求查询报单-旧版本接口

返回

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

参数

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

备注

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

5.3.2.65 QueryOrdersByPageEx()

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

分页请求查询报单-新版本接口

返回

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

参数

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

备注

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

5.3.2.66 QueryOrdersEx()

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

请求查询报单-新版本接口

返回

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

参数

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

备注

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

5.3.2.67 QueryOtherServerFund()

```
virtual int QueryOtherServerFund (
            XTPFundQueryReq * query_param,
            uint64_t session_id,
            int request_id ) [pure virtual]
```

请求查询其他节点可用资金

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

query_param	查询时需要提供的信息
session_id	资金账户对应的session_id,登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

5.3.2.68 QueryPosition()

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

请求查询投资者持仓

返回

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

参数

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

备注

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

5.3.2.69 QueryStrategy()

```
virtual int QueryStrategy (
            uint32_t strategy_type,
            uint64_t client_strategy_id,
            uint64_t xtp_strategy_id,
            uint64_t session_id,
            int32_t request_id )  [pure virtual]
```

algo业务中查询用户策略请求

返回

请求发送是否成功，“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

strategy_type	需要查询的策略类型，可填0
client_strategy_id	需要查询的策略用户自定义id，可填0
xtp_strategy_id	需要查询的策略在xtp系统中的id，如果指定，就一定按指定查询，如果填0，则按其他筛选条件查询
session_id	资金账户对应的session_id,登录时得到
request_id	用于用户定位查询响应的ID，由用户自定义

备注

xtp_strategy_id条件的优先级最高，只有当xtp_strategy_id为0时，其他条件才生效，此条请求可能对应多条回应消息

5.3.2.70 QueryStructuredFund()

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

请求查询分级基金

返回

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

参数

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

备注

此函数查询出的结果可能对应多个查询结果响应

5.3.2.71 QueryTrades()

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

请求查询已成交

返回

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

参数

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

备注

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

5.3.2.72 QueryTradesByPage()

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

分页请求查询成交回报

返回

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

参数

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

备注

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

5.3.2.73 QueryTradesByXTPID()

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

根据委托编号请求查询相关成交

返回

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

参数

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

备注

此函数查询出的结果可能对应多个查询结果响应

5.3.2.74 QueryUnfinishedOrders()

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

请求查询未完结报单-旧版本接口

返回

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

参数

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

5.3.2.75 QueryUnfinishedOrdersEx()

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

请求查询未完结报单-新版本接口

返回

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

参数

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

5.3.2.76 RegisterSpi()

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

注册回调接口

参数

spi	派生自回调接口类的实例，请在登录之前设定

5.3.2.77 Release()

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

删除接口对象本身

备注

不再使用本接口对象时,调用该函数删除接口对象

5.3.2.78 SetHeartBeatInterval()

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

设置心跳检测时间间隔，单位为秒

参数

interval	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

5.3.2.79 SetSoftwareKey()

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

设置软件开发Key

参数

key	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

5.3.2.80 SetSoftwareVersion()

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

设置软件开发版本号

参数

version	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

5.3.2.81 SubscribePublicTopic()

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

订阅公共流。

参数

resume_type	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:

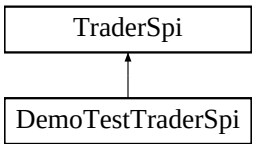
- xtp_trader_api.h

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 OnQueryAccountTradeMarket (int trade_location, XTPRI *error_info, int request_id, uint64_t session_id)
- 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 OnQueryOrderEx (XTPOrderInfoEx *order_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOrderByPage (XTPQueryOrderRsp *order_info, int64_t req_count, int64_t order_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOrderByPageEx (XTPOrderInfoEx *order_info, int64_t req_count, int64_t order_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryTrade (XTPQueryTradeRsp *trade_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryTradeByPage (XTPQueryTradeRsp *trade_info, int64_t req_count, int64_t trade_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryPosition (XTPQueryStkPositionRsp *position, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryAsset (XTPQueryAssetRsp *asset, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryStructuredFund (XTPStructuredFundInfo *fund_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryFundTransfer (XTPFundTransferNotice *fund_transfer_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnFundTransfer (XTPFundTransferNotice *fund_transfer_info, XTPRI *error_info, uint64_t session_id)
- virtual void OnQueryOtherServerFund (XTPFundQueryRsp *fund_info, XTPRI *error_info, int request_id, uint64_t session_id)
- virtual void OnQueryETF (XTPQueryETFBaseRsp *etf_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryETFBasket (XTPQueryETFComponentRsp *etf_component_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryIPOInfoList (XTPQueryIPOTickerRsp *ipo_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryIPOQuotaInfo (XTPQueryIPOQuotaRsp *quota_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOptionAuctionInfo (XTPQueryOptionAuctionInfoRsp *option_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnCreditCashRepay (XTPCrdCashRepayRsp *cash_repay_info, XTPRI *error_info, uint64_t session_id)
- virtual void OnCreditCashRepayDebtInterestFee (XTPCrdCashRepayDebtInterestFeeRsp *cash_repay_info, XTPRI *error_info, uint64_t session_id)
- virtual void OnQueryCreditCashRepayInfo (XTPCrdCashRepayInfo *cash_repay_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)

- virtual void OnQueryCreditFundInfo (XTPCrdFundInfo *fund_info, XTPRI *error_info, int request_id, uint64_t session_id)
- virtual void OnQueryCreditDebtInfo (XTPCrdDebtInfo *debt_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryCreditTickerDebtInfo (XTPCrdDebtStockInfo *debt_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryCreditAssetDebtInfo (double remain_amount, XTPRI *error_info, int request_id, uint64_t session_id)
- virtual void OnQueryCreditTickerAssignInfo (XTPClientQueryCrdPositionStkInfo *assign_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryCreditExcessStock (XTPClientQueryCrdSurplusStkRspInfo *stock_info, XTPRI *error_info, int request_id, uint64_t session_id)
- virtual void OnQueryMulCreditExcessStock (XTPClientQueryCrdSurplusStkRspInfo *stock_info, XTPRI *error_info, int request_id, uint64_t session_id, bool is_last)
- virtual void OnCreditExtendDebtDate (XTPCreditDebtExtendNotice *debt_extend_info, XTPRI *error_info, uint64_t session_id)
- virtual void OnQueryCreditExtendDebtDateOrders (XTPCreditDebtExtendNotice *debt_extend_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryCreditFundExtraInfo (XTPCrdFundExtraInfo *fund_info, XTPRI *error_info, int request_id, uint64_t session_id)
- virtual void OnQueryCreditPositionExtraInfo (XTPCrdPositionExtraInfo *fund_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnOptionCombinedOrderEvent (XTPOptCombOrderInfo *order_info, XTPRI *error_info, uint64_t session_id)
- virtual void OnOptionCombinedTradeEvent (XTPOptCombTradeReport *trade_info, uint64_t session_id)
- virtual void OnCancelOptionCombinedOrderError (XTPOptCombOrderCancelInfo *cancel_info, XTPRI *error_info, uint64_t session_id)
- virtual void OnQueryOptionCombinedOrders (XTPQueryOptCombOrderRsp *order_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOptionCombinedOrdersEx (XTPOptCombOrderInfoEx *order_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOptionCombinedOrdersByPage (XTPQueryOptCombOrderRsp *order_info, int64_t req_count, int64_t order_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOptionCombinedOrdersByPageEx (XTPOptCombOrderInfoEx *order_info, int64_t req_count, int64_t order_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOptionCombinedTrades (XTPQueryOptCombTradeRsp *trade_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOptionCombinedTradesByPage (XTPQueryOptCombTradeRsp *trade_info, int64_t req_count, int64_t trade_sequence, int64_t query_reference, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOptionCombinedPosition (XTPQueryOptCombPositionRsp *position_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOptionCombinedStrategyInfo (XTPQueryCombineStrategyInfoRsp *strategy_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryOptionCombinedExecPosition (XTPQueryOptCombExecPosRsp *position_info, XTPRI *error_info, int request_id, bool is_last, uint64_t session_id)
- virtual void OnQueryStrategy (XTPStrategyInfoStruct *strategy_info, char *strategy_param, XTPRI *error_info, int32_t request_id, bool is_last, uint64_t session_id)
- virtual void OnStrategyStateReport (XTPStrategyStateReportStruct *strategy_state, uint64_t session_id)
- virtual void OnALGOUserEstablishChannel (char *user, XTPRI *error_info, uint64_t session_id)
- virtual void OnInsertAlgoOrder (XTPStrategyInfoStruct *strategy_info, XTPRI *error_info, uint64_t session_id)
- virtual void OnCancelAlgoOrder (XTPStrategyInfoStruct *strategy_info, XTPRI *error_info, uint64_t session_id)
- virtual void OnAlgoDisconnected (int reason)

- virtual void OnAlgoConnected ()
  当客户端与*AlgoBus*断线后重新连接时，该方法被调用，仅在断线重连成功后会被调用。
- virtual void OnStrategySymbolStateReport (XTPStrategySymbolStateReport *strategy_symbol_state, 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 OnAlgoDisconnected()

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

当客户端与AlgoBus通信连接断开时，该方法被调用。

参数

reason	错误原因，请与错误代码表对应

备注

请不要堵塞此线程，否则会影响algo的登录，与Algo之间的连接，断线后会自动重连，用户无需做其他操作

被 DemoTestTraderSpi 重载.

5.4.2.2 OnALGOUserEstablishChannel()

```
virtual void OnALGOUserEstablishChannel (
            char * user,
            XTPRI * error_info,
            uint64_t session_id )  [inline], [virtual]
```

algo业务中用户建立算法通道的消息响应

参数

user	用户名
error_info	建立算法通道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，即算法通道成功
session_id	资金账户对应的session_id，登录时得到

备注

算法通道建立成功后，才能对用户创建策略等操作，一个用户只能拥有一个算法通道，如果之前已经建立，则无需重复建立

被 DemoTestTraderSpi 重载.

5.4.2.3 OnCancelAlgoOrder()

```
virtual void OnCancelAlgoOrder (
            XTPStrategyInfoStruct * strategy_info,
            XTPRI * error_info,
            uint64_t session_id )  [inline], [virtual]
```

algo业务中撤销策略单的响应

参数

strategy_info	用户撤销的策略单的具体信息
error_info	撤销策略单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
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 DemoTestTraderSpi 重载.

5.4.2.4 OnCancelOptionCombinedOrderError()

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

期权组合策略撤单出错响应

参数

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

备注

此响应只会在撤单发生错误时被回调

5.4.2.5 OnCancelOrderError()

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

撤单出错响应

参数

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

备注

此响应只会在撤单发生错误时被回调

5.4.2.6 OnCreditCashRepay()

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

融资融券业务中现金直接还款的响应

参数

cash_repay_info	现金直接还款通知的具体信息，用户可以通过cash_repay_info.xtp_id来管理订单，通过GetClientIDByXTPID() == client_id来过滤自己的订单。
error_info	现金还款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
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.7 OnCreditCashRepayDebtInterestFee()

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

融资融券业务中现金还息的响应

参数

cash_repay_info	现金还息通知的具体信息，用户可以通过cash_repay_info.xtp_id来管理订单，通过GetClientIDByXTPID() == client_id来过滤自己的订单。
error_info	现金还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
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.8 OnCreditExtendDebtDate()

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

融资融券业务中负债合约展期的通知

参数

debt_extend_info	负债合约展期通知的具体信息，用户可以通过debt_extend_info.xtpid来管理订单，通过GetClientIDByXTPID() == client_id来过滤自己的订单。
error_info	负债合约展期订单被拒绝或者发生错误时错误代码和错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误。
session_id	资金账户对应的session_id，登录时得到

备注

当负债合约展期订单有状态变化的时候，会被调用，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。所有登录了此用户的客户端都将收到此用户的负债合约展期通知。

5.4.2.9 OnDisconnected()

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

当客户端的某个连接与交易后台通信连接断开时，该方法被调用。

参数

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

备注

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

被 DemoTestTraderSpi 重载.

5.4.2.10 OnError()

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

错误应答

参数

error_info	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

5.4.2.11 OnFundTransfer()

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

资金划拨通知

参数

fund_transfer_info	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。
error_info	资金划拨订单被拒绝或者发生错误时错误代码和错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误。当资金划拨方向为一号两中心节点之间划拨，且error_info.error_id=11000384时，error_info.error_msg中含有对方结点中可用于划拨的资金（以整数为准），用户需解析后进行stringToInt的转化，可据此填写合适的资金，再次发起划拨请求
session_id	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。

5.4.2.12 OnInsertAlgoOrder()

```
virtual void OnInsertAlgoOrder (
            XTPStrategyInfoStruct * strategy_info,
            XTPRI * error_info,
            uint64_t session_id )  [inline], [virtual]
```

algo业务中报送策略单的响应

参数

strategy_info	用户报送的策略单的具体信息
error_info	报送策略单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
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 DemoTestTraderSpi 重载.

5.4.2.13 OnOptionCombinedOrderEvent()

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

期权组合策略报单通知

参数

order_info	订单响应具体信息，用户可以通过order_info.order_xtp_id来管理订单，通过GetClientIDByXTPID() == client_id来过滤自己的订单，order_info.qty_left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。order_info.order_cancel_xtp_id为其所对应的撤单ID，不为0时表示此单被撤成功
error_info	订单被拒绝或者发生错误时错误代码和错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
session_id	资金账户对应的session_id，登录时得到

备注

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

5.4.2.14 OnOptionCombinedTradeEvent()

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

期权组合策略成交通知

参数

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

备注

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

5.4.2.15 OnOrderEvent()

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

报单通知

参数

order_info	订单响应具体信息，用户可以通过order_info.order_xtp_id来管理订单，通过GetClientIDByXTPID() == client_id来过滤自己的订单，order_info.qty_left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。order_info.order_cancel_xtp_id为其所对应的撤单ID，不为0时表示此单被撤成功
error_info	订单被拒绝或者发生错误时错误代码和错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
session_id	资金账户对应的session_id，登录时得到

备注

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

5.4.2.16 OnQueryAccountTradeMarket()

```
virtual void OnQueryAccountTradeMarket (
            int trade_location,
            XTPRI * error_info,
            int request_id,
            uint64_t session_id )  [inline], [virtual]
```

请求查询用户在本节点上可交易市场的响应

参数

trade_location	查询到的交易市场信息，按位来看，从低位开始数，第0位表示沪市，即如果(trade_location&0x01) == 0x01，代表可交易沪市，第1位表示深市，即如果(trade_location&0x02) == 0x02，表示可交易深市，如果第0位和第1位均是1，即(trade_location&(0x01|0x02)) == 0x03，就表示可交易沪深2个市场
error_info	查询可交易市场发生错误时，返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
request_id	此消息响应函数对应的请求ID
session_id	资金账户对应的session_id，登录时得到

备注

此查询只会有一个结果

5.4.2.17 OnQueryAsset()

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

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

参数

asset	查询到的资金账户情况
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.18 OnQueryCreditAssetDebtInfo()

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

请求查询信用账户待还资金的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.19 OnQueryCreditCashRepayInfo()

```
virtual void OnQueryCreditCashRepayInfo (
            XTPCrdCashRepayInfo * cash_repay_info,
            XTPRI * error_info,
            int request_id,
            bool is_last,
            uint64_t session_id )  [inline], [virtual]
```

请求查询融资融券业务中的现金直接还款报单的响应

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.20 OnQueryCreditDebtInfo()

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

请求查询信用账户负债信息的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.21 OnQueryCreditExcessStock()

