

XTP极速交易系统QuoteAPI

制作者 中泰证券股份有限公司

2017年 十一月 15日 星期三 11:17:26

Contents

1	XTP 极速行情交易系统 Quote API 1.1.16.9	1
2	继承关系索引	3
2.1	类继承关系	3
3	结构体索引	5
3.1	结构体	5
4	文件索引	7
4.1	文件列表	7
5	结构体说明	9
5.1	DemoTestMdSpi类 参考	9
5.1.1	详细描述	10
5.1.2	成员函数说明	10
5.1.2.1	OnTickByTick(XTP_TBT *tbt_data)	10
5.2	OrderBookStruct结构体 参考	10
5.2.1	详细描述	11
5.3	QuoteApi类 参考	11
5.3.1	详细描述	12
5.3.2	成员函数说明	12
5.3.2.1	CreateQuoteApi(uint8_t client_id, const char *save_file_path, XTP_LOG_LEVEL log_level=XTP_LOG_LEVEL_DEBUG)	12
5.3.2.2	GetApiLastError()=0	13
5.3.2.3	GetApiVersion()=0	13
5.3.2.4	GetTradingDay()=0	13
5.3.2.5	Login(const char *ip, int port, const char *user, const char *password, XTP_PROTOCOL_TYPE sock_type)=0	13
5.3.2.6	Logout()=0	14
5.3.2.7	QueryAllTickers(XTP_EXCHANGE_TYPE exchange_id)=0	14
5.3.2.8	QueryAllTickersPriceInfo()=0	14
5.3.2.9	QueryTickersPriceInfo(char *ticker[], int count, XTP_EXCHANGE_TYPE exchange_id)=0	14
5.3.2.10	RegisterSpi(QuoteSpi *spi)=0	15

5.3.2.11	Release() ₀	15
5.3.2.12	SetHeartBeatInterval(uint32_t interval) ₀	15
5.3.2.13	SetUDPBufferSize(uint32_t buff_size) ₀	15
5.3.2.14	SubscribeAllMarketData() ₀	15
5.3.2.15	SubscribeAllOrderBook() ₀	16
5.3.2.16	SubscribeAllTickByTick() ₀	16
5.3.2.17	SubscribeMarketData(char *ticker[], int count, XTP_EXCHANGE_TYP↵ E exchange_id) ₀	16
5.3.2.18	SubscribeOrderBook(char *ticker[], int count, XTP_EXCHANGE_TYP↵ E exchange_id) ₀	16
5.3.2.19	SubscribeTickByTick(char *ticker[], int count, XTP_EXCHANGE_TYP↵ E exchange_id) ₀	17
5.3.2.20	UnSubscribeAllMarketData() ₀	17
5.3.2.21	UnSubscribeAllOrderBook() ₀	17
5.3.2.22	UnSubscribeAllTickByTick() ₀	18
5.3.2.23	UnSubscribeMarketData(char *ticker[], int count, XTP_EXCHANGE_TYP↵ E exchange_id) ₀	18
5.3.2.24	UnSubscribeOrderBook(char *ticker[], int count, XTP_EXCHANGE_TYP↵ E exchange_id) ₀	18
5.3.2.25	UnSubscribeTickByTick(char *ticker[], int count, XTP_EXCHANGE_TYP↵ E exchange_id) ₀	18
5.4	QuoteSpi类 参考	19
5.4.1	详细描述	20
5.4.2	成员函数说明	20
5.4.2.1	OnDepthMarketData(XTPMD *market_data, int64_t bid1_qty[], int32_t bid1↵ count, int32_t max_bid1_count, int64_t ask1_qty[], int32_t ask1_count, int32_t max_ask1_count)	20
5.4.2.2	OnDisconnected(int reason)	20
5.4.2.3	OnError(XTPRI *error_info)	20
5.4.2.4	OnOrderBook(XTPOB *order_book)	21
5.4.2.5	OnQueryAllTickers(XTPQSI *ticker_info, XTPRI *error_info, bool is_last)	21
5.4.2.6	OnQueryTickersPriceInfo(XTPPTPI *ticker_info, XTPRI *error_info, bool is_last)	21
5.4.2.7	OnSubMarketData(XTPST *ticker, XTPRI *error_info, bool is_last)	21
5.4.2.8	OnSubOrderBook(XTPST *ticker, XTPRI *error_info, bool is_last)	22
5.4.2.9	OnSubscribeAllMarketData(XTPRI *error_info)	22
5.4.2.10	OnSubscribeAllOrderBook(XTPRI *error_info)	22
5.4.2.11	OnSubscribeAllTickByTick(XTPRI *error_info)	23
5.4.2.12	OnSubTickByTick(XTPST *ticker, XTPRI *error_info, bool is_last)	23
5.4.2.13	OnTickByTick(XTPBT *tbt_data)	23
5.4.2.14	OnUnSubMarketData(XTPST *ticker, XTPRI *error_info, bool is_last)	24
5.4.2.15	OnUnSubOrderBook(XTPST *ticker, XTPRI *error_info, bool is_last)	25
5.4.2.16	OnUnSubscribeAllMarketData(XTPRI *error_info)	25