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

融资融券业务中请求查询指定余券信息的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

stock_info	查询到的余券信息
error_info	查询信用账户余券信息发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误

参数

request_id	此消息响应函数对应的请求ID
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.22 OnQueryCreditExtendDebtDateOrders()

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

查询融资融券业务中负债合约展期订单响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

debt_extend_info	查询到的负债合约展期情况
error_info	查询负债合约展期发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误。当error_info.error_id=11000350时，表明没有记录，当为其他非0值时，表明合约发生拒单时的错误原因
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.23 OnQueryCreditFundExtraInfo()

```
virtual void OnQueryCreditFundExtraInfo (
            XTPCrdFundExtraInfo * fund_info,
            XTPRI * error_info,
```

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

查询融资融券业务中信用账户附加信息的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

fund_info	信用账户附加信息
error_info	查询信用账户附加信息发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
request_id	此消息响应函数对应的请求ID
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.24 OnQueryCreditFundInfo()

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

请求查询信用账户额外信息的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

fund_info	查询到的信用账户额外信息情况
error_info	查询信用账户额外信息发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
request_id	此消息响应函数对应的请求ID
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.25 OnQueryCreditPositionExtraInfo()

```
virtual void OnQueryCreditPositionExtraInfo (
            XTPCrdPositionExtraInfo * fund_info,
            XTPRI * error_info,
```

```
            int request_id,
            bool is_last,
            uint64_t session_id )  [inline], [virtual]
```

查询融资融券业务中信用账户指定证券的附加信息的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.26 OnQueryCreditTickerAssignInfo()

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

请求查询信用账户可融券头寸信息的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.27 OnQueryCreditTickerDebtInfo()

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

请求查询信用账户指定证券负债未还信息响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.28 OnQueryETF()

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

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

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.29 OnQueryETFBasket()

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

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

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.30 OnQueryFundTransfer()

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

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

参数

fund_transfer_info	查询到的资金账户情况
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.31 OnQueryIPOInfoList()

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

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

参数

ipo_info	查询到的今日新股申购的一只股票信息
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.32 OnQueryIPOQuotaInfo()

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

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

参数

quota_info	查询到的用户某个市场的今日新股申购额度信息
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.33 OnQueryMulCreditExcessStock()

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

融资融券业务中请求查询余券信息的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.34 OnQueryOptionAuctionInfo()

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

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

参数

option_info	查询到的期权合约情况
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.35 OnQueryOptionCombinedExecPosition()

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

查询期权行权合并头寸的响应

参数

position_info	查询到的一个行权合并头寸信息
error_info	查询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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

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

5.4.2.36 OnQueryOptionCombinedOrders()

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

请求查询期权组合策略报单响应-旧版本接口

参数

order_info	查询到的一个报单
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

由于支持分时段查询，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。此对应的请求函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.4.2.37 OnQueryOptionCombinedOrdersByPage()

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

分页请求查询期权组合策略报单响应-旧版本接口

参数

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

备注

当order_sequence为0，表明当次查询没有查到任何记录，当is_last为true时，如果order_sequence等于req_count，那么表示还有报单，可以进行下一次分页查询，如果不等，表示所有报单已经查询完毕。一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。

5.4.2.38 OnQueryOptionCombinedOrdersByPageEx()

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

分页请求查询期权组合策略报单响应-新版本接口

参数

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

备注

当order_sequence为0，表明当次查询没有查到任何记录，当is_last为true时，如果order_sequence等于req_count，那么表示还有报单，可以进行下一次分页查询，如果不等，表示所有报单已经查询完毕。一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。

5.4.2.39 OnQueryOptionCombinedOrdersEx()

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

请求查询期权组合策略报单响应-新版本接口

参数

order_info	查询到的一个报单
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

由于支持分时段查询，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。此对应的请求函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.4.2.40 OnQueryOptionCombinedPosition()

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

请求查询期权组合策略持仓响应

参数

position_info	查询到的一个持仓信息
error_info	查询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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

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

5.4.2.41 OnQueryOptionCombinedStrategyInfo()

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

请求查询期权组合策略信息响应

参数

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

备注

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

5.4.2.42 OnQueryOptionCombinedTrades()

```
virtual void OnQueryOptionCombinedTrades (
            XTPQueryOptCombTradeRsp * trade_info,
```

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

请求查询期权组合策略成交响应

参数

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

备注

由于支持分时段查询，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。此对应的请求函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.4.2.43 OnQueryOptionCombinedTradesByPage()

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

分页请求查询期权组合策略成交响应

参数

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

备注

当trade_sequence为0，表明当次查询没有查到任何记录，当is_last为true时，如果trade_sequence等于req_count，那么表示还有回报，可以进行下一次分页查询，如果不等，表示所有回报已经查询完毕。一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。

5.4.2.44 OnQueryOrder()

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

请求查询报单响应-旧版本接口

参数

order_info	查询到的一个报单
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

由于支持分时段查询，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。此对应的请求函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.4.2.45 OnQueryOrderByPage()

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

分页请求查询报单响应-旧版本接口

参数

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

备注

当order_sequence为0，表明当次查询没有查到任何记录，当is_last为true时，如果order_sequence等于req_count，那么表示还有报单，可以进行下一次分页查询，如果不等，表示所有报单已经查询完毕。一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。

5.4.2.46 OnQueryOrderByPageEx()

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

分页请求查询报单响应-新版本接口

参数

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

备注

当order_sequence为0，表明当次查询没有查到任何记录，当is_last为true时，如果order_sequence等于req_count，那么表示还有报单，可以进行下一次分页查询，如果不等，表示所有报单已经查询完毕。一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。

5.4.2.47 OnQueryOrderEx()

```
virtual void OnQueryOrderEx (
            XTPOrderInfoEx * 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-新版本接口

参数

order_info	查询到的一个报单信息
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

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

5.4.2.48 OnQueryOtherServerFund()

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

请求查询其他节点可用资金的响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

参数

fund_info	查询到的其他节点可用资金情况
error_info	查询其他节点可用资金发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
request_id	此消息响应函数对应的请求ID
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.49 OnQueryPosition()

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

请求查询投资者持仓响应

参数

position	查询到的一只股票的持仓情况
error_info	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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

由于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

5.4.2.50 OnQueryStrategy()

```
virtual void OnQueryStrategy (
            XTPStrategyInfoStruct * strategy_info,
```

```
            char * strategy_param,
            XTPRI * error_info,
            int32_t request_id,
            bool is_last,
            uint64_t session_id )  [inline], [virtual]
```

algo业务中查询策略列表的响应

参数

strategy_info	策略具体信息
strategy_param	此策略中包含的参数，如果error_info.error_id为0时，有意义
error_info	查询查询策略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
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 DemoTestTraderSpi 重载.

5.4.2.51 OnQueryStructuredFund()

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

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

参数

fund_info	查询到的分级基金情况
error_info	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为0时，表明没有错误
request_id	此消息响应函数对应的请求ID
is_last	此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响应
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.52 OnQueryTrade()

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

请求查询成交响应

参数

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

备注

由于支持分时段查询，一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。此对应的请求函数不建议轮询使用，当报单量过多时，容易造成用户线路拥堵，导致api断线

5.4.2.53 OnQueryTradeByPage()

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

分页请求查询成交响应

参数

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

备注

当trade_sequence为0，表明当次查询没有查到任何记录，当is_last为true时，如果trade_sequence等于req_count，那么表示还有回报，可以进行下一次分页查询，如果不等，表示所有回报已经查询完毕。一个查询请求可能对应多个响应，需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线。

5.4.2.54 OnStrategyStateReport()

```
virtual void OnStrategyStateReport (
            XTPStrategyStateReportStruct * strategy_state,
            uint64_t session_id )  [inline], [virtual]
```

algo业务中策略运行时策略状态通知

参数

strategy_state	用户策略运行情况的状态通知
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 DemoTestTraderSpi 重载.

5.4.2.55 OnStrategySymbolStateReport()

```
virtual void OnStrategySymbolStateReport (
            XTPStrategySymbolStateReport * strategy_symbol_state,
            uint64_t session_id )  [inline], [virtual]
```

algo业务中策略运行时策略指定证券执行状态通知

参数

strategy_symbol_state	用户策略指定证券运行情况的状态通知
session_id	资金账户对应的session_id，登录时得到

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

5.4.2.56 OnTradeEvent()

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

成交通知

参数

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

备注

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

该类的文档由以下文件生成:

- xtp_trader_api.h

5.5 XTPClientQueryCrdDebtStockReq结构体 参考

融资融券指定证券上的负债未还数量请求结构体

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

成员变量

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

5.5.1 详细描述

融资融券指定证券上的负债未还数量请求结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.6 XTPClientQueryCrdPositionStkInfo结构体 参考

融券头寸证券信息

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

成员变量

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

5.6.1 详细描述

融券头寸证券信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.7 XTPClientQueryCrdPositionStockReq结构体 参考

融券头寸证券查询请求结构体

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

成员变量

- XTP_MARKET_TYPE market
  证券市场
- char ticker [XTP_TICKER_LEN]
  证券代码

5.7.1 详细描述

融券头寸证券查询请求结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.8 XTPClientQueryCrdSurplusStkReqInfo结构体 参考

信用业务余券查询请求结构体

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

成员变量

- XTP_MARKET_TYPE market
  证券市场
- char ticker [XTP_TICKER_LEN]
  证券代码

5.8.1 详细描述

信用业务余券查询请求结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.9 XTPClientQueryCrdSurplusStkRspInfo结构体 参考

信用业务余券信息

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

成员变量

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

5.9.1 详细描述

信用业务余券信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.10 XTPCombLegStrategy结构体 参考

期权组合策略的成分合约信息

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

成员变量

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

5.10.1 详细描述

期权组合策略的成分合约信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.11 XTPCrdCashRepayDebtInterestFeeRsp结构体 参考

融资融券现金还息费响应信息

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

成员变量

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

5.11.1 详细描述

融资融券现金还息费响应信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.12 XTPCrdCashRepayInfo结构体 参考

单条融资融券直接还款记录信息

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

成员变量

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

5.12.1 详细描述

单条融资融券直接还款记录信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.13 XTPCrdCashRepayRsp结构体 参考

融资融券直接还款响应信息

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

成员变量

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

5.13.1 详细描述

融资融券直接还款响应信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.14 XTPCrdDebtInfo结构体 参考

单条融资融券负债记录信息

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

成员变量

- int32_t debt_type
  负债合约类型：*0*为融资，*1*为融券，*2*未知
- char debt_id [33]
  负债合约编号
- int64_t position_id
  负债对应两融头寸编号
- uint64_t order_xtp_id
  生成负债的订单编号，非当日负债无此项
- int32_t debt_status
  负债合约状态：*0*为未偿还或部分偿还，*1*为已偿还，*2*为过期未平仓，*3*未知
- XTP_MARKET_TYPE market
  市场
- char ticker [XTP_TICKER_LEN]
  证券代码
- uint64_t order_date
  委托日期
- uint64_t end_date
  负债截止日期
- uint64_t orig_end_date
  负债原始截止日期
- bool is_extended
  当日是否接收到展期请求：*false*为没收到，*true*为收到
- double remain_amt
  未偿还金额
- int64_t remain_qty
  未偿还融券数量
- double remain_principal
  未偿还本金金额
- int64_t due_right_qty
  应偿还权益数量
- int64_t unknown [2]
  保留字段

5.14.1 详细描述

单条融资融券负债记录信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.15 XTPCrdDebtStockInfo结构体 参考

融资融券指定证券的融券负债相关信息

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

成员变量

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

5.15.1 详细描述

融资融券指定证券的融券负债相关信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.16 XTPCrdFundExtraInfo结构体 参考

融资融券帐户附加信息

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

成员变量

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

5.16.1 详细描述

融资融券帐户附加信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.17 XTPCrdFundInfo结构体 参考

融资融券特有帐户数据

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

成员变量

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

5.17.1 详细描述

融资融券特有帐户数据

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.18 XTPCrdPositionExtraInfo结构体 参考

融资融券帐户持仓附加信息

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

成员变量

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

5.18.1 详细描述

融资融券帐户持仓附加信息

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.19 XTPCreditDebtExtendNotice结构体 参考

用户展期请求的通知

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

成员变量

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

5.19.1 详细描述

用户展期请求的通知

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.20 XTPCreditDebtExtendReq结构体 参考

用户展期请求

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

成员变量

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

5.20.1 详细描述

用户展期请求

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.21 XTPFundQueryReq结构体 参考

用户资金查询请求结构体

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

成员变量

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

5.21.1 详细描述

用户资金查询请求结构体

该结构体的文档由以下文件生成:

- xoms_api_fund_struct.h

5.22 XTPFundQueryRsp结构体 参考

用户资金查询响应结构体

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

成员变量

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

5.22.1 详细描述

用户资金查询响应结构体

该结构体的文档由以下文件生成:

- xoms_api_fund_struct.h

5.23 XTPFundTransferNotice结构体 参考

资金内转流水通知

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

成员变量

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

5.23.1 详细描述

资金内转流水通知

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.24 XTPFundTransferReq结构体 参考

用户资金请求

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

成员变量

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

5.24.1 详细描述

用户资金请求

该结构体的文档由以下文件生成:

- xoms_api_fund_struct.h

5.25 XTPMarketDataBondExData结构体 参考

债券额外数据

```
#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 match_lastpx
  匹配成交最近价*(SZ)*
- double ma_bond_price
  债券加权平均价格*(SH)*
- double r2
  预留
- double r3
  预留
- double r4
  预留
- double r5
  预留
- double r6
  预留
- double r7
  预留
- double r8
  预留
- 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)*
- char instrument_status [8]
  时段*(SHL2)*，*L1*快照数据没有此字段，具体字段说明参阅《上海新债券*Level2*行情说明.*doc*》文档

5.25.1 详细描述

债券额外数据

该结构体的文档由以下文件生成:

- xquote_api_struct.h

5.26 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)*

5.26.1 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

5.27 XTPMarketDataStockExData结构体 参考

股票、基金 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

5.27.1 详细描述

股票、基金 等额外数据

该结构体的文档由以下文件生成:

- xquote_api_struct.h

5.28 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（*SH*）
- double settl_price
  今日结算价（*SH*）
- 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
  预留*(无意义)*
- double bid [10]
  十档申买价
- double ask [10]
  十档申卖价
- int64_t bid_qty [10]
  十档申买量
- int64_t ask_qty [10]
  十档申卖量
- int64_t trades_count
  成交笔数
- char ticker_status [8]
  当前交易状态说明，参阅《*XTP API*常见问题*.doc*》文档
-

```
union {
  XTPMarketDataStockExData stk
  XTPMarketDataOptionExData opt
  XTPMarketDataBondExData bond
};
```

数据

- XTP_MARKETDATA_TYPE data_type
  决定了*union*是哪种数据类型 *(2.2.32*版本以前所用字段，仅为了保持兼容，不建议使用该字段*)*
- XTP_MARKETDATA_TYPE_V2 data_type_v2
  决定了*union*是哪种数据类型（*2.2.32*版本新增字段，更详细区分了行情快照数据类型）

5.28.1 详细描述

行情

该结构体的文档由以下文件生成:

- xquote_api_struct.h

5.29 XTPOptCombLegInfo结构体 参考

组合策略腿合约信息结构体

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

成员变量

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

5.29.1 详细描述

组合策略腿合约信息结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.30 XTPOptCombOrderInfo结构体 参考

期权组合策略报单响应结构体

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

成员变量

- uint64_t order_xtp_id
  *XTP*系统订单*ID*，在*XTP*系统中唯一
- uint32_t order_client_id
  报单引用，用户自定义
- uint32_t order_cancel_client_id
  报单操作引用，用户自定义（暂未使用）
- uint64_t order_cancel_xtp_id
  撤单在*XTP*系统中的*id*，在*XTP*系统中唯一
- XTP_MARKET_TYPE market
- int64_t quantity
  数量，此订单的报单数量
- XTP_SIDE_TYPE side
  组合方向
- XTP_BUSINESS_TYPE business_type
  业务类型
- int64_t qty_traded
  今成交数量，为此订单累计成交数量
- int64_t qty_left
  剩余数量，当撤单成功时，表示撤单数量
- int64_t insert_time
  委托时间，格式为*YYYYMMDDHHMMSSsss*
- int64_t update_time
  最后修改时间，格式为*YYYYMMDDHHMMSSsss*
- int64_t cancel_time
  撤销时间，格式为*YYYYMMDDHHMMSSsss*
- double trade_amount
  成交金额，组合拆分涉及的保证金*(*保留字段*)*
- char order_local_id [XTP_LOCAL_ORDER_LEN]
  本地报单编号 *OMS*生成的单号，不等同于*order_xtp_id*，为服务器传到报盘的单号
- XTP_ORDER_STATUS_TYPE order_status
  报单状态，订单响应中没有部分成交状态的推送，在查询订单结果中，会有部分成交状态
- XTP_ORDER_SUBMIT_STATUS_TYPE order_submit_status
  报单提交状态，用户可用此字段来区分撤单和报单
- TXTPOrderTypeType order_type
  报单类型
- XTPOptCombPlugin opt_comb_info
  期权组合策略信息

5.30.1 详细描述

期权组合策略报单响应结构体

5.30.2 结构体成员变量说明

5.30.2.1 market

```
XTP_MARKET_TYPE market
```

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

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.31 XTPOptCombOrderInfoEx结构体 参考

期权组合策略报单响应结构体，新版本

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

成员变量

- uint64_t order_xtp_id
  *XTP*系统订单*ID*，在*XTP*系统中唯一
- uint32_t order_client_id
  报单引用，用户自定义
- uint32_t order_cancel_client_id
  报单操作引用，用户自定义（暂未使用）
- uint64_t order_cancel_xtp_id
  撤单在*XTP*系统中的*id*，在*XTP*系统中唯一
- XTP_MARKET_TYPE market
- int64_t quantity
  数量，此订单的报单数量
- XTP_SIDE_TYPE side
  组合方向
- XTP_BUSINESS_TYPE business_type
  业务类型
- int64_t qty_traded
  今成交数量，为此订单累计成交数量
- int64_t qty_left
  剩余数量，当撤单成功时，表示撤单数量
- int64_t insert_time
  委托时间，格式为*YYYYMMDDHHMMSSsss*
- int64_t update_time
  最后修改时间，格式为*YYYYMMDDHHMMSSsss*
- int64_t cancel_time
  撤销时间，格式为*YYYYMMDDHHMMSSsss*
- double trade_amount
  成交金额，组合拆分涉及的保证金
- char order_local_id [XTP_LOCAL_ORDER_LEN]
  本地报单编号 *OMS*生成的单号，不等同于*order_xtp_id*，为服务器传到报盘的单号
- XTP_ORDER_STATUS_TYPE order_status
  报单状态，订单响应中没有部分成交状态的推送，在查询订单结果中，会有部分成交状态

- XTP_ORDER_SUBMIT_STATUS_TYPE order_submit_status
  报单提交状态，*OMS*内部使用，用户无需关心
- TXTPOrderTypeType order_type
  报单类型
- XTPOptCombPlugin opt_comb_info
  期权组合策略信息
- char order_exch_id [XTP_ORDER_EXCH_LEN]
  报单编号 –交易所单号，上交所为空，深交所有此字段
- XTPRI order_err_t
  订单的错误信息
- uint64_t unknown [2]
  保留字段

5.31.1 详细描述

期权组合策略报单响应结构体，新版本

5.31.2 结构体成员变量说明

5.31.2.1 market

```
XTP_MARKET_TYPE market
```

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

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.32 XTPOptCombOrderInsertInfo结构体 参考

期权组合策略新订单请求

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

成员变量

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

5.32.1 详细描述

期权组合策略新订单请求

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.33 XTPOptCombPlugin结构体 参考

期权组合策略报单附加信息结构体

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

成员变量

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

5.33.1 详细描述

期权组合策略报单附加信息结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.34 XTPOptCombTradeReport结构体 参考

期权组合策略报单成交结构体

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

成员变量

- uint64_t order_xtp_id

  *XTP*系统订单*ID*，此成交回报相关的订单*ID*，在*XTP*系统中唯一
- uint32_t order_client_id

  报单引用
- XTP_MARKET_TYPE market

  交易市场
- uint64_t local_order_id

  订单号，引入*XTPID*后，该字段实际和*order_xtp_id*重复。接口中暂时保留。
- char exec_id [XTP_EXEC_ID_LEN]

  成交编号，深交所唯一，上交所每笔交易唯一，当发现*2*笔成交回报拥有相同的*exec_id*，则可以认为此笔交易自成交
- int64_t quantity

  数量，此次成交的数量，不是累计数量
- int64_t trade_time

  成交时间，格式为*YYYYMMDDHHMMSSsss*
- double trade_amount

  成交金额，组合拆分涉及的保证金
- uint64_t report_index

  成交序号 –回报记录号，每个交易所唯一,*report_index+market*字段可以组成唯一标识表示成交回报
- char order_exch_id [XTP_ORDER_EXCH_LEN]

  报单编号 –交易所单号*(*保留字段*)*
- TXTPTradeTypeType trade_type

  成交类型 –成交回报中的执行类型
- XTP_SIDE_TYPE side

  组合方向
- XTP_BUSINESS_TYPE business_type

  业务类型
- char branch_pbu [XTP_BRANCH_PBU_LEN]

  交易所交易员代码
- XTPOptCombPlugin opt_comb_info

  期权组合策略信息

5.34.1 详细描述

期权组合策略报单成交结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.35 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*

5.35.1 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

5.36 XTPOrderInfo结构体 参考

报单响应结构体

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

成员变量

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

union {
uint32_t u32
*32*位字段，用来兼容老版本*api*，用户无需关心
struct {
XTP_SIDE_TYPE side
买卖方向
XTP_POSITION_EFFECT_TYPE position_effect
开平标志，期权用户关注字段，其余用户填*0*即可
uint8_t reserved1
预留字段*1*
uint8_t reserved2
预留字段*2*
}
};