5.4.2.17 OnUnSubscribeAllOrderBook(XTPRI *error_info)	25
5.4.2.18 OnUnSubscribeAllTickByTick(XTPRI *error_info)	26
5.4.2.19 OnUnSubTickByTick(XTPST *ticker, XTPRI *error_info, bool is_last)	26
5.5 XTPFundTransferNotice结构体 参考	26
5.5.1 详细描述	27
5.6 XTPFundTransferReq结构体 参考	27
5.6.1 详细描述	27
5.7 XTPMarketDataStruct结构体 参考	27
5.7.1 详细描述	30
5.8 XTPOrderCancelInfo结构体 参考	30
5.8.1 详细描述	30
5.9 XTPOrderInfo结构体 参考	31
5.9.1 详细描述	32
5.10 XTPOrderInsertInfo结构体 参考	32
5.10.1 详细描述	32
5.11 XTPQueryAssetRsp结构体 参考	32
5.11.1 详细描述	33
5.12 XTPQueryETFBaseReq结构体 参考	33
5.12.1 详细描述	33
5.13 XTPQueryETFBaseRsp结构体 参考	34
5.13.1 详细描述	34
5.14 XTPQueryETFComponentReq结构体 参考	34
5.14.1 详细描述	35
5.15 XTPQueryETFComponentRsp结构体 参考	35
5.15.1 详细描述	35
5.16 XTPQueryFundTransferLogReq结构体 参考	35
5.16.1 详细描述	36
5.17 XTPQueryIPOQuotaRsp结构体 参考	36
5.17.1 详细描述	36
5.18 XTPQueryIPOTickerRsp结构体 参考	36
5.18.1 详细描述	37
5.19 XTPQueryOrderReq结构体 参考	37
5.19.1 详细描述	37
5.20 XTPQueryReportByExecIdReq结构体 参考	37
5.20.1 详细描述	37
5.21 XTPQueryStkPositionRsp结构体 参考	38
5.21.1 详细描述	38
5.22 XTPQueryStructuredFundInfoReq结构体 参考	38
5.22.1 详细描述	39
5.23 XTPQueryTraderReq结构体 参考	39

5.23.1 详细描述	39
5.24 XTPQuoteStaticInfo结构体 参考	39
5.24.1 详细描述	40
5.25 XTPRspInfoStruct结构体 参考	40
5.25.1 详细描述	40
5.26 XTPSpecificTickerStruct结构体 参考	40
5.26.1 详细描述	41
5.27 XTPStructuredFundInfo结构体 参考	41
5.27.1 详细描述	41
5.28 XTPTickByTickEntrust结构体 参考	41
5.28.1 详细描述	42
5.29 XTPTickByTickStruct结构体 参考	42
5.29.1 详细描述	42
5.30 XTPTickByTickTrade结构体 参考	43
5.30.1 详细描述	43
5.30.2 结构体成员变量说明	43
5.30.2.1 trade_flag	43
5.31 XTPTickerPriceInfo结构体 参考	43
5.31.1 详细描述	44
5.32 XTPTradeReport结构体 参考	44
5.32.1 详细描述	45
6 文件说明	47
6.1 demo_test_quote_api.cpp 文件参考	47
6.1.1 详细描述	47
6.1.2 函数说明	48
6.1.2.1 main()	48
6.1.3 变量说明	48
6.1.3.1 ticker_count	48
6.2 demo_test_quote_spi.h 文件参考	48
6.2.1 详细描述	48
6.3 xoms_api_fund_struct.h 文件参考	49
6.3.1 详细描述	49
6.4 xoms_api_struct.h 文件参考	49
6.4.1 详细描述	50
6.5 xquote_api_struct.h 文件参考	51
6.5.1 详细描述	51
6.6 xtp_api_data_type.h 文件参考	52
6.6.1 详细描述	55
6.6.2 枚举类型说明	55