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

5.36.1 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

5.37 XTPOrderInfoEx结构体 参考

报单响应结构体，新版本

```
#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
- TXTPOrderTypeType order_type
  报单类型
- char order_exch_id [XTP_ORDER_EXCH_LEN]
  报单编号 –交易所单号，上交所为空，深交所有此字段
- XTPRI order_err_t
  订单的错误信息
- uint64_t unknown [2]
  保留字段

5.37.1 详细描述

报单响应结构体，新版本

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.38 XTPOrderInsertInfo结构体 参考

新订单请求

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

成员变量

- uint64_t order_xtp_id
  *XTP*系统订单*ID*，无需用户填写，在*XTP*系统中唯一
- uint32_t order_client_id
  报单引用，由客户自定义
- char ticker [XTP_TICKER_LEN]
  合约代码 客户端请求不带空格，以*'\0'*结尾
- XTP_MARKET_TYPE market
  交易市场
- double price
  价格
- double stop_price
  止损价（保留字段）
- int64_t quantity
  数量*(*股票单位为股，逆回购单位为张*)*
- XTP_PRICE_TYPE price_type
  报单价格
-

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

- XTP_BUSINESS_TYPE business_type
  业务类型

5.38.1 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

5.39 XTPQueryAssetRsp结构体 参考

账户资金查询响应结构体

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

成员变量

- double total_asset
  总资产（现货账户/期权账户参考公式：总资产 = 可用资金 + 证券资产（目前为*0*）+ 预扣的资金），（信用账户参考公式：总资产 = 可用资金 + 融券卖出所得资金余额 + 证券资产 + 预扣的资金）
- double buying_power
  可用资金
- double security_asset
  证券资产（保留字段，目前为*0*）
- double fund_buy_amount
  累计买入成交证券占用资金（仅限现货账户/期权账户，信用账户暂不可用）
- double fund_buy_fee
  累计买入成交交易费用（仅限现货账户/期权账户，信用账户暂不可用）
- double fund_sell_amount
  累计卖出成交证券所得资金（仅限现货账户/期权账户，信用账户暂不可用）
- double fund_sell_fee
  累计卖出成交交易费用（仅限现货账户/期权账户，信用账户暂不可用）

- double withholding_amount
  *XTP*系统预扣的资金（包括买卖股票时预扣的交易资金+预扣手续费）
- XTP_ACCOUNT_TYPE account_type
  账户类型
- double frozen_margin
  冻结的保证金（仅限期权账户）
- double frozen_exec_cash
  行权冻结资金（仅限期权账户）
- double frozen_exec_fee
  行权费用（仅限期权账户）
- double pay_later
  垫付资金（仅限期权账户）
- double preadva_pay
  预垫付资金（仅限期权账户）
- double orig_banlance
  昨日余额（仅限期权账户）
- double banlance
  当前余额（仅限期权账户）
- double deposit_withdraw
  当天出入金（仅限期权账户）
- double trade_netting
  当日交易资金轧差（仅限期权账户）
- double captial_asset
  资金资产（仅限期权账户）
- double force_freeze_amount
  强锁资金（仅限期权账户）
- double preferred_amount
  可取资金（仅限期权账户）
- double repay_stock_aval_banlance
  融券卖出所得资金余额（仅限信用账户，只能用于买券还券）
- double fund_order_data_charges
  累计订单流量费
- double fund_cancel_data_charges
  累计撤单流量费
- double exchange_cur_risk_degree
  交易所实时风险度（仅限期权账户,后续服务器版本支持，目前为*0*）
- double company_cur_risk_degree
  公司实时风险度（仅限期权账户,后续服务器版本支持，目前为*0*）
- uint64_t unknown [43 - 12 - 1 - 2 - 2]
  *(*保留字段*)*

5.39.1 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

5.40 XTPQueryCombineStrategyInfoRsp结构体 参考

查询期权组合策略信息的响应

`#include <xoms_api_struct.h>`

成员变量

- char strategy_id [XTP_STRATEGY_ID_LEN]
  组合策略代码，*CNSJC*、*PXSJC*、*PNSJC*、*CXSJC*、*KS*、*KKS*
- char strategy_name [XTP_STRATEGY_NAME_LEN]
  组合策略名称，认购牛市价差策略、认沽熊市价差策略、认沽牛市价差策略、认购熊市价差策略、跨式空头、宽跨式空头
- XTP_MARKET_TYPE market
  交易市场
- int32_t leg_num
  成分合约个数，*1-4*个，即下面数组的实际大小
- XTPCombLegStrategy leg_strategy [XTP_STRATEGE_LEG_NUM]
  成分合约信息，最多四条腿
- XTP_EXPIRE_DATE_TYPE expire_date_type
  到期日要求。枚举值为：同到期日，不同到期日，无到期日要求
- XTP_UNDERLYING_TYPE underlying_type
  标的要求。枚举值为：相同标的，不同标的，无标的要求
- XTP_AUTO_SPLIT_TYPE auto_sep_type
  自动解除类型。枚举值为：*-1*：不适用；*0*：到期日自动解除；*1*：*E-1*日自动解除，依次类推
- uint64_t reserved [10]
  预留的字段

5.40.1 详细描述

查询期权组合策略信息的响应

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.41 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

5.41.1 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

5.42 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*日预估金额差额
- double cash_component
  *T-X*日现金差额
- double net_value
  基金单位净值
- double total_amount
  最小申赎单位净值总金额=*net_value*∗*unit*

5.42.1 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

5.43 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

5.43.1 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

5.44 XTPQueryETFComponentRsp结构体 参考

查询股票ETF成分股信息–响应结构体

`#include <xoms_api_struct.h>`

成员变量

- XTP_MARKET_TYPE market
  交易市场
- char ticker [XTP_TICKER_LEN]
  *ETF*代码
- char component_ticker [XTP_TICKER_LEN]
  成份股代码
- char component_name [XTP_TICKER_NAME_LEN]
  成份股名称
- int64_t quantity
  成份股数量
- XTP_MARKET_TYPE component_market
  成份股交易市场
- ETF_REPLACE_TYPE replace_type
  成份股替代标识
- double premium_ratio
  溢价比例
- double amount
  成分股替代标识为必须现金替代时候的总金额
- double creation_premium_ratio
  申购溢价比例
- double redemption_discount_ratio
  赎回溢价比例
- double creation_amount
  申购时，成分股替代标识为必须现金替代时候的总金额
- double redemption_amount
  赎回时，成分股替代标识为必须现金替代时候的总金额

5.44.1 详细描述

查询股票ETF成分股信息–响应结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.45 XTPQueryETFComponentRspV1结构体 参考

查询股票ETF成分股信息–响应结构体，旧版本。

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

成员变量

- XTP_MARKET_TYPE market
  交易市场
- char ticker [XTP_TICKER_LEN]
  *ETF*代码
- char component_ticker [XTP_TICKER_LEN]
  成份股代码
- char component_name [XTP_TICKER_NAME_LEN]
  成份股名称
- int64_t quantity
  成份股数量
- XTP_MARKET_TYPE component_market
  成份股交易市场
- ETF_REPLACE_TYPE replace_type
  成份股替代标识
- double premium_ratio
  溢价比例
- double amount
  成分股替代标识为必须现金替代时候的总金额

5.45.1 详细描述

查询股票ETF成分股信息–响应结构体，旧版本。

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.46 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

5.46.1 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

5.47 XTPQueryIPOQuotaRsp结构体 参考

查询用户申购额度-包含创业板额度

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

成员变量

- XTP_MARKET_TYPE market
  交易市场
- int32_t quantity
  可申购额度
- int32_t tech_quantity
  上海科创板额度
- int32_t unused
  保留

5.47.1 详细描述

查询用户申购额度-包含创业板额度

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.48 XTPQueryIPOQuotaRspV1结构体 参考

查询用户申购额度-旧版

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

成员变量

- XTP_MARKET_TYPE market
  交易市场
- int32_t quantity
  可申购额度

5.48.1 详细描述

查询用户申购额度-旧版

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.49 XTPQueryIPOTickerRsp结构体 参考

查询当日可申购新股信息

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

成员变量

- XTP_MARKET_TYPE market
  交易市场
- char ticker [XTP_TICKER_LEN]
  申购代码
- char ticker_name [XTP_TICKER_NAME_LEN]
  申购股票名称
- XTP_TICKER_TYPE ticker_type
  证券类别
- double price
  申购价格
- int32_t unit
  申购单元
- int32_t qty_upper_limit
  最大允许申购数量

5.49.1 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

5.50 XTPQueryOptCombExecPosReq结构体 参考

查询期权行权合并头寸请求结构体

`#include <xoms_api_struct.h>`

成员变量

- XTP_MARKET_TYPE market
  市场
- char cntrt_code_1 [XTP_TICKER_LEN]
  成分合约*1*代码
- char cntrt_code_2 [XTP_TICKER_LEN]
  成分合约*2*代码

5.50.1 详细描述

查询期权行权合并头寸请求结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.51 XTPQueryOptCombExecPosRsp结构体 参考

查询期权行权合并头寸的响应

`#include <xoms_api_struct.h>`

成员变量

- XTP_MARKET_TYPE market
  市场
- char cntrt_code_1 [XTP_TICKER_LEN]
  成分合约*1*代码
- char cntrt_name_1 [XTP_TICKER_NAME_LEN]
  成分合约*1*名称
- XTP_POSITION_DIRECTION_TYPE position_side_1
  成分合约*1*持仓方向
- XTP_OPT_CALL_OR_PUT_TYPE call_or_put_1
  成分合约*1*类型
- int64_t avl_qty_1
  成分合约*1*可用持仓数量
- int64_t orig_own_qty_1
  成分合约*1*昨日持仓数量
- int64_t own_qty_1

成分合约1当前持仓数量
- char cntrt_code_2 [XTP_TICKER_LEN]
  成分合约2代码
- char cntrt_name_2 [XTP_TICKER_NAME_LEN]
  成分合约2名称
- XTP_POSITION_DIRECTION_TYPE position_side_2
  成分合约2持仓方向
- XTP_OPT_CALL_OR_PUT_TYPE call_or_put_2
  成分合约2类型
- int64_t avl_qty_2
  成分合约2可用持仓数量
- int64_t orig_own_qty_2
  成分合约2昨日持仓数量
- int64_t own_qty_2
  成分合约2当前持仓数量
- int64_t net_qty
  权利仓净头寸
- int64_t order_qty
  行权合并委托数量，不含已拒单已撤单。
- int64_t confirm_qty
  行权合并已确认数量
- int64_t avl_qty
  可行权合并数量
- uint64_t reserved [49]
  保留字段

5.51.1 详细描述

查询期权行权合并头寸的响应

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.52 XTPQueryOptCombOrderByPageReq结构体 参考

查询期权组合策略订单请求-分页查询

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

成员变量

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

5.52.1 详细描述

查询期权组合策略订单请求-分页查询

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.53 XTPQueryOptCombOrderReq结构体 参考

期权组合策略报单查询 // 期权组合策略报单查询请求-条件查询

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

成员变量

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

5.53.1 详细描述

期权组合策略报单查询 // 期权组合策略报单查询请求-条件查询

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.54 XTPQueryOptCombPositionReq结构体 参考

查询期权组合策略持仓情况请求结构体

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

成员变量

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

5.54.1 详细描述

查询期权组合策略持仓情况请求结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.55 XTPQueryOptCombPositionRsp结构体 参考

查询期权组合策略持仓信息的响应

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

成员变量

- char strategy_id [XTP_STRATEGY_ID_LEN]
  组合策略代码
- char strategy_name [XTP_STRATEGY_NAME_LEN]
  组合策略名称
- XTP_MARKET_TYPE market
  交易市场
- int64_t total_qty
  总持仓
- int64_t available_qty
  可拆分持仓
- int64_t yesterday_position
  昨日持仓
- XTPOptCombPlugin opt_comb_info
  期权组合策略信息
- uint64_t reserved [50]
  保留字段

5.55.1 详细描述

查询期权组合策略持仓信息的响应

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.56 XTPQueryOptCombReportByExecIdReq结构体 参考

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

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

成员变量

- uint64_t order_xtp_id
  *XTP*订单系统*ID*
- char exec_id [XTP_EXEC_ID_LEN]
  成交执行编号

5.56.1 详细描述

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

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.57 XTPQueryOptCombTraderByPageReq结构体 参考

查询期权组合策略成交回报请求-分页查询

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

成员变量

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

5.57.1 详细描述

查询期权组合策略成交回报请求-分页查询

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.58 XTPQueryOptCombTraderReq结构体 参考

查询期权组合策略成交回报请求-查询条件

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

成员变量

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

5.58.1 详细描述

查询期权组合策略成交回报请求-查询条件

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.59 XTPQueryOptExecInfoRsp结构体 参考

查询期权合约行权信息的响应

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

成员变量

- XTP_MARKET_TYPE market
  市场
- char cntrt_code [XTP_TICKER_LEN]
  合约代码
- int64_t own_qty_long
  权利仓数量
- int64_t own_qty_short
  义务仓数量
- int64_t own_qty_short_cover
  备兑义务仓数量
- int64_t net_qty
  净头寸
- int64_t combed_qty_long
  权利仓已组合数量
- int64_t combed_qty_short
  义务仓已组合数量
- int64_t combed_qty_short_cover
  备兑义务仓已组合数量
- int64_t total_execute_gene_order_qty
  累计普通行权委托数量
- int64_t total_execute_gene_confirm_qty
  累计普通行权确认数量
- int64_t total_execute_comb_order_qty
  累计行权合并委托数量
- int64_t total_execute_comb_confirm_qty
  累计行权合并确认数量
- uint64_t reserved [50]
  保留字段

5.59.1 详细描述

查询期权合约行权信息的响应

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.60 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

5.60.1 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

5.61 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）

5.61.1 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

5.62 XTPQueryOrderByPageReq结构体 参考

查询订单请求-分页查询

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

成员变量

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

5.62.1 详细描述

查询订单请求-分页查询

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.63 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

5.63.1 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

5.64 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

5.64.1 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

5.65 XTPQueryStkPositionReq结构体 参考

查询股票持仓情况请求结构体

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

成员变量

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

5.65.1 详细描述

查询股票持仓情况请求结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.66 XTPQueryStkPositionRsp结构体 参考

查询股票持仓情况

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

成员变量

- char ticker [XTP_TICKER_LEN]
  证券代码
- char ticker_name [XTP_TICKER_NAME_LEN]
  证券名称
- XTP_MARKET_TYPE market
  交易市场
- int64_t total_qty
  总持仓
- int64_t sellable_qty
  可卖持仓
- double avg_price
  持仓成本
- double unrealized_pnl
  浮动盈亏（保留字段）
- int64_t yesterday_position
  昨日持仓
- int64_t purchase_redeemable_qty
  今日申购赎回数量（申购和赎回数量不可能同时存在，因此可以共用一个字段）
- XTP_POSITION_DIRECTION_TYPE position_direction
  持仓方向
- XTP_POSITION_SECURITY_TYPE position_security_type
  持仓类型*(*此字段所有账户都可能用到，可以用来区分股份是否为配售*)*
- int64_t executable_option
  可行权合约
- int64_t lockable_position
  可锁定标的
- int64_t executable_underlying
  可行权标的
- int64_t locked_position
  已锁定标的
- int64_t usable_locked_position
  可用已锁定标的
- double profit_price
  盈亏成本价
- double buy_cost
  买入成本
- double profit_cost
  盈亏成本
- double market_value
  持仓市值（此字段目前只有期权账户有值，其他类型账户为*0*）
- uint64_t unknown [50 - 10]
  *(*保留字段*)*

5.66.1 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

5.67 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

5.67.1 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

5.68 XTPQueryTraderByPageReq结构体 参考

查询成交回报请求-分页查询

`#include <xoms_api_struct.h>`

成员变量

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

5.68.1 详细描述

查询成交回报请求-分页查询

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.69 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

5.69.1 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

5.70 XTPQuoteFullInfo结构体 参考

股票行情全量静态信息

`#include <xquote_api_struct.h>`

成员变量

- XTP_EXCHANGE_TYPE exchange_id
  交易所代码
- char ticker [XTP_TICKER_LEN]
  证券代码
- char ticker_name [XTP_TICKER_NAME_LEN]
  证券名称
- XTP_SECURITY_TYPE security_type
  合约详细类型
- XTP_QUALIFICATION_TYPE ticker_qualification_class
  合约适当性类别
- bool is_registration
  是否注册制*(仅适用创业板股票，创新企业股票及存托凭证)*
- bool is_VIE
  是否具有协议控制架构*(仅适用创业板股票，创新企业股票及存托凭证)*
- bool is_noprofit

是否尚未盈利*(仅适用创业板股票，创新企业股票及存托凭证)*

- bool is_weighted_voting_rights
  是否存在投票权差异*(仅适用创业板股票，创新企业股票及存托凭证)*
- bool is_have_price_limit
  是否有涨跌幅限制*(注：不提供具体幅度，可通过涨跌停价和昨收价来计算幅度)*
- double upper_limit_price
  涨停价（仅在有涨跌幅限制时有效）
- double lower_limit_price
  跌停价（仅在有涨跌幅限制时有效）
- double pre_close_price
  昨收价
- double price_tick
  价格最小变动价位
- int32_t bid_qty_upper_limit
  限价买委托数量上限
- int32_t bid_qty_lower_limit
  限价买委托数量下限
- int32_t bid_qty_unit
  限价买数量单位
- int32_t ask_qty_upper_limit
  限价卖委托数量上限
- int32_t ask_qty_lower_limit
  限价卖委托数量下限
- int32_t ask_qty_unit
  限价卖数量单位
- int32_t market_bid_qty_upper_limit
  市价买委托数量上限
- int32_t market_bid_qty_lower_limit
  市价买委托数量下限
- int32_t market_bid_qty_unit
  市价买数量单位
- int32_t market_ask_qty_upper_limit
  市价卖委托数量上限
- int32_t market_ask_qty_lower_limit
  市价卖委托数量上限
- int32_t market_ask_qty_unit
  市价卖数量单位
- XTP_SECURITY_STATUS security_status
  证券状态
- uint32_t unknown1
  保留字段
- uint64_t unknown [3]
  保留字段

5.70.1 详细描述

股票行情全量静态信息

该结构体的文档由以下文件生成:

- xquote_api_struct.h

5.71 XTPQuoteRebuildReq结构体 参考

实时行情回补查询

```
#include <xquote_api_rebuild_tbt_struct.h>
```

成员变量

- int32_t request_id

  请求*id* 请求端自行管理定义
- XTP_QUOTE_REBUILD_DATA_TYPE data_type

  请求数据类型 *1*-快照 *2*-逐笔
- XTP_EXCHANGE_TYPE exchange_id

  请求市场 *1*-上海 *2*-深圳
- char ticker [16]

  合约代码 以*'\0'*结尾 沪深*A*股*6*位 期权*8*位
- int16_t channel_number

  *data_type=*逐笔 表示逐笔通道号
- int64_t begin

  *data_type=*逐笔 表示序列号*begin*； *=*快照 表示时间*begin(*格式为*YYYYMMDDHHMMSSsss)*
- int64_t end

  *data_type=*逐笔 表示序列号*end*； *=*快照 表示时间*end(*格式为*YYYYMMDDHHMMSSsss)* 逐笔区间：*[begin, end]*前后闭区间 快照区间：*[begin, end)* 前闭后开区间

5.71.1 详细描述

实时行情回补查询

实时行情回补请求结构体

该结构体的文档由以下文件生成:

- xquote_api_rebuild_tbt_struct.h

5.72 XTPQuoteRebuildResultRsp结构体 参考

实时行情回补响应结构体

```
#include <xquote_api_rebuild_tbt_struct.h>
```

成员变量

- int32_t request_id
  请求*id* 请求包带过来的*id*
- XTP_EXCHANGE_TYPE exchange_id
  市场类型 上海 深圳
- uint32_t size
  总共返回的数据条数
- int16_t channel_number
  逐笔数据 通道号
- int64_t begin
  逐笔 表示序列号*begin;* 快照 表示时间*begin(*格式为*YYYYMMDDHHMMSSsss)*
- int64_t end
  逐笔 表示序列号*end;* 快照 表示时间*end(*格式为*YYYYMMDDHHMMSSsss)*
- XTP_REBUILD_RET_TYPE result_code
  结果类型码
- char msg [64]
  结果信息文本

5.72.1 详细描述

实时行情回补响应结构体

该结构体的文档由以下文件生成:

- xquote_api_rebuild_tbt_struct.h

5.73 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)*

5.73.1 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

5.74 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

5.74.1 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

5.75 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

5.75.1 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

5.76 XTPStrategyInfoStruct结构体 参考

策略信息结构体

```
#include <algo_api_struct.h>
```

成员变量

- uint16_t m_strategy_type
  策略类型
- XTPStrategyStateType m_strategy_state
  策略状态
- uint64_t m_client_strategy_id
  客户策略*id*
- uint64_t m_xtp_strategy_id
  *xtp*策略*id*

5.76.1 详细描述

策略信息结构体

该结构体的文档由以下文件生成:

- algo_api_struct.h

5.77 XTPStrategyStateReportStruct结构体 参考

策略状态结构体

```
#include <algo_api_struct.h>
```

成员变量

- XTPStrategyInfoStruct m_strategy_info
  策略信息
- int64_t m_strategy_qty
  策略总量
- int64_t m_strategy_ordered_qty
  策略已委托数量
- int64_t m_strategy_cancelled_qty
  策略已撤单数量
- int64_t m_strategy_execution_qty
  策略已成交数量
- int64_t m_strategy_unclosed_qty
  策略未平仓数量*(T0*卖出数量-买入数量*)*
- double m_strategy_asset
  策略总金额
- double m_strategy_ordered_asset
  策略已委托金额
- double m_strategy_execution_asset
  策略已成交金额
- double m_strategy_execution_price
  策略执行价格
- double m_strategy_market_price
  策略市场价
- double m_strategy_price_diff
  策略执行价差
- double m_strategy_asset_diff
  策略执行绩效*(T0*资金预净收入*)*
- XTPRI m_error_info
  错误信息

5.77.1 详细描述

策略状态结构体

该结构体的文档由以下文件生成:

- algo_api_struct.h

5.78 XTPStrategySymbolInfoStruct结构体 参考

策略中指定证券信息结构体

```
#include <algo_api_struct.h>
```

成员变量

- XTPStrategyInfoStruct m_strategy_info
  策略信息
- char m_ticker [XTP_TICKER_LEN]
  证券代码
- XTP_MARKET_TYPE m_market
  市场

5.78.1 详细描述

策略中指定证券信息结构体

该结构体的文档由以下文件生成:

- algo_api_struct.h

5.79 XTPStrategySymbolReqStruct结构体 参考

指定策略指定证券的请求结构体

```
#include <algo_api_struct.h>
```

成员变量

- uint64_t m_xtp_strategy_id
  *xtp*策略*id*
- char m_ticker [XTP_TICKER_LEN]
  证券代码
- XTP_MARKET_TYPE m_market
  市场

5.79.1 详细描述

指定策略指定证券的请求结构体

该结构体的文档由以下文件生成:

- algo_api_struct.h

5.80 XTPStrategySymbolStateReportStruct结构体 参考

策略中指定证券的算法执行状态结构体

```
#include <algo_api_struct.h>
```

成员变量

- XTPStrategyInfoStruct m_strategy_info
  策略信息
- char m_ticker [XTP_TICKER_LEN]
  证券代码
- XTP_MARKET_TYPE m_market
  市场
- XTP_SIDE_TYPE m_side
  买卖方向，*=0*时为*T0*单
- int64_t m_strategy_qty
  策略总量
- int64_t m_strategy_ordered_qty
  策略已委托数量
- int64_t m_strategy_cancelled_qty
  策略已撤单数量
- int64_t m_strategy_execution_qty
  策略已成交数量
- int64_t m_strategy_buy_qty
  策略已买入数量*(T0)*
- int64_t m_strategy_sell_qty
  策略已卖出数量*(T0)*
- int64_t m_strategy_unclosed_qty
  策略未平仓数量*(T0*卖出数量-买入数量*)*
- double m_strategy_asset
  策略总金额
- double m_strategy_ordered_asset
  策略已委托金额
- double m_strategy_execution_asset
  策略已成交金额
- double m_strategy_buy_asset
  策略买入金额*(T0)*
- double m_strategy_sell_asset
  策略卖出金额*(TO)*
- double m_strategy_unclosed_asset
  策略未平仓金额*(T0)*
- double m_strategy_asset_diff
  策略毛收益增强金额*(T0)*
- double m_strategy_execution_price
  策略执行价格
- double m_strategy_market_price
  策略市场价
- double m_strategy_price_diff
  策略执行价差*(T0*时为毛增强收益率*)*
- XTPRI m_error_info
  错误信息

5.80.1 详细描述

策略中指定证券的算法执行状态结构体

该结构体的文档由以下文件生成:

- algo_api_struct.h

5.81 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

5.81.1 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

5.82 XTPTickByTickEntrust结构体 参考

逐笔委托

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

成员变量

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

5.82.1 详细描述

逐笔委托

5.82.2 结构体成员变量说明

5.82.2.1 ord_type