6.6.2.1	ETF_REPLACE_TYPE	55
6.6.2.2	XTP_ACCOUNT_TYPE	55
6.6.2.3	XTP_BUSINESS_TYPE	55
6.6.2.4	XTP_EXCHANGE_TYPE	56
6.6.2.5	XTP_FUND_OPER_STATUS	56
6.6.2.6	XTP_FUND_TRANSFER_TYPE	56
6.6.2.7	XTP_LOG_LEVEL	57
6.6.2.8	XTP_MARKET_TYPE	57
6.6.2.9	XTP_ORDER_ACTION_STATUS_TYPE	57
6.6.2.10	XTP_ORDER_STATUS_TYPE	57
6.6.2.11	XTP_ORDER_SUBMIT_STATUS_TYPE	58
6.6.2.12	XTP_PRICE_TYPE	58
6.6.2.13	XTP_PROTOCOL_TYPE	58
6.6.2.14	XTP_SIDE_TYPE	58
6.6.2.15	XTP_SPLIT_MERGE_STATUS	59
6.6.2.16	XTP_TBT_TYPE	59
6.6.2.17	XTP_TE_RESUME_TYPE	59
6.6.2.18	XTP_TICKER_TYPE	59
6.7	xtp_api_struct.h 文件参考	60
6.7.1	详细描述	60
6.8	xtp_api_struct_common.h 文件参考	60
6.8.1	详细描述	60
6.9	xtp_quote_api.h 文件参考	61
6.9.1	详细描述	61
索引		63

Chapter 1

XTP 极速行情交易系统 Quote API 1.1.16.9

本项目是XTP项目中的行情类接口

(1) XTP的行情订阅接口和响应类 [xtp_quote_api.h](#)

(2) 行情订阅测试Demo [demo_test_quote_api.cpp](#)

Chapter 2

继承关系索引

2.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

OrderBookStruct	10
QuoteApi	11
QuoteSpi	19
DemoTestMdSpi	9
XTPFundTransferNotice	26
XTPFundTransferReq	27
XTPMarketDataStruct	27
XTPOrderCancelInfo	30
XTPOrderInfo	31
XTPOrderInsertInfo	32
XTPQueryAssetRsp	32
XTPQueryETFBaseReq	33
XTPQueryETFBaseRsp	34
XTPQueryETFComponentReq	34
XTPQueryETFComponentRsp	35
XTPQueryFundTransferLogReq	35
XTPQueryIPOQuotaRsp	36
XTPQueryIPOTickerRsp	36
XTPQueryOrderReq	37
XTPQueryReportByExecIdReq	37
XTPQueryStkPositionRsp	38
XTPQueryStructuredFundInfoReq	38
XTPQueryTraderReq	39
XTPQuoteStaticInfo	39
XTPRspInfoStruct	40
XTPSpecificTickerStruct	40
XTPStructuredFundInfo	41
XTPTickByTickEntrust	41
XTPTickByTickStruct	42
XTPTickByTickTrade	43
XPTTickerPriceInfo	43
XTPTradeReport	44

结构体索引

这里列出了所有结构体，并附带简要说明：

DemoTestMdSpi		
Demo	自定义行情订阅接口响应类	9
OrderBookStruct		
定单簿		10
QuoteApi		
行情订阅接口类		11
QuoteSpi		
行情回调类		19
XTPFundTransferNotice		
资金内转流水通知		26
XTPFundTransferReq		
用户资金请求		27
XTPMarketDataStruct		
行情		27
XTPOrderCancelInfo		
撤单失败响应消息		30
XTPOrderInfo		
报单响应结构体		31
XTPOrderInsertInfo		
新订单请求		32
XTPQueryAssetRsp		
账户资金查询响应结构体		32
XTPQueryETFBaseReq		33
XTPQueryETFBaseRsp		
查询股票ETF合约基本情况-响应结构体		34
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		34
XTPQueryETFComponentRsp		
查询股票ETF合约成分股信息-响应结构体		35
XTPQueryFundTransferLogReq		
资金内转流水查询请求与响应		35
XTPQueryIPOQuotaRsp		
查询用户申购额度		36
XTPQueryIPTickerRsp		
查询当日可申购新股信息		36
XTPQueryOrderReq		
报单查询	报单查询请求-条件查询	37

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）	37
XTPQueryStkPositionRsp	
查询股票持仓情况	38
XTPQueryStructuredFundInfoReq	
查询分级基金信息结构体	38
XTPQueryTraderReq	
查询成交回报请求-查询条件	39
XTPQuoteStaticInfo	
股票行情静态信息	39
XTPRspInfoStruct	
响应信息	40
XTPSpecificTickerStruct	
指定的合约	40
XTPStructuredFundInfo	
查询分级基金信息响应结构体	41
XTPTickByTickEntrust	
逐笔委托(仅适用深交所)	41
XTPTickByTickStruct	
逐笔数据信息	42
XTPTickByTickTrade	
逐笔成交	43
XPTickerPriceInfo	
供查询的最新信息	43
XTPTradeReport	
报单成交结构体	44