```
char ord_type
```

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

5.82.2.2 order_no

```
int64_t order_no
```

SH: 原始订单号 SZ: 无意义

5.82.2.3 qty

```
int64_t qty
```

SH: 剩余委托数量(balance) SZ: 委托数量

5.82.2.4 seq

```
int64_t seq
```

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

5.82.2.5 side

```
char side
```

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

该结构体的文档由以下文件生成:

- xquote_api_struct.h

5.83 XTPTickByTickStatus结构体 参考

逐笔状态订单

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

成员变量

- int32_t channel_no
  频道代码
- int64_t seq
  同一*channel_no*内连续
- char flag [8]
  状态信息

5.83.1 详细描述

逐笔状态订单

该结构体的文档由以下文件生成:

- xquote_api_struct.h

5.84 XTPTickByTickStruct结构体 参考

逐笔数据信息

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

成员变量

- XTP_EXCHANGE_TYPE exchange_id
  交易所代码
- char ticker [XTP_TICKER_LEN]
  合约代码（不包含交易所信息），不带空格，以'\0'结尾
- int64_t seq
- int64_t data_time
  委托时间 *or* 成交时间
- XTP_TBT_TYPE type
  委托 *or* 成交
-

union {
XTPTickByTickEntrust **entrust**
XTPTickByTickTrade **trade**
XTPTickByTickStatus **state**
};

5.84.1 详细描述

逐笔数据信息

5.84.2 结构体成员变量说明

5.84.2.1 seq

```
int64_t seq
```

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

该结构体的文档由以下文件生成:

- xquote_api_struct.h

5.85 XTPTickByTickTrade结构体 参考

逐笔成交

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

成员变量

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

5.85.1 详细描述

逐笔成交

5.85.2 结构体成员变量说明

5.85.2.1 seq

```
int64_t seq
```

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

5.85.2.2 trade_flag

```
char trade_flag
```

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

该结构体的文档由以下文件生成:

- xquote_api_struct.h

5.86 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

5.86.1 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

5.87 XTPTradeReport结构体 参考

报单成交结构体

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

成员变量

- uint64_t order_xtp_id
  *XTP*系统订单*ID*，此成交回报相关的订单*ID*，在*XTP*系统中唯一
- uint32_t order_client_id
  报单引用
- char ticker [XTP_TICKER_LEN]
  合约代码
- XTP_MARKET_TYPE market
  交易市场
- uint64_t local_order_id
  订单号，引入*XTPID*后，该字段实际和*order_xtp_id*重复。接口中暂时保留。
- char exec_id [XTP_EXEC_ID_LEN]
  成交编号，深交所唯一，上交所每笔交易唯一，当发现*2*笔成交回报拥有相同的*exec_id*，则可以认为此笔交易自成交
- double price
  价格，此次成交的价格
- int64_t quantity
  数量，此次成交的数量，不是累计数量
- int64_t trade_time
  成交时间，格式为*YYYYMMDDHHMMSSsss*
- double trade_amount
  成交金额，此次成交的总金额 = *price*∗*quantity*
- uint64_t report_index

成交序号 –回报记录号，对于单个账户来说，深交所每个平台（不同交易品种）唯一，上交所唯一，对于多账户来说，不唯一

- char order_exch_id [XTP_ORDER_EXCH_LEN]
  报单编号 –交易所单号，上交所为空，深交所有此字段
- TXTPTradeTypeType trade_type
  成交类型 –成交回报中的执行类型
-
  union {
    uint32_t u32
      *32*位字段，用来兼容老版本*api*，用户无需关心
    struct {
      XTP_SIDE_TYPE side
        买卖方向
      XTP_POSITION_EFFECT_TYPE position_effect
        开平标志
      uint8_t reserved1
        预留字段*1*
      uint8_t reserved2
        预留字段*2*
    }
  };
- XTP_BUSINESS_TYPE business_type
  业务类型
- char branch_pbu [XTP_BRANCH_PBU_LEN]
  交易所交易员代码

5.87.1 详细描述

报单成交结构体

该结构体的文档由以下文件生成:

- xoms_api_struct.h

5.88 XTPUserTerminalInfoReq结构体 参考

申报用户的ip和mac等信息，仅限授权用户使用

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

成员变量

- char local_ip [XTP_INET_ADDRESS_STR_LEN]
  本地*IP*地址
- char mac_addr [XTP_MAC_ADDRESS_LEN]
  *MAC*地址
- char hd [XTP_HARDDISK_SN_LEN]
  硬盘序列号

- XTPTerminalType term_type
  终端类型
- char internet_ip [XTP_INET_ADDRESS_STR_LEN]
  公网*IP*地址
- int32_t internet_port
  公网端口号
- XTPVersionType client_version
  客户端版本号
- char macos_sno [XTP_MACOS_SNO_LEN]
  *MacOS*系统的序列号，仅为*MacOS*系统需要填写
- char unused [27]
  预留

5.88.1 详细描述

申报用户的ip和mac等信息，仅限授权用户使用

该结构体的文档由以下文件生成:

- xoms_api_struct.h

Chapter 6

文件说明

6.1 algo_api_struct.h 文件参考

定义业务公共数据结构

```
#include "algo_data_type.h"
#include "xtp_api_struct_common.h"
```

结构体

- struct XTPStrategyInfoStruct
  策略信息结构体
- struct XTPStrategySymbolInfoStruct
  策略中指定证券信息结构体
- struct XTPStrategyStateReportStruct
  策略状态结构体
- struct XTPStrategySymbolReqStruct
  指定策略指定证券的请求结构体
- struct XTPStrategySymbolStateReportStruct
  策略中指定证券的算法执行状态结构体

类型定义

- typedef struct XTPStrategyInfoStruct XTPStrategyInfoStruct
  策略信息结构体
- typedef struct XTPStrategySymbolInfoStruct XTPStrategySymbolInfo
  策略中指定证券信息结构体
- typedef struct XTPStrategyStateReportStruct XTPStrategyStateReport
  策略状态结构体
- typedef struct XTPStrategySymbolReqStruct XTPStrategySymbolReq
  指定策略指定证券的请求结构体
- typedef struct XTPStrategySymbolStateReportStruct XTPStrategySymbolStateReport
  策略中指定证券的算法执行状态结构体

6.1.1 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

6.2 algo_data_type.h 文件参考

定义业务公共数据结构

宏定义

- #define XTP_STRATEGY_STATE_CREATING 0
  创建中
- #define XTP_STRATEGY_STATE_CREATED 1
  已创建
- #define XTP_STRATEGY_STATE_STARTING 2
  开始执行中
- #define XTP_STRATEGY_STATE_STARTED 3
  已执行
- #define XTP_STRATEGY_STATE_STOPPING 4
  停止中
- #define XTP_STRATEGY_STATE_STOPPED 5
  已停止
- #define XTP_STRATEGY_STATE_DESTROYING 6
  销毁中
- #define XTP_STRATEGY_STATE_DESTROYED 7
  已销毁
- #define XTP_STRATEGY_STATE_ERROR 8
  发生错误

类型定义

- typedef uint8_t XTPStrategyStateType
  *XTPStrategyStateType*策略状态类型

6.2.1 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

6.3 demo_test_trade_api.cpp 文件参考

定义控制台测试应用程序的入口点

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

函数

- int main ()
  //API的全局变量指针 XTP::API::TraderApi user_api_pointer;*

变量

- XTP::API::TraderApi * user_api_pointer
  API的全局变量指针

6.3.1 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

6.3.2 函数说明

6.3.2.1 main()

```
int main ( )
```

//API的全局变量指针 XTP::API::TraderApi* user_api_pointer;

测试Demo入口函数
int main()
{
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
user_api_pointer = XTP::API::TraderApi::CreateTraderApi(client_id, filepath, XTP_LOG_LEVEL_DEBUG); // 创

```
建UserApi
user_api_pointer->SubscribePublicTopic(XTP_TERT_QUICK);//设定公共流传输方式
user_api_pointer->SetSoftwareKey("xxxxxxxxxxxxxxxxxxxx");//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); // 注册响应事件类
uint64_t temp_session_ = user_api_pointer->Login(server_ip_oms.c_str(), server_port_oms, account_name_↩
oms.c_str(), account_pw_oms.c_str(), XTP_PROTOCOL_TCP);//登陆交易服务器
if (temp_session_ != 0)
{
//登录AlgoBus算法服务器
int login_ret = user_api_pointer->LoginALGO(server_ip_algo.c_str(), server_port_algo, account_name_algo.c_↩
str(), account_pw_algo.c_str(), XTP_PROTOCOL_TCP);
if (login_ret != 0)
{
//登录算法服务器失败
std::cout << account_name_algo << " login to AlgoBus error!!!!!!!!!!" << std::endl;
XTPRI* error_info = user_api_pointer->GetApiLastError(); //登录算法服务器失败时的错误原因代码
}
else
{
std::cout << account_name_algo << " login to AlgoBus success." << std::endl;
///在用户成功登录算法服务器后，算法用户建立算法通道
std::cout << account_name_oms << " begin to establish channel." << std::endl;
int user_ret = user_api_pointer->ALGOUserEstablishChannel(server_ip_oms.c_str(), server_port_oms, account↩
_name_oms.c_str(), account_pw_oms.c_str(), temp_session_);
if (user_ret != 0)
{
std::cout << account_name_oms << " establish channel send error!!!!!!!!!!" << std::endl;
XTPRI* error_info = user_api_pointer->GetApiLastError(); //建立算法通道消息发送失败时的错误原因代码
}
else
{
std::cout << account_name_oms << " establish channel send success." << std::endl;
}
}
}
else
{
//登录交易服务器失败
XTPRI* error_info = user_api_pointer->GetApiLastError();
std::cout << "Login to server error, " << error_info->error_id << " : " << error_info->error_msg << std::endl;
}
//保持主线程不退出
while (true)
{
#ifdef _WIN32
Sleep(1000);
#else
sleep(1);
#endif // WIN32
}
return 0;
}
}
在用户成功登录算法服务器后，算法用户建立算法通道
```

6.4 demo_test_trade_spi.cpp 文件参考

Demo自定义客户端交易响应接口类

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

变量

- XTP::API::TraderApi * user_api_pointer
  *API*的全局变量指针

6.4.1 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

6.5 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

6.5.1 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

6.6 xoms_api_fund_struct.h 文件参考

定义资金划拨相关结构体类型

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

结构体

- struct XTPFundTransferReq
  用户资金请求
- struct XTPFundQueryReq
  用户资金查询请求结构体
- struct XTPFundQueryRsp
  用户资金查询响应结构体

宏定义

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

类型定义

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

6.6.1 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

6.7 xoms_api_struct.h 文件参考

定义交易类相关数据结构

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

结构体

- struct XTPOrderInsertInfo
  新订单请求
- struct XTPOrderCancelInfo
  撤单失败响应消息
- struct XTPOrderInfo
  报单响应结构体
- struct XTPOrderInfoEx
  报单响应结构体，新版本
- struct XTPTradeReport
  报单成交结构体
- struct XTPQueryOrderReq
  报单查询 // 报单查询请求-条件查询
- struct XTPQueryOrderByPageReq
  查询订单请求-分页查询
- struct XTPQueryReportByExecIdReq
  成交回报查询 // 查询成交报告请求-根据执行编号查询（保留字段）
- struct XTPQueryTraderReq
  查询成交回报请求-查询条件
- struct XTPQueryTraderByPageReq
  查询成交回报请求-分页查询
- struct XTPQueryAssetRsp
  账户资金查询响应结构体
- struct XTPQueryStkPositionReq
  查询股票持仓情况请求结构体
- struct XTPQueryStkPositionRsp
  查询股票持仓情况
- struct XTPCreditDebtExtendNotice
  用户展期请求的通知
- struct XTPFundTransferNotice
  资金内转流水通知
- struct XTPQueryFundTransferLogReq
  资金内转流水查询请求与响应
- struct XTPQueryStructuredFundInfoReq
  查询分级基金信息结构体
- struct XTPStructuredFundInfo
  查询分级基金信息响应结构体
- struct XTPQueryETFBaseReq
- struct XTPQueryETFBaseRsp
  查询股票*ETF*合约基本情况–响应结构体
- struct XTPQueryETFComponentReq
  查询股票*ETF*合约成分股信息–请求结构体,请求参数为:交易市场+*ETF*买卖代码
- struct XTPQueryETFComponentRspV1
  查询股票*ETF*成分股信息–响应结构体，旧版本。
- struct XTPQueryETFComponentRsp
  查询股票*ETF*成分股信息–响应结构体
- struct XTPQueryIPOTickerRsp
  查询当日可申购新股信息
- struct XTPQueryIPOQuotaRspV1

查询用户申购额度-旧版

- struct XTPQueryIPOQuotaRsp
  查询用户申购额度-包含创业板额度
- struct XTPUserTerminalInfoReq
  申报用户的*ip*和*mac*等信息，仅限授权用户使用
- struct XTPQueryOptionAuctionInfoReq
  查询期权竞价交易业务参考信息–请求结构体,请求参数为:交易市场+*8*位期权代码
- struct XTPQueryOptionAuctionInfoRsp
  查询期权竞价交易业务参考信息
- struct XTPCombLegStrategy
  期权组合策略的成分合约信息
- struct XTPQueryCombineStrategyInfoRsp
  查询期权组合策略信息的响应
- struct XTPOptCombLegInfo
  组合策略腿合约信息结构体
- struct XTPOptCombPlugin
  期权组合策略报单附加信息结构体
- struct XTPQueryOptCombPositionReq
  查询期权组合策略持仓情况请求结构体
- struct XTPQueryOptCombPositionRsp
  查询期权组合策略持仓信息的响应
- struct XTPQueryOptExecInfoRsp
  查询期权合约行权信息的响应
- struct XTPQueryOptCombExecPosReq
  查询期权行权合并头寸请求结构体
- struct XTPQueryOptCombExecPosRsp
  查询期权行权合并头寸的响应
- struct XTPCrdCashRepayRsp
  融资融券直接还款响应信息
- struct XTPCrdCashRepayDebtInterestFeeRsp
  融资融券现金还息费响应信息
- struct XTPCrdCashRepayInfo
  单条融资融券直接还款记录信息
- struct XTPCrdDebtInfo
  单条融资融券负债记录信息
- struct XTPCrdFundInfo
  融资融券特有帐户数据
- struct XTPClientQueryCrdDebtStockReq
  融资融券指定证券上的负债未还数量请求结构体
- struct XTPCrdDebtStockInfo
  融资融券指定证券的融券负债相关信息
- struct XTPClientQueryCrdPositionStockReq
  融券头寸证券查询请求结构体
- struct XTPClientQueryCrdPositionStkInfo
  融券头寸证券信息
- struct XTPClientQueryCrdSurplusStkReqInfo
  信用业务余券查询请求结构体
- struct XTPClientQueryCrdSurplusStkRspInfo
  信用业务余券信息
- struct XTPCreditDebtExtendReq
  用户展期请求

- struct XTPCrdFundExtraInfo
  融资融券帐户附加信息
- struct XTPCrdPositionExtraInfo
  融资融券帐户持仓附加信息
- struct XTPOptCombOrderInsertInfo
  期权组合策略新订单请求
- struct XTPOptCombOrderInfo
  期权组合策略报单响应结构体
- struct XTPOptCombOrderInfoEx
  期权组合策略报单响应结构体，新版本
- struct XTPOptCombTradeReport
  期权组合策略报单成交结构体
- struct XTPQueryOptCombOrderReq
  期权组合策略报单查询 // 期权组合策略报单查询请求-条件查询
- struct XTPQueryOptCombOrderByPageReq
  查询期权组合策略订单请求-分页查询
- struct XTPQueryOptCombReportByExecIdReq
  期权组合策略成交回报查询 // 查询期权组合策略成交报告请求-根据执行编号查询（保留字段）
- struct XTPQueryOptCombTraderReq
  查询期权组合策略成交回报请求-查询条件
- struct XTPQueryOptCombTraderByPageReq
  查询期权组合策略成交回报请求-分页查询

宏定义

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

类型定义

- typedef struct XTPOrderInfo XTPQueryOrderRsp
  报单查询响应结构体
- typedef struct XTPTradeReport XTPQueryTradeRsp
  成交回报查询响应结构体
- typedef struct XTPFundTransferNotice XTPFundTransferLog
  资金内转流水记录结构体
- typedef struct XTPQueryETFBaseRsp XTPQueryETFBaseRsp
  查询股票*ETF*合约基本情况–响应结构体
- typedef struct XTPQueryETFComponentReq XTPQueryETFComponentReq
  查询股票*ETF*合约成分股信息–请求结构体,请求参数为:交易市场+*ETF*买卖代码
- typedef struct XTPOrderCancelInfo XTPOptCombOrderCancelInfo
  期权组合策略撤单错误响应结构体
- typedef struct XTPOptCombLegInfo XTPOptCombLegInfo
  组合策略腿合约信息结构体
- typedef struct XTPOptCombPlugin XTPOptCombPlugin
  期权组合策略报单附加信息结构体
- typedef struct XTPCrdDebtInfo XTPCrdDebtInfo
  单条融资融券负债记录信息

- typedef struct XTPCrdFundInfo XTPCrdFundInfo
  融资融券特有帐户数据
- typedef struct XTPClientQueryCrdDebtStockReq XTPClientQueryCrdDebtStockReq
  融资融券指定证券上的负债未还数量请求结构体
- typedef struct XTPCrdDebtStockInfo XTPCrdDebtStockInfo
  融资融券指定证券的融券负债相关信息
- typedef struct XTPClientQueryCrdPositionStockReq XTPClientQueryCrdPositionStockReq
  融券头寸证券查询请求结构体
- typedef struct XTPClientQueryCrdPositionStkInfo XTPClientQueryCrdPositionStkInfo
  融券头寸证券信息
- typedef struct XTPClientQueryCrdSurplusStkReqInfo XTPClientQueryCrdSurplusStkReqInfo
  信用业务余券查询请求结构体
- typedef struct XTPClientQueryCrdSurplusStkRspInfo XTPClientQueryCrdSurplusStkRspInfo
  信用业务余券信息
- typedef struct XTPCreditDebtExtendNotice XTPCreditDebtExtendAck
  用户展期请求的响应结构
- typedef struct XTPCrdFundExtraInfo XTPCrdFundExtraInfo
  融资融券帐户附加信息
- typedef struct XTPCrdPositionExtraInfo XTPCrdPositionExtraInfo
  融资融券帐户持仓附加信息
- typedef struct XTPOptCombOrderInfo XTPQueryOptCombOrderRsp
  期权组合策略报单查询响应结构体
- typedef struct XTPOptCombTradeReport XTPQueryOptCombTradeRsp
  成交回报查询响应结构体

6.7.1 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

6.8 xquote_api_rebuild_tbt_struct.h 文件参考

定义行情类相关数据结构

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

结构体

- struct XTPQuoteRebuildReq
  实时行情回补查询
- struct XTPQuoteRebuildResultRsp
  实时行情回补响应结构体

类型定义

- typedef struct XTPQuoteRebuildReq XTPQuoteRebuildReq
  实时行情回补查询
- typedef struct XTPQuoteRebuildResultRsp XTPQuoteRebuildResultRsp
  实时行情回补响应结构体

6.8.1 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

6.8.2 类型定义说明

6.8.2.1 XTPQuoteRebuildReq