Chapter 4

文件索引

4.1 文件列表

这里列出了所有文档化的文件，并附带简要说明：

demo_test_quote_api.cpp	
定义控制台测试应用程序的入口点	47
demo_test_quote_spi.h	
Demo自定义客户端行情订阅响应接口类	48
xoms_api_fund_struct.h	
定义资金划拨相关结构体类型	49
xoms_api_struct.h	
定义交易类相关数据结构	49
xquote_api_struct.h	
定义行情类相关数据结构	51
xtp_api_data_type.h	
定义兼容数据基本类型	52
xtp_api_struct.h	
定义业务数据结构	60
xtp_api_struct_common.h	
定义业务公共数据结构	60
xtp_quote_api.h	
定义行情订阅客户端接口	61

Chapter 5

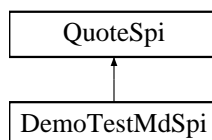
结构体说明

5.1 DemoTestMdSpi类 参考

Demo自定义行情订阅接口响应类

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

类 DemoTestMdSpi 继承关系图:



Public 成员函数

- virtual void `OnDisconnected` (int reason)
当客户端与行情后台通信连接断开
- virtual void `OnError` (XTPRI *error_info)
错误应答
- virtual void `OnSubMarketData` (XTPST *ticker, XTPRI *error_info, bool is_last)
重写订阅行情应答
- virtual void `OnUnSubMarketData` (XTPST *ticker, XTPRI *error_info, bool is_last)
重写取消订阅行情应答
- virtual void `OnDepthMarketData` (XTPMD *market_data, int64_t bid1_qty[], int32_t bid1_count, int32_t max_bid1_count, int64_t ask1_qty[], int32_t ask1_count, int32_t max_ask1_count)
重写行情通知
- virtual void `OnSubOrderBook` (XTPST *ticker, XTPRI *error_info, bool is_last)
重写订阅行情订单簿应答
- virtual void `OnUnSubOrderBook` (XTPST *ticker, XTPRI *error_info, bool is_last)
重写退订行情订单簿应答
- virtual void `OnSubTickByTick` (XTPST *ticker, XTPRI *error_info, bool is_last)
订阅逐笔行情应答
- virtual void `OnUnSubTickByTick` (XTPST *ticker, XTPRI *error_info, bool is_last)
退订逐笔行情应答
- virtual void `OnOrderBook` (XTPOB *order_book)
行情订单簿通知
- virtual void `OnTickByTick` (XTPTBT *tbt_data)

- virtual void [OnSubscribeAllMarketData](#) (XTPRI *error_info)
订阅全市场的行情应答
- virtual void [OnUnSubscribeAllMarketData](#) (XTPRI *error_info)
退订全市场的行情应答
- virtual void [OnSubscribeAllOrderBook](#) (XTPRI *error_info)
订阅全市场的行情订单簿应答
- virtual void [OnUnSubscribeAllOrderBook](#) (XTPRI *error_info)
退订全市场的行情订单簿应答
- virtual void [OnSubscribeAllTickByTick](#) (XTPRI *error_info)
订阅全市场的逐笔行情应答
- virtual void [OnUnSubscribeAllTickByTick](#) (XTPRI *error_info)
退订全市场的逐笔行情应答
- virtual void [OnQueryAllTickers](#) (XTPQSI *ticker_info, XTPRI *error_info, bool is_last)
查询可交易合约应答
- virtual void [OnQueryTickersPriceInfo](#) (XTPPTPI *ticker_info, XTPRI *error_info, bool is_last)
查询合约的最新价格信息应答

5.1.1 详细描述

Demo自定义行情订阅接口响应类

作者

中泰证券股份有限公司

日期

十月 2015

5.1.2 成员函数说明

5.1.2.1 virtual void OnTickByTick (XTPBT * tbt_data) [virtual]

逐笔行情通知

参数

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

重载 [QuoteSpi](#) .

该类的文档由以下文件生成:

- [demo_test_quote_spi.h](#)

5.2 OrderBookStruct结构体 参考

定单簿

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

成员变量

- [XTP_EXCHANGE_TYPE](#) `exchange_id`
交易所代码
- `char` [ticker](#) [[XTP_TICKER_LEN](#)]
合约代码（不包含交易所信息），不带空格，以'\0'结尾
- `double` [last_price](#)
最新价
- `int64_t` [qty](#)
数量，为总成交量
- `double` [turnover](#)
成交金额，为总成交金额
- `int64_t` [trades_count](#)
成交笔数
- `double` [bid](#) [10]
十档申买价
- `double` [ask](#) [10]
十档申卖价
- `int64_t` [bid_qty](#) [10]
十档申买量
- `int64_t` [ask_qty](#) [10]
十档申卖量
- `int64_t` [data_time](#)
时间类