```
typedef struct XTPQuoteRebuildReq XTPQuoteRebuildReq
```

实时行情回补查询

实时行情回补请求结构体

6.9 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 等额外数据
- struct XTPMarketDataBondExData
  债券额外数据
- struct XTPMarketDataOptionExData
  期权额外数据
- struct XTPMarketDataStruct
  行情
- struct XTPQuoteStaticInfo
  股票行情静态信息
- struct OrderBookStruct
  订单薄
- struct XTPTickByTickEntrust
  逐笔委托
- struct XTPTickByTickTrade
  逐笔成交
- struct XTPTickByTickStatus
  逐笔状态订单
- struct XTPTickByTickStruct
  逐笔数据信息
- struct XTPTickerPriceInfo
  供查询的最新信息
- struct XTPQuoteFullInfo
  股票行情全量静态信息

类型定义

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

枚举

- enum XTP_MARKETDATA_TYPE { **XTP_MARKETDATA_ACTUAL** = 0, **XTP_MARKETDATA_OPTION** = 1 }

  *XTP_MARKETDATA_TYPE*是行情快照数据类型，*2.2.32*以前版本所用
- enum XTP_MARKETDATA_TYPE_V2 { **XTP_MARKETDATA_V2_INDEX** = 0, **XTP_MARKETDATA_V2_OPTION** = 1, **XTP_MARKETDATA_V2_ACTUAL** = 2, **XTP_MARKETDATA_V2_BOND** = 3 }

  *XTP_MARKETDATA_TYPE_V2*是行情快照数据类型，*2.2.32*版本新增字段

6.9.1 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

6.10 xtp_api_data_type.h 文件参考

定义兼容数据基本类型

宏定义

- #define MAX_TGW_CNT_PER_PBU 10

  每个*PBU*最多被*10*个*TGW*使用。
- #define XTP_VERSION_LEN 16

  存放版本号的字符串长度
- #define XTP_TRADING_DAY_LEN 9

  可交易日字符串长度
- #define XTP_TICKER_LEN 16

  存放证券代码的字符串长度
- #define XTP_TICKER_NAME_LEN 64

  存放证券名称的字符串长度
- #define XTP_LOCAL_ORDER_LEN 11

  本地报单编号的字符串长度
- #define XTP_ORDER_EXCH_LEN 17

  交易所单号的字符串长度
- #define XTP_EXEC_ID_LEN 18

  成交执行编号的字符串长度
- #define XTP_BRANCH_PBU_LEN 7

  交易所交易员代码字符串长度
- #define XTP_ACCOUNT_NAME_LEN 16

  用户资金账户的字符串长度
- #define XTP_CREDIT_DEBT_ID_LEN 33

  信用业务合约负债编号长度
- #define XTP_INET_ADDRESS_STR_LEN 64

  *IP*地址的字符串长度
- #define XTP_MAC_ADDRESS_LEN 16

*MAC*地址的字符串长度

- #define XTP_HARDDISK_SN_LEN 24
  硬盘序列号的字符串长度
- #define XTP_MACOS_SNO_LEN 21
  *MacOS*系统序列号的字符串长度
- #define XTP_STRATEGE_LEG_NUM 4
  期权组合策略最多腿数
- #define XTP_STRATEGY_ID_LEN 10
  期权组合策略代码字符串长度
- #define XTP_STRATEGY_NAME_LEN 32
  期权组合策略名称字符串长度
- #define XTP_SECONDARY_ORDER_ID_LEN 18
  期权组合策略组合编码字符串长度
- #define XTP_CNTRT_COMB_STRA_LIST_LEN 2048
  期权合约可支持的组合策略列表字符串长度
- #define XTP_COMBINED_EXECUTION_LEG_NUM 2
  期权行权合并最多成分合约数量
- #define XTP_SIDE_BUY 1
  买（新股申购，*ETF*买，配股，信用交易中担保品买）
- #define XTP_SIDE_SELL 2
  卖（逆回购，*ETF*卖，信用交易中担保品卖）
- #define XTP_SIDE_PURCHASE 7
  申购
- #define XTP_SIDE_REDEMPTION 8
  赎回
- #define XTP_SIDE_SPLIT 9
  拆分
- #define XTP_SIDE_MERGE 10
  合并
- #define XTP_SIDE_COVER 11
  改版之后的*side*的备兑，暂不支持
- #define XTP_SIDE_FREEZE 12
  改版之后的*side*锁定（对应开平标识为开）/解锁（对应开平标识为平）
- #define XTP_SIDE_MARGIN_TRADE 21
  融资买入
- #define XTP_SIDE_SHORT_SELL 22
  融券卖出
- #define XTP_SIDE_REPAY_MARGIN 23
  卖券还款
- #define XTP_SIDE_REPAY_STOCK 24
  买券还券
- #define XTP_SIDE_STOCK_REPAY_STOCK 26
  现金还款（不放在普通订单协议，另加请求和查询协议）
- #define XTP_SIDE_SURSTK_TRANS 27
  余券划转
- #define XTP_SIDE_GRTSTK_TRANSIN 28
  担保品转入
- #define XTP_SIDE_GRTSTK_TRANSOUT 29
  担保品转出
- #define XTP_SIDE_OPT_COMBINE 31
  组合策略的组合

- #define XTP_SIDE_OPT_SPLIT 32
  组合策略的拆分
- #define XTP_SIDE_OPT_SPLIT_FORCE 33
  组合策略的管理员强制拆分
- #define XTP_SIDE_OPT_SPLIT_FORCE_EXCH 34
  组合策略的交易所强制拆分
- #define XTP_SIDE_UNKNOWN 50
  未知或者无效买卖方向
- #define XTP_POSITION_EFFECT_INIT 0
  初始值或未知值开平标识，除期权外，均使用此值
- #define XTP_POSITION_EFFECT_OPEN 1
  开
- #define XTP_POSITION_EFFECT_CLOSE 2
  平
- #define XTP_POSITION_EFFECT_FORCECLOSE 3
  强平
- #define XTP_POSITION_EFFECT_CLOSETODAY 4
  平今
- #define XTP_POSITION_EFFECT_CLOSEYESTERDAY 5
  平昨
- #define XTP_POSITION_EFFECT_FORCEOFF 6
  强减
- #define XTP_POSITION_EFFECT_LOCALFORCECLOSE 7
  本地强平
- #define XTP_POSITION_EFFECT_CREDIT_FORCE_COVER 8
  信用业务追保强平
- #define XTP_POSITION_EFFECT_CREDIT_FORCE_CLEAR 9
  信用业务清偿强平
- #define XTP_POSITION_EFFECT_CREDIT_FORCE_DEBT 10
  信用业务合约到期强平
- #define XTP_POSITION_EFFECT_CREDIT_FORCE_UNCOND 11
  信用业务无条件强平
- #define XTP_POSITION_EFFECT_UNKNOWN 12
  未知的开平标识类型
- #define XTP_TRDT_COMMON '0'
  普通成交
- #define XTP_TRDT_CASH '1'
  现金替代
- #define XTP_TRDT_PRIMARY '2'
  一级市场成交
- #define XTP_TRDT_CROSS_MKT_CASH '3'
  跨市场资金成交
- #define XTP_TRDT_HK_MKT_CASH '4'
  港市资金成交
- #define XTP_TRDT_NON_SHSZ_MKT_CASH '5'
  非沪深资金成交
- #define XTP_ORDT_Normal '0'
  正常
- #define XTP_ORDT_DeriveFromQuote '1'
  报价衍生
- #define XTP_ORDT_DeriveFromCombination '2'

组合衍生
- #define XTP_ORDT_Combination '3'
  组合报单
- #define XTP_ORDT_ConditionalOrder '4'
  条件单
- #define XTP_ORDT_Swap '5'
  互换单

类型定义

- typedef char XTPVersionType[XTP_VERSION_LEN]
  版本号类型
- typedef enum XTP_LOG_LEVEL XTP_LOG_LEVEL
  *XTP_LOG_LEVEL*是日志输出级别类型
- typedef enum XTP_PROTOCOL_TYPE XTP_PROTOCOL_TYPE
  *XTP_PROTOCOL_TYPE*是通讯传输协议方式
- typedef enum XTP_EXCHANGE_TYPE XTP_EXCHANGE_TYPE
  *XTP_EXCHANGE_TYPE*是交易所类型，行情里使用
- typedef enum XTP_MARKET_TYPE XTP_MARKET_TYPE
  *XTP_MARKET_TYPE*市场类型，交易里使用
- typedef enum XTP_PRICE_TYPE XTP_PRICE_TYPE
  *XTP_PRICE_TYPE*是价格类型
- typedef uint8_t XTP_SIDE_TYPE
  *XTP_SIDE_TYPE*是买卖方向类型
- typedef uint8_t XTP_POSITION_EFFECT_TYPE
  *XTP_POSITION_EFFECT_TYPE*是开平标识类型
- typedef enum XTP_ORDER_ACTION_STATUS_TYPE XTP_ORDER_ACTION_STATUS_TYPE
  *XTP_ORDER_ACTION_STATUS_TYPE*是报单操作状态类型
- typedef enum XTP_ORDER_STATUS_TYPE XTP_ORDER_STATUS_TYPE
  *XTP_ORDER_STATUS_TYPE*是报单状态类型
- typedef enum XTP_ORDER_SUBMIT_STATUS_TYPE XTP_ORDER_SUBMIT_STATUS_TYPE
  *XTP_ORDER_SUBMIT_STATUS_TYPE*是报单提交状态类型
- typedef enum XTP_TE_RESUME_TYPE XTP_TE_RESUME_TYPE
  *XTP_TE_RESUME_TYPE*是公有流（订单响应、成交回报）重传方式
- typedef enum ETF_REPLACE_TYPE ETF_REPLACE_TYPE
  *ETF_REPLACE_TYPE*现金替代标识定义
- typedef enum XTP_TICKER_TYPE XTP_TICKER_TYPE
  *XTP_TICKER_TYPE*证券类型
- typedef enum XTP_BUSINESS_TYPE XTP_BUSINESS_TYPE
  *XTP_BUSINESS_TYPE*证券业务类型
- typedef enum XTP_ACCOUNT_TYPE XTP_ACCOUNT_TYPE
  *XTP_ACCOUNT_TYPE*账户类型
- typedef enum XTP_FUND_TRANSFER_TYPE XTP_FUND_TRANSFER_TYPE
  *XTP_FUND_TRANSFER_TYPE*是资金流转方向类型
- typedef enum XTP_FUND_QUERY_TYPE XTP_FUND_QUERY_TYPE
  *XTP_FUND_QUERY_TYPE*是柜台资金查询类型
- typedef enum XTP_FUND_OPER_STATUS XTP_FUND_OPER_STATUS
  *XTP_FUND_OPER_STATUS*柜台资金操作结果
- typedef enum XTP_DEBT_EXTEND_OPER_STATUS XTP_DEBT_EXTEND_OPER_STATUS
  *XTP_DEBT_EXTEND_OPER_STATUS*柜台负债展期操作状态

- typedef enum XTP_SPLIT_MERGE_STATUS XTP_SPLIT_MERGE_STATUS
  *XTP_SPLIT_MERGE_STATUS*是一个基金当天拆分合并状态类型
- typedef enum XTP_TBT_TYPE XTP_TBT_TYPE
  *XTP_TBT_TYPE*是一个逐笔回报类型
- typedef enum XTP_QUOTE_REBUILD_DATA_TYPE XTP_QUOTE_REBUILD_DATA_TYPE
  *XTP_QUOTE_DATA_TYPE*是行情数据类型 逐笔，快照等
- typedef enum XTP_REBUILD_RET_TYPE XTP_REBUILD_RET_TYPE
  XTP_REBUILD_RET_TYPE 实时行情回补返回结果类型
- typedef enum XTP_OPT_CALL_OR_PUT_TYPE XTP_OPT_CALL_OR_PUT_TYPE
  *XTP_OPT_CALL_OR_PUT_TYPE*是一个认沽或认购类型
- typedef enum XTP_OPT_EXERCISE_TYPE_TYPE XTP_OPT_EXERCISE_TYPE_TYPE
  *XTP_OPT_EXERCISE_TYPE_TYPE*是一个行权方式类型
- typedef enum XTP_POSITION_DIRECTION_TYPE XTP_POSITION_DIRECTION_TYPE
  *XTP_POSITION_DIRECTION_TYPE*是一个持仓方向类型
- typedef enum XTP_OPT_COVERED_OR_UNCOVERED XTP_OPT_COVERED_OR_UNCOVERED
  *XTP_OPT_COVERED_OR_UNCOVERED*是否备兑的标签
- typedef enum XTP_CRD_CR_STATUS XTP_CRD_CR_STATUS
  *XTP_CRD_CASH_REPAY_STATUS*是一个融资融券直接还款状态类型
- typedef enum XTP_OPT_POSITION_TYPE XTP_OPT_POSITION_TYPE
  *XTP_OPT_POSITION_TYPE*是一个期权持仓类型
- typedef char TXTPTradeTypeType
  *TXTPTradeTypeType*是成交类型类型
- typedef char TXTPOrderTypeType
  *TXTPOrderTypeType*是报单类型类型
- typedef enum XTP_EXPIRE_DATE_TYPE XTP_EXPIRE_DATE_TYPE
  *XTP_EXPIRE_DATE_TYPE*是一个期权组合策略合约到期日要求类型
- typedef enum XTP_UNDERLYING_TYPE XTP_UNDERLYING_TYPE
  *XTP_UNDERLYING_TYPE*是一个期权组合策略标的要求类型
- typedef enum XTP_AUTO_SPLIT_TYPE XTP_AUTO_SPLIT_TYPE
  *XTP_AUTO_SPLIT_TYPE*是一个期权组合策略自动解除枚举类型
- typedef char TXTPExerciseSeqType
  行权价顺序类型， 从*1*开始，*1*表示行权价最高，*2*次之。如果行权价相同，则填写相同数字，用*A*表示行权价大于等于*B*，*B*大于等于*C*依次类推（*C*、*D*）
- typedef enum XTP_QUALIFICATION_TYPE XTP_QUALIFICATION_TYPE
  *XTP_QUALIFICATION_TYPE*是一个证券适当性枚举类型