5.2.1 详细描述

定单簿

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.3 QuoteApi类 参考

行情订阅接口类

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

Public 成员函数

- `virtual void` [Release](#) ()=0
- `virtual const char *` [GetTradingDay](#) ()=0
- `virtual const char *` [GetApiVersion](#) ()=0
- `virtual XTPRI *` [GetApiLastError](#) ()=0
- `virtual void` [SetUDPBufferSize](#) (uint32_t buff_size)=0
- `virtual void` [RegisterSpi](#) (QuoteSpi *spi)=0
- `virtual void` [SetHeartBeatInterval](#) (uint32_t interval)=0
- `virtual int` [SubscribeMarketData](#) (char *ticker[], int count, [XTP_EXCHANGE_TYPE](#) exchange_id)=0
- `virtual int` [UnSubscribeMarketData](#) (char *ticker[], int count, [XTP_EXCHANGE_TYPE](#) exchange_id)=0
- `virtual int` [SubscribeOrderBook](#) (char *ticker[], int count, [XTP_EXCHANGE_TYPE](#) exchange_id)=0
- `virtual int` [UnSubscribeOrderBook](#) (char *ticker[], int count, [XTP_EXCHANGE_TYPE](#) exchange_id)=0

- virtual int [SubscribeTickByTick](#) (char *ticker[], int count, [XTP_EXCHANGE_TYPE](#) exchange_id)=0
- virtual int [UnSubscribeTickByTick](#) (char *ticker[], int count, [XTP_EXCHANGE_TYPE](#) exchange_id)=0
- virtual int [SubscribeAllMarketData](#) ()=0
- virtual int [UnSubscribeAllMarketData](#) ()=0
- virtual int [SubscribeAllOrderBook](#) ()=0
- virtual int [UnSubscribeAllOrderBook](#) ()=0
- virtual int [SubscribeAllTickByTick](#) ()=0
- virtual int [UnSubscribeAllTickByTick](#) ()=0
- virtual int [Login](#) (const char *ip, int port, const char *user, const char *password, [XTP_PROTOCOL_TYPE](#) sock_type)=0
- virtual int [Logout](#) ()=0
- virtual int [QueryAllTickers](#) ([XTP_EXCHANGE_TYPE](#) exchange_id)=0
- virtual int [QueryTickersPriceInfo](#) (char *ticker[], int count, [XTP_EXCHANGE_TYPE](#) exchange_id)=0
- virtual int [QueryAllTickersPriceInfo](#) ()=0

静态 **Public** 成员函数

- static [QuoteApi](#) * [CreateQuoteApi](#) (uint8_t client_id, const char *save_file_path, [XTP_LOG_LEVEL](#) log_level=[XTP_LOG_LEVEL_DEBUG](#))

5.3.1 详细描述

行情订阅接口类

作者

中泰证券股份有限公司

日期

十月 2015

5.3.2 成员函数说明

5.3.2.1 static [QuoteApi](#)* [CreateQuoteApi](#) (uint8_t client_id, const char * save_file_path, [XTP_LOG_LEVEL](#) log_level = [XTP_LOG_LEVEL_DEBUG](#)) [static]

创建[QuoteApi](#)

参数

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

返回

创建出的[UserApi](#)

备注

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

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

获取API的系统错误

返回

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

备注

可以在调用api接口失败时调用，例如login失败时

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

获取API的发行版本号

返回

返回api发行版本号

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

获取当前交易日

返回

获取到的交易日

备注

只有登录成功后,才能得到正确的交易日

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

用户登录请求

返回

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

参数

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

<i>sock_type</i>	“1”代表TCP, “2”代表UDP
------------------	--------------------

备注

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

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

登出请求

返回

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

备注

此函数为同步阻塞式, 不需要异步等待登出, 当函数返回即可进行后续操作

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

获取当前交易日可交易合约

返回

查询是否成功, “0”表示查询成功, 非“0”表示查询不成功

参数

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

5.3.2.8 `virtual int QueryAllTickersPriceInfo () [pure virtual]`

获取所有合约的最新价格信息

返回

查询是否成功, “0”表示查询成功, 非“0”表示查询不成功

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

获取合约的最新价格信息

返回

查询是否成功, “0”表示查询成功, 非“0”表示查询不成功

参数

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

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

注册回调接口

参数

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

5.3.2.11 virtual void Release () [pure virtual]

删除接口对象本身

备注

不再使用本接口对象时,调用该函数删除接口对象

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

设置心跳检测时间间隔，单位为秒

参数

<i>interval</i>	心跳检测时间间隔，单位为秒
-----------------	---------------

备注

此函数必须在Login之前调用

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

设置采用UDP方式连接时的接收缓冲区大小

备注

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

5.3.2.14 virtual int SubscribeAllMarketData () [pure virtual]

订阅全市场的行情

返回

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

备注

需要与全市场退订行情接口配套使用

5.3.2.15 virtual int SubscribeAllOrderBook () [pure virtual]

订阅全市场的行情订单簿

返回

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

备注

需要与全市场退订行情订单簿接口配套使用

5.3.2.16 virtual int SubscribeAllTickByTick () [pure virtual]

订阅全市场的逐笔行情

返回

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

备注

需要与全市场退订逐笔行情接口配套使用

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

订阅行情。

返回

订阅接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

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

备注

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

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

订阅行情订单簿。

返回

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

参数

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

备注

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

5.3.2.19 `virtual int SubscribeTickByTick (char * ticker[], int count, XTP_EXCHANGE_TYPE exchage_id)` [pure virtual]

订阅逐笔行情。

返回

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

参数

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

备注

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

5.3.2.20 `virtual int UnSubscribeAllMarketData ()` [pure virtual]

退订全市场的行情

返回

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

备注

需要与订阅全市场行情接口配套使用

5.3.2.21 `virtual int UnSubscribeAllOrderBook ()` [pure virtual]

退订全市场的行情订单簿

返回

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

备注

需要与订阅全市场行情订单簿接口配套使用

5.3.2.22 virtual int UnSubscribeAllTickByTick () [pure virtual]

退订全市场的逐笔行情

返回

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

备注

需要与订阅全市场逐笔行情接口配套使用

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

退订行情。

返回

取消订阅接口调用是否成功，“0”表示接口调用成功，非“0”表示接口调用出错

参数

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

备注

可以一次性取消订阅同一证券交易所的多个合约，需要与订阅行情接口配套使用

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

退订行情订单簿。

返回

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

参数

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

备注

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

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

退订逐笔行情。

返回

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

参数

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

备注

可以一次性取消订阅同一证券交易所的多个合约，需要与订阅逐笔行情接口配套使用

该类的文档由以下文件生成:

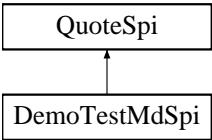
- [xtp_quote_api.h](#)

5.4 QuoteSpi类 参考

行情回调类

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

类 QuoteSpi 继承关系图:



Public 成员函数

- virtual void [OnDisconnected](#) (int reason)
- virtual void [OnError](#) (XTPRI *error_info)
- virtual void [OnSubMarketData](#) (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void [OnUnSubMarketData](#) (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void [OnDepthMarketData](#) (XTPMD *market_data, int64_t bid1_qty[], int32_t bid1_count, int32_t max_bid1_count, int64_t ask1_qty[], int32_t ask1_count, int32_t max_ask1_count)
- virtual void [OnSubOrderBook](#) (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void [OnUnSubOrderBook](#) (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void [OnOrderBook](#) (XTPOB *order_book)
- virtual void [OnSubTickByTick](#) (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void [OnUnSubTickByTick](#) (XTPST *ticker, XTPRI *error_info, bool is_last)
- virtual void [OnTickByTick](#) (XTPBT *tbt_data)
- virtual void [OnSubscribeAllMarketData](#) (XTPRI *error_info)
- virtual void [OnUnSubscribeAllMarketData](#) (XTPRI *error_info)
- virtual void [OnSubscribeAllOrderBook](#) (XTPRI *error_info)
- virtual void [OnUnSubscribeAllOrderBook](#) (XTPRI *error_info)
- virtual void [OnSubscribeAllTickByTick](#) (XTPRI *error_info)
- virtual void [OnUnSubscribeAllTickByTick](#) (XTPRI *error_info)
- virtual void [OnQueryAllTickers](#) (XTPQSI *ticker_info, XTPRI *error_info, bool is_last)
- virtual void [OnQueryTickersPriceInfo](#) (XTPPTPI *ticker_info, XTPRI *error_info, bool is_last)

5.4.1 详细描述

行情回调类

作者

中泰证券股份有限公司

日期

十月 2015

5.4.2 成员函数说明

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

深度行情通知，包含买一卖一队列

参数

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

备注

需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线

被 [DemoTestMdSpi](#) 重载.

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

当客户端与行情后台通信连接断开时，该方法被调用。

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

错误应答

参数

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

备注

此函数只有在服务器发生错误时才会调用，一般无需用户处理

被 [DemoTestMdSpi](#) 重载.

5.4.2.4 virtual void OnOrderBook (XTPOB * order_book) [inline],[virtual]

行情订单簿通知

参数

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

被 [DemoTestMdSpi](#) 重载.