- typedef enum XTP_SECURITY_TYPE XTP_SECURITY_TYPE
  *XTP_SECURITY_TYPE*是一个证券详细分类枚举类型
- typedef enum XTP_POSITION_SECURITY_TYPE XTP_POSITION_SECURITY_TYPE
  *XTP_POSITION_SECURITY_TYPE*是一个持仓证券枚举类型
- typedef enum XTP_SECURITY_STATUS XTP_SECURITY_STATUS
  *XTP_SECURITY_STATUS*是一个证券状态枚举类型

枚举

- enum XTP_LOG_LEVEL {
  XTP_LOG_LEVEL_FATAL, XTP_LOG_LEVEL_ERROR, XTP_LOG_LEVEL_WARNING, XTP_LOG_LEVEL_INFO, XTP_LOG_LEVEL_DEBUG, XTP_LOG_LEVEL_TRACE }
  *XTP_LOG_LEVEL*是日志输出级别类型
- enum XTP_PROTOCOL_TYPE { XTP_PROTOCOL_TCP = 1, XTP_PROTOCOL_UDP }
  *XTP_PROTOCOL_TYPE*是通讯传输协议方式

- enum XTP_EXCHANGE_TYPE { XTP_EXCHANGE_SH = 1, XTP_EXCHANGE_SZ, XTP_EXCHANGE_UNKNOWN }

  *XTP_EXCHANGE_TYPE*是交易所类型，行情里使用
- enum XTP_MARKET_TYPE { XTP_MKT_INIT = 0, XTP_MKT_SZ_A = 1, XTP_MKT_SH_A, XTP_MKT_UNKNOWN }

  *XTP_MARKET_TYPE*市场类型，交易里使用
- enum XTP_PRICE_TYPE {
  XTP_PRICE_LIMIT = 1, XTP_PRICE_BEST_OR_CANCEL, XTP_PRICE_BEST5_OR_LIMIT, XTP_PRICE_BEST5_OR_CAN
  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_PARTTRADEDQUEUE
  XTP_ORDER_STATUS_PARTTRADEDNOTQUEUEING,
  XTP_ORDER_STATUS_NOTRADEQUEUEING, XTP_ORDER_STATUS_CANCELED, XTP_ORDER_STATUS_REJECTED,
  XTP_ORDER_STATUS_UNKNOWN }

  *XTP_ORDER_STATUS_TYPE*是报单状态类型
- enum XTP_ORDER_SUBMIT_STATUS_TYPE {
  XTP_ORDER_SUBMIT_STATUS_INSERT_SUBMITTED = 1, XTP_ORDER_SUBMIT_STATUS_INSERT_ACCEPTED,
  XTP_ORDER_SUBMIT_STATUS_INSERT_REJECTED, XTP_ORDER_SUBMIT_STATUS_CANCEL_SUBMITTED,
  XTP_ORDER_SUBMIT_STATUS_CANCEL_REJECTED, XTP_ORDER_SUBMIT_STATUS_CANCEL_ACCEPTED
  }

  *XTP_ORDER_SUBMIT_STATUS_TYPE*是报单提交状态类型
- enum XTP_TE_RESUME_TYPE { XTP_TERT_RESTART = 0, XTP_TERT_RESUME, XTP_TERT_QUICK }

  *XTP_TE_RESUME_TYPE*是公有流（订单响应、成交回报）重传方式
- enum ETF_REPLACE_TYPE {
  ERT_CASH_FORBIDDEN = 0, ERT_CASH_OPTIONAL, ERT_CASH_MUST, ERT_CASH_RECOMPUTE_INTER_SZ,
  ERT_CASH_MUST_INTER_SZ, ERT_CASH_RECOMPUTE_INTER_OTHER, ERT_CASH_MUST_INTER_OTHER,
  ERT_CASH_RECOMPUTE_INTER_HK,
  ERT_CASH_MUST_INTER_HK, EPT_INVALID }

  *ETF_REPLACE_TYPE*现金替代标识定义
- enum XTP_TICKER_TYPE {
  XTP_TICKER_TYPE_STOCK = 0, XTP_TICKER_TYPE_INDEX, XTP_TICKER_TYPE_FUND, XTP_TICKER_TYPE_BOND,
  XTP_TICKER_TYPE_OPTION, XTP_TICKER_TYPE_TECH_STOCK, XTP_TICKER_TYPE_UNKNOWN }

  *XTP_TICKER_TYPE*证券类型
- enum XTP_BUSINESS_TYPE {
  XTP_BUSINESS_TYPE_CASH = 0, XTP_BUSINESS_TYPE_IPOS, XTP_BUSINESS_TYPE_REPO,
  XTP_BUSINESS_TYPE_ETF,
  XTP_BUSINESS_TYPE_MARGIN, XTP_BUSINESS_TYPE_DESIGNATION, XTP_BUSINESS_TYPE_ALLOTMENT,
  XTP_BUSINESS_TYPE_STRUCTURED_FUND_PURCHASE_REDEMPTION,
  XTP_BUSINESS_TYPE_STRUCTURED_FUND_SPLIT_MERGE, XTP_BUSINESS_TYPE_MONEY_FUND,
  XTP_BUSINESS_TYPE_OPTION, XTP_BUSINESS_TYPE_EXECUTE,
  XTP_BUSINESS_TYPE_FREEZE, XTP_BUSINESS_TYPE_OPTION_COMBINE, XTP_BUSINESS_TYPE_EXECUTE_COME
  XTP_BUSINESS_TYPE_UNKNOWN }

  *XTP_BUSINESS_TYPE*证券业务类型
- enum XTP_ACCOUNT_TYPE { XTP_ACCOUNT_NORMAL = 0, XTP_ACCOUNT_CREDIT, XTP_ACCOUNT_DERIVE, XTP_ACCOUNT_UNKNOWN }

  *XTP_ACCOUNT_TYPE*账户类型
- enum XTP_FUND_TRANSFER_TYPE {
  XTP_FUND_TRANSFER_OUT = 0, XTP_FUND_TRANSFER_IN, XTP_FUND_INTER_TRANSFER_OUT,
  XTP_FUND_INTER_TRANSFER_IN,
  XTP_FUND_TRANSFER_UNKNOWN }

*XTP_FUND_TRANSFER_TYPE*是资金流转方向类型

- enum XTP_FUND_QUERY_TYPE { XTP_FUND_QUERY_JZ = 0, XTP_FUND_QUERY_INTERNAL, XTP_FUND_QUERY_UNKNOWN }

  *XTP_FUND_QUERY_TYPE*是柜台资金查询类型
- enum XTP_FUND_OPER_STATUS { XTP_FUND_OPER_PROCESSING = 0, XTP_FUND_OPER_SUCCESS, XTP_FUND_OPER_FAILED, XTP_FUND_OPER_SUBMITTED, XTP_FUND_OPER_UNKNOWN }

  *XTP_FUND_OPER_STATUS*柜台资金操作结果
- enum XTP_DEBT_EXTEND_OPER_STATUS { XTP_DEBT_EXTEND_OPER_PROCESSING = 0, XTP_DEBT_EXTEND_OPER_SUBMITTED, XTP_DEBT_EXTEND_OPER_ XTP_DEBT_EXTEND_OPER_FAILED, XTP_DEBT_EXTEND_OPER_UNKNOWN }

  *XTP_DEBT_EXTEND_OPER_STATUS*柜台负债展期操作状态
- enum XTP_SPLIT_MERGE_STATUS { XTP_SPLIT_MERGE_STATUS_ALLOW = 0, XTP_SPLIT_MERGE_STATUS_ONLY_S XTP_SPLIT_MERGE_STATUS_ONLY_MERGE, XTP_SPLIT_MERGE_STATUS_FORBIDDEN }

  *XTP_SPLIT_MERGE_STATUS*是一个基金当天拆分合并状态类型
- enum XTP_TBT_TYPE { XTP_TBT_ENTRUST = 1, XTP_TBT_TRADE = 2, XTP_TBT_STATE = 3 }

  *XTP_TBT_TYPE*是一个逐笔回报类型
- enum XTP_QUOTE_REBUILD_DATA_TYPE { XTP_QUOTE_REBUILD_UNKNOW = 0, XTP_QUOTE_REBUILD_MD = 1, XTP_QUOTE_REBUILD_TBT = 2 }

  *XTP_QUOTE_DATA_TYPE*是行情数据类型 逐笔，快照等
- enum XTP_REBUILD_RET_TYPE { XTP_REBUILD_RET_COMPLETE = 1, XTP_REBUILD_RET_PARTLY = 2, XTP_REBUILD_RET_NO_DATA = 3, XTP_REBUILD_RET_PARAM_ERR = 4, XTP_REBUILD_RET_FREQUENTLY = 5 }

  XTP_REBUILD_RET_TYPE 实时行情回补返回结果类型
- enum XTP_OPT_CALL_OR_PUT_TYPE { XTP_OPT_CALL = 1, XTP_OPT_PUT = 2 }

  *XTP_OPT_CALL_OR_PUT_TYPE*是一个认沽或认购类型
- enum XTP_OPT_EXERCISE_TYPE_TYPE { XTP_OPT_EXERCISE_TYPE_EUR = 1, XTP_OPT_EXERCISE_TYPE_AME = 2 }

  *XTP_OPT_EXERCISE_TYPE_TYPE*是一个行权方式类型
- enum XTP_POSITION_DIRECTION_TYPE { XTP_POSITION_DIRECTION_NET = 0, XTP_POSITION_DIRECTION_LONG, XTP_POSITION_DIRECTION_SHORT, XTP_POSITION_DIRECTION_COVERED }

  *XTP_POSITION_DIRECTION_TYPE*是一个持仓方向类型
- enum XTP_OPT_COVERED_OR_UNCOVERED { XTP_POSITION_UNCOVERED = 0, XTP_POSITION_COVERED }

  *XTP_OPT_COVERED_OR_UNCOVERED*是否备兑的标签
- enum XTP_CRD_CR_STATUS { XTP_CRD_CR_INIT = 0, XTP_CRD_CR_SUCCESS, XTP_CRD_CR_FAILED }

  *XTP_CRD_CASH_REPAY_STATUS*是一个融资融券直接还款状态类型
- enum XTP_OPT_POSITION_TYPE { XTP_OPT_POSITION_TYPE_CONTRACT = 0, XTP_OPT_POSITION_TYPE_COMBINE = 1 }

  *XTP_OPT_POSITION_TYPE*是一个期权持仓类型
- enum XTP_ORDER_DETAIL_TYPE { XTP_ORDER_DETAIL_TYPE_NEW_ORDER = 0, XTP_ORDER_DETAIL_TYPE_CANC = 1, XTP_ORDER_DETAIL_TYPE_OPT_COMB_NEW_ORDER = 2, XTP_ORDER_DETAIL_TYPE_OPT_COMB_CANCEL_C = 3 }

  *XTP_ORDER_TYPE*是一个订单的类型
- enum XTPTerminalType { XTP_TERMINAL_PC = 1, XTP_TERMINAL_ANDROID, XTP_TERMINAL_IOS, XTP_TERMINAL_WP, XTP_TERMINAL_STATION, XTP_TERMINAL_TEL, XTP_TERMINAL_PC_LINUX }

  *XTPTerminalType*是一种终端类型枚举，仅供授权系统使用
- enum XTP_EXPIRE_DATE_TYPE { XTP_EXP_DATE_SAME = 0, XTP_EXP_DATE_DIFF, XTP_EXP_DATE_NON }

*XTP_EXPIRE_DATE_TYPE*是一个期权组合策略合约到期日要求类型

- enum XTP_UNDERLYING_TYPE { XTP_UNDERLYING_SAME = 0, XTP_UNDERLYING_DIFF, XTP_UNDERLYING_NON }

  *XTP_UNDERLYING_TYPE*是一个期权组合策略标的要求类型
- enum XTP_AUTO_SPLIT_TYPE { XTP_AUTO_SPLIT_EXPDAY = 0, XTP_AUTO_SPLIT_PREDAY, XTP_AUTO_SPLIT_PRE2DAY, XTP_AUTO_SPLIT_NON }

  *XTP_AUTO_SPLIT_TYPE*是一个期权组合策略自动解除枚举类型
- enum XTP_QUALIFICATION_TYPE { XTP_QUALIFICATION_PUBLIC = 0, XTP_QUALIFICATION_COMMON = 1, XTP_QUALIFICATION_ORGANIZATION = 2, XTP_QUALIFICATION_UNKNOWN = 3 }

  *XTP_QUALIFICATION_TYPE*是一个证券适当性枚举类型
- enum XTP_SECURITY_TYPE { XTP_SECURITY_MAIN_BOARD = 0, XTP_SECURITY_SECOND_BOARD, XTP_SECURITY_STARTUP_BOARD, XTP_SECURITY_INDEX, XTP_SECURITY_TECH_BOARD = 4, XTP_SECURITY_STATE_BOND = 5, XTP_SECURITY_ENTERPRICE_BOND = 6, XTP_SECURITY_COMPANEY_BOND = 7, XTP_SECURITY_CONVERTABLE_BOND = 8, XTP_SECURITY_NATIONAL_BOND_REVERSE_REPO = 12, XTP_SECURITY_ETF_SINGLE_MARKET_STOCK = 14, XTP_SECURITY_ETF_INTER_MARKET_STOCK, **XTP_SECURITY_ETF_CROSS_BORDER_STOCK** = 16, XTP_SECURITY_ETF_SINGLE_MARKET_BOND = 17, XTP_SECURITY_ETF_GOLD = 19, XTP_SECURITY_STRUCTURED_FUND_CHILD = 24, XTP_SECURITY_SZSE_RECREATION_FUND = 26, XTP_SECURITY_STOCK_OPTION = 29, XTP_SECURITY_ETF_OPTIO = 30, XTP_SECURITY_ALLOTMENT = 100, XTP_SECURITY_MONETARY_FUND_SHCR = 110, XTP_SECURITY_MONETARY_FUND_SHTR = 111, XTP_SECURITY_MONETARY_FUND_SZ = 112, XTP_SECURITY_OTHERS = 255 }

  *XTP_SECURITY_TYPE*是一个证券详细分类枚举类型
- enum XTP_POSITION_SECURITY_TYPE { XTP_POSITION_SECURITY_NORMAL = 0, XTP_POSITION_SECURITY_PLACE = 1, XTP_POSITION_SECURITY_UNKNOWN = 2 }

  *XTP_POSITION_SECURITY_TYPE*是一个持仓证券枚举类型
- enum XTP_SECURITY_STATUS { XTP_SECURITY_STATUS_ST = 0, XTP_SECURITY_STATUS_N_IPO, XTP_SECURITY_STATUS_COMMON, XTP_SECURITY_STATUS_RESUME, XTP_SECURITY_STATUS_DELISTING = 10, XTP_SECURITY_STATUS_OTHERS = 255 }

  *XTP_SECURITY_STATUS*是一个证券状态枚举类型

6.10.1 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

6.10.2 宏定义说明

6.10.2.1 XTP_SIDE_STOCK_REPAY_STOCK