5.4.2.5 virtual void OnQueryAllTickers (XTPQSI * ticker_info, XTPRI * error_info, bool is_last) [inline],[virtual]

查询可交易合约的应答

参数

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

被 [DemoTestMdSpi](#) 重载.

5.4.2.6 virtual void OnQueryTickersPricelInfo (XTPTPI * ticker_info, XTPRI * error_info, bool is_last) [inline],[virtual]

查询合约的最新价格信息应答

参数

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

被 [DemoTestMdSpi](#) 重载.

5.4.2.7 virtual void OnSubMarketData (XTPST * ticker, XTPRI * error_info, bool is_last) [inline],[virtual]

订阅行情应答

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

订阅行情订单簿应答

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

订阅全市场的行情应答

参数

<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
-------------------	---

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

订阅全市场的行情订单簿应答

参数

<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
-------------------	---

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

5.4.2.11 virtual void OnSubscribeAllTickByTick (XTPRI * error_info) [inline],[virtual]

订阅全市场的逐笔行情应答

参数

<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
-------------------	---

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

订阅逐笔行情应答

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

逐笔行情通知

参数

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

被 [DemoTestMdSpi](#) 重载.

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

退订行情应答

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

退订行情订单簿应答

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

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

退订全市场的行情应答

参数

<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当 <i>error_info</i> 为空，或者 <i>error_info.error_id</i> 为0时，表明没有错误
-------------------	---

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

退订全市场的行情订单簿应答

参数

<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
-------------------	---

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

退订全市场的逐笔行情应答

参数

<i>error_info</i>	取消订阅合约时发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
-------------------	---

备注

需要快速返回

被 [DemoTestMdSpi](#) 重载.

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

退订逐笔行情应答

参数

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

备注

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

被 [DemoTestMdSpi](#) 重载.

该类的文档由以下文件生成:

- [xtp_quote_api.h](#)

5.5 XTPFundTransferNotice结构体 参考

资金内转流水通知

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

成员变量

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

5.5.1 详细描述

资金内转流水通知

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.6 XTPFundTransferReq结构体 参考

用户资金请求

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

成员变量

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

5.6.1 详细描述

用户资金请求

该结构体的文档由以下文件生成:

- [xoms_api_fund_struct.h](#)

5.7 XTPMarketDataStruct结构体 参考

行情

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

成员变量

- [XTP_EXCHANGE_TYPE](#) `exchange_id`
交易所代码
- `char` [ticker](#) [[XTP_TICKER_LEN](#)]
合约代码（不包含交易所信息），不带空格，以'\0'结尾
- `double` [last_price](#)
最新价
- `double` [pre_close_price](#)
昨收盘
- `double` [open_price](#)
今开盘
- `double` [high_price](#)
最高价
- `double` [low_price](#)
最低价
- `double` [close_price](#)
今收盘
- `double` [pre_open_interest](#)
昨持仓量（目前未填写）
- `double` [open_interest](#)
持仓量（目前未填写）
- `double` [pre_settlement_price](#)
上次结算价（目前未填写）
- `double` [settlement_price](#)
本次结算价（目前未填写）
- `double` [upper_limit_price](#)
涨停板价（目前未填写）
- `double` [lower_limit_price](#)
跌停板价（目前未填写）
- `double` [pre_delta](#)
昨虚实度（目前未填写）
- `double` [curr_delta](#)
今虚实度（目前未填写）
- `int64_t` [data_time](#)
时间类，格式为YYYYMMDDHHMMSSsss
- `int64_t` [qty](#)
数量，为总成交量（单位股，与交易所一致）
- `double` [turnover](#)
成交金额，为总成交金额（单位元，与交易所一致）
- `double` [avg_price](#)
当日均价= $(turnover/qty)$
- `double` [bid](#) [10]
十档申买价
- `double` [ask](#) [10]
十档申卖价
- `int64_t` [bid_qty](#) [10]
十档申买量
- `int64_t` [ask_qty](#) [10]
十档申卖量
- `int64_t` [trades_count](#)