```
#define XTP_SIDE_STOCK_REPAY_STOCK 26
```

现金还款（不放在普通订单协议，另加请求和查询协议）

现券还券

6.10.3 枚举类型说明

6.10.3.1 ETF_REPLACE_TYPE

enum ETF_REPLACE_TYPE

ETF_REPLACE_TYPE现金替代标识定义

枚举值

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

6.10.3.2 XTP_ACCOUNT_TYPE

enum XTP_ACCOUNT_TYPE

XTP_ACCOUNT_TYPE账户类型

枚举值

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

6.10.3.3 XTP_AUTO_SPLIT_TYPE

enum XTP_AUTO_SPLIT_TYPE

XTP_AUTO_SPLIT_TYPE是一个期权组合策略自动解除枚举类型

枚举值

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

6.10.3.4 XTP_BUSINESS_TYPE

enum XTP_BUSINESS_TYPE

XTP_BUSINESS_TYPE证券业务类型

枚举值

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

6.10.3.5 XTP_CRD_CR_STATUS

enum XTP_CRD_CR_STATUS

XTP_CRD_CASH_REPAY_STATUS是一个融资融券直接还款状态类型

枚举值

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

6.10.3.6 XTP_DEBT_EXTEND_OPER_STATUS

enum XTP_DEBT_EXTEND_OPER_STATUS

XTP_DEBT_EXTEND_OPER_STATUS柜台负债展期操作状态

枚举值

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

6.10.3.7 XTP_EXCHANGE_TYPE

enum XTP_EXCHANGE_TYPE

XTP_EXCHANGE_TYPE是交易所类型，行情里使用

枚举值

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

6.10.3.8 XTP_EXPIRE_DATE_TYPE

enum XTP_EXPIRE_DATE_TYPE

XTP_EXPIRE_DATE_TYPE是一个期权组合策略合约到期日要求类型

枚举值

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

6.10.3.9 XTP_FUND_OPER_STATUS

enum XTP_FUND_OPER_STATUS

XTP_FUND_OPER_STATUS柜台资金操作结果

枚举值

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

6.10.3.10 XTP_FUND_QUERY_TYPE

enum XTP_FUND_QUERY_TYPE

XTP_FUND_QUERY_TYPE是柜台资金查询类型

枚举值

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

6.10.3.11 XTP_FUND_TRANSFER_TYPE

enum XTP_FUND_TRANSFER_TYPE

XTP_FUND_TRANSFER_TYPE是资金流转方向类型

枚举值

XTP_FUND_TRANSFER_OUT	转出 从XTP转出到柜台
XTP_FUND_TRANSFER_IN	转入 从柜台转入XTP
XTP_FUND_INTER_TRANSFER_OUT	跨节点转出 从本XTP节点1，转出到对端XTP节点2，XTP服务器之间划拨，只能“一账号两中心”跨节点用户使用
XTP_FUND_INTER_TRANSFER_IN	跨节点转入 从对端XTP节点2，转入到本XTP节点1，XTP服务器之间划拨，只能“一账号两中心”跨节点用户使用
XTP_FUND_TRANSFER_UNKNOWN	未知类型

6.10.3.12 XTP_LOG_LEVEL

enum XTP_LOG_LEVEL

XTP_LOG_LEVEL是日志输出级别类型

枚举值

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

6.10.3.13 XTP_MARKET_TYPE

enum XTP_MARKET_TYPE

XTP_MARKET_TYPE市场类型，交易里使用

枚举值

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

6.10.3.14 XTP_OPT_CALL_OR_PUT_TYPE

enum XTP_OPT_CALL_OR_PUT_TYPE

XTP_OPT_CALL_OR_PUT_TYPE是一个认沽或认购类型

枚举值

XTP_OPT_CALL	认购
XTP_OPT_PUT	认沽

6.10.3.15 XTP_OPT_COVERED_OR_UNCOVERED

enum XTP_OPT_COVERED_OR_UNCOVERED

XTP_OPT_COVERED_OR_UNCOVERED是否备兑的标签

枚举值

XTP_POSITION_UNCOVERED	非备兑
XTP_POSITION_COVERED	备兑

6.10.3.16 XTP_OPT_EXERCISE_TYPE_TYPE

enum XTP_OPT_EXERCISE_TYPE_TYPE

XTP_OPT_EXERCISE_TYPE_TYPE是一个行权方式类型

枚举值

XTP_OPT_EXERCISE_TYPE_EUR	欧式
XTP_OPT_EXERCISE_TYPE_AME	美式

6.10.3.17 XTP_OPT_POSITION_TYPE

enum XTP_OPT_POSITION_TYPE

XTP_OPT_POSITION_TYPE是一个期权持仓类型

枚举值

XTP_OPT_POSITION_TYPE_CONTRACT	单合约持仓
XTP_OPT_POSITION_TYPE_COMBINED	组合策略持仓

6.10.3.18 XTP_ORDER_ACTION_STATUS_TYPE

enum XTP_ORDER_ACTION_STATUS_TYPE

XTP_ORDER_ACTION_STATUS_TYPE是报单操作状态类型

枚举值

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

6.10.3.19 XTP_ORDER_DETAIL_TYPE

enum XTP_ORDER_DETAIL_TYPE

XTP_ORDER_TYPE是一个订单的类型

枚举值

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

6.10.3.20 XTP_ORDER_STATUS_TYPE

enum XTP_ORDER_STATUS_TYPE

XTP_ORDER_STATUS_TYPE是报单状态类型

枚举值

XTP_ORDER_STATUS_INIT	初始化

枚举值

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

6.10.3.21 XTP_ORDER_SUBMIT_STATUS_TYPE

enum XTP_ORDER_SUBMIT_STATUS_TYPE

XTP_ORDER_SUBMIT_STATUS_TYPE是报单提交状态类型

枚举值

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

6.10.3.22 XTP_POSITION_DIRECTION_TYPE

enum XTP_POSITION_DIRECTION_TYPE

XTP_POSITION_DIRECTION_TYPE是一个持仓方向类型

枚举值

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

6.10.3.23 XTP_POSITION_SECURITY_TYPE

enum XTP_POSITION_SECURITY_TYPE

XTP_POSITION_SECURITY_TYPE是一个持仓证券枚举类型

枚举值

XTP_POSITION_SECURITY_NORMAL	普通持仓
XTP_POSITION_SECURITY_PLACEMENT	配售类型的持仓，包含配股、配债等
XTP_POSITION_SECURITY_UNKNOWN	未知类型

6.10.3.24 XTP_PRICE_TYPE

enum XTP_PRICE_TYPE

XTP_PRICE_TYPE是价格类型

枚举值

XTP_PRICE_LIMIT	限价单-沪 / 深 / 沪期权 / 深期权 （除普通股票业务外，其余未特指的业务均使用此种类型）
XTP_PRICE_BEST_OR_CANCEL	即时成交剩余转撤销，市价单-深 / 沪期权 / 深期权
XTP_PRICE_BEST5_OR_LIMIT	最优五档即时成交剩余转限价，市价单-沪
XTP_PRICE_BEST5_OR_CANCEL	最优5档即时成交剩余转撤销，市价单-沪深 / 深期权
XTP_PRICE_ALL_OR_CANCEL	全部成交或撤销,市价单-深 / 沪期权 / 深期权
XTP_PRICE_FORWARD_BEST	本方最优，市价单-深 / 深期权 / 沪科创板
XTP_PRICE_REVERSE_BEST_LIMIT	对方最优剩余转限价，市价单-深 / 沪期权 / 深期权 / 沪科创板
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

6.10.3.25 XTP_PROTOCOL_TYPE

enum XTP_PROTOCOL_TYPE

XTP_PROTOCOL_TYPE是通讯传输协议方式

枚举值

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

6.10.3.26 XTP_QUALIFICATION_TYPE

enum XTP_QUALIFICATION_TYPE

XTP_QUALIFICATION_TYPE是一个证券适当性枚举类型

枚举值

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

6.10.3.27 XTP_QUOTE_REBUILD_DATA_TYPE

enum XTP_QUOTE_REBUILD_DATA_TYPE

XTP_QUOTE_DATA_TYPE是行情数据类型 逐笔，快照等

枚举值

XTP_QUOTE_REBUILD_UNKNOW	未知类型
XTP_QUOTE_REBUILD_MD	快照类型
XTP_QUOTE_REBUILD_TBT	逐笔类型

6.10.3.28 XTP_REBUILD_RET_TYPE

enum XTP_REBUILD_RET_TYPE

XTP_REBUILD_RET_TYPE 实时行情回补返回结果类型

枚举值

XTP_REBUILD_RET_COMPLETE	全部数据
XTP_REBUILD_RET_PARTLY	部分数据
XTP_REBUILD_RET_NO_DATA	没有数据
XTP_REBUILD_RET_PARAM_ERR	参数错误
XTP_REBUILD_RET_FREQUENTLY	请求频繁

6.10.3.29 XTP_SECURITY_STATUS

enum XTP_SECURITY_STATUS

XTP_SECURITY_STATUS是一个证券状态枚举类型

枚举值

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

6.10.3.30 XTP_SECURITY_TYPE

enum XTP_SECURITY_TYPE

XTP_SECURITY_TYPE是一个证券详细分类枚举类型

枚举值

XTP_SECURITY_MAIN_BOARD	主板股票
XTP_SECURITY_SECOND_BOARD	中小板股票
XTP_SECURITY_STARTUP_BOARD	创业板股票
XTP_SECURITY_INDEX	指数
XTP_SECURITY_TECH_BOARD	科创板股票(上海)
XTP_SECURITY_STATE_BOND	国债
XTP_SECURITY_ENTERPRICE_BOND	企业债
XTP_SECURITY_COMPANEY_BOND	公司债
XTP_SECURITY_CONVERTABLE_BOND	转换债券
XTP_SECURITY_NATIONAL_BOND_REVERSE_REPO	国债逆回购
XTP_SECURITY_ETF_SINGLE_MARKET_STOCK	本市场股票 ETF
XTP_SECURITY_ETF_INTER_MARKET_STOCK	跨市场股票 ETF
XTP_SECURITY_ETF_SINGLE_MARKET_BOND	本市场实物债券 ETF
XTP_SECURITY_ETF_GOLD	黄金 ETF
XTP_SECURITY_STRUCTURED_FUND_CHILD	分级基金子基金
XTP_SECURITY_SZSE_RECREATION_FUND	深交所仅申赎基金
XTP_SECURITY_STOCK_OPTION	个股期权
XTP_SECURITY_ETF_OPTION	ETF期权
XTP_SECURITY_ALLOTMENT	配股

枚举值

XTP_SECURITY_MONETARY_FUND_SHCR	上交所申赎型货币基金
XTP_SECURITY_MONETARY_FUND_SHTR	上交所交易型货币基金
XTP_SECURITY_MONETARY_FUND_SZ	深交所货币基金
XTP_SECURITY_OTHERS	其他

6.10.3.31 XTP_SPLIT_MERGE_STATUS

enum XTP_SPLIT_MERGE_STATUS

XTP_SPLIT_MERGE_STATUS是一个基金当天拆分合并状态类型

枚举值

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

6.10.3.32 XTP_TBT_TYPE

enum XTP_TBT_TYPE

XTP_TBT_TYPE是一个逐笔回报类型

枚举值

XTP_TBT_ENTRUST	逐笔委托
XTP_TBT_TRADE	逐笔成交
XTP_TBT_STATE	逐笔状态订单，2.2.32版本新增字段，为上海新债券Level2行情中独有

6.10.3.33 XTP_TE_RESUME_TYPE

enum XTP_TE_RESUME_TYPE

XTP_TE_RESUME_TYPE是公有流（订单响应、成交回报）重传方式

枚举值

XTP_TERT_RESTART	从本交易日开始重传
XTP_TERT_RESUME	从从上次收到的续传（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

6.10.3.34 XTP_TICKER_TYPE

enum XTP_TICKER_TYPE

XTP_TICKER_TYPE证券类型

枚举值

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

6.10.3.35 XTP_UNDERLYING_TYPE

enum XTP_UNDERLYING_TYPE

XTP_UNDERLYING_TYPE是一个期权组合策略标的要求类型

枚举值

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

6.10.3.36 XTPTerminalType

enum XTPTerminalType

XTPTerminalType是一种终端类型枚举，仅供授权系统使用

枚举值

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

6.11 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"
#include "xquote_api_rebuild_tbt_struct.h"
```

6.11.1 详细描述

定义业务数据结构

作者

中泰证券股份有限公司

6.12 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

6.12.1 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

6.13 xtp_trader_api.h 文件参考

定义客户端交易接口

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

结构体

- class TraderSpi
  交易接口响应类
- class TraderApi
  交易接口类

6.13.1 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