- 成交笔数
- char [ticker_status](#) [8]
当前交易状态说明
- int64_t [total_bid_qty](#)
委托买入总量
- int64_t [total_ask_qty](#)
委托卖出总量
- double [ma_bid_price](#)
加权平均委买价格
- double [ma_ask_price](#)
加权平均委卖价格
- double [ma_bond_bid_price](#)
债券加权平均委买价格
- double [ma_bond_ask_price](#)
债券加权平均委卖价格
- double [yield_to_maturity](#)
债券到期收益率
- double [iopv](#)
*ETF*净值估值
- int32_t [etf_buy_count](#)
*ETF*申购笔数
- int32_t [etf_sell_count](#)
*ETF*赎回笔数
- double [etf_buy_qty](#)
*ETF*申购数量
- double [etf_buy_money](#)
*ETF*申购金额
- double [etf_sell_qty](#)
*ETF*赎回数量
- double [etf_sell_money](#)
*ETF*赎回金额
- double [total_warrant_exec_qty](#)
权证执行的总数量
- double [warrant_lower_price](#)
权证跌停价格（元）
- double [warrant_upper_price](#)
权证涨停价格（元）
- int32_t [cancel_buy_count](#)
买入撤单笔数
- int32_t [cancel_sell_count](#)
卖出撤单笔数
- double [cancel_buy_qty](#)
买入撤单数量
- double [cancel_sell_qty](#)
卖出撤单数量
- double [cancel_buy_money](#)
买入撤单金额
- double [cancel_sell_money](#)
卖出撤单金额
- int64_t [total_buy_count](#)
买入总笔数

- `int64_t total_sell_count`
卖出总笔数
- `int32_t duration_after_buy`
买入委托成交最大等待时间
- `int32_t duration_after_sell`
卖出委托成交最大等待时间
- `int32_t num_bid_orders`
买方委托价位数
- `int32_t num_ask_orders`
卖方委托价位数
- `int32_t exec_time`
成交时间 (UA3113)
- `char is_market_closed [4]`
闭市标志 (UA103/UA104)
- `double total_position`
合约持仓量 (UA103)
- `double pe_ratio1`
市盈率1 (UA103)
- `double pe_ratio2`
市盈率2 (UA103)

5.7.1 详细描述

行情

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.8 XTPOrderCancelInfo结构体 参考

撤单失败响应消息

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

成员变量

- `uint64_t order_cancel_xtp_id`
撤单XTPID
- `uint64_t order_xtp_id`
原始订单XTPID

5.8.1 详细描述

撤单失败响应消息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.9 XTPOrderInfo结构体 参考

报单响应结构体

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

成员变量

- `uint64_t order_xtp_id`
XTP系统订单ID, 在XTP系统中唯一
- `uint32_t order_client_id`
报单引用, 用户自定义
- `uint32_t order_cancel_client_id`
报单操作引用, 用户自定义 (暂未使用)
- `uint64_t order_cancel_xtp_id`
撤单在XTP系统中的id, 在XTP系统中唯一
- `char ticker [XTP_TICKER_LEN]`
合约代码
- `XTP_MARKET_TYPE market`
交易市场
- `double price`
价格
- `int64_t quantity`
数量, 此订单的报单数量
- `XTP_PRICE_TYPE price_type`
报单价格条件
- `XTP_SIDE_TYPE side`
买卖方向
- `XTP_BUSINESS_TYPE business_type`
业务类型
- `int64_t qty_traded`
今成交数量, 为此订单累计成交数量
- `int64_t qty_left`
剩余数量, 当撤单成功时, 表示撤单数量
- `int64_t insert_time`
委托时间, 格式为YYYYMMDDHHMMSSsss
- `int64_t update_time`
最后修改时间, 格式为YYYYMMDDHHMMSSsss
- `int64_t cancel_time`
撤销时间, 格式为YYYYMMDDHHMMSSsss
- `double trade_amount`
成交金额, 为此订单的成交总金额
- `char order_local_id [XTP_LOCAL_ORDER_LEN]`
本地报单编号 OMS生成的单号, 不等同于order_xtp_id, 为服务器传到报盘的单号
- `XTP_ORDER_STATUS_TYPE order_status`
报单状态, 订单响应中没有部分成交状态的推送, 在查询订单结果中, 会有部分成交状态
- `XTP_ORDER_SUBMIT_STATUS_TYPE order_submit_status`
报单提交状态, OMS内部使用, 用户无需关心
- `TXTPOrderTypeType order_type`
报单类型

5.9.1 详细描述

报单响应结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.10 XTPOrderInsertInfo结构体 参考

新订单请求

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

成员变量

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

5.10.1 详细描述

新订单请求

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.11 XTPQueryAssetRsp结构体 参考

账户资金查询响应结构体

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


成员变量

- double [total_asset](#)
总资产(=可用资金 + 证券资产 (目前为0) + 预扣的资金)
- double [buying_power](#)
可用资金
- double [security_asset](#)
证券资产 (保留字段, 目前为0)
- double [fund_buy_amount](#)
累计买入成交证券占用资金
- double [fund_buy_fee](#)
累计买入成交交易费用
- double [fund_sell_amount](#)
累计卖出成交证券所得资金
- double [fund_sell_fee](#)
累计卖出成交交易费用
- double [withholding_amount](#)
XTP系统预扣的资金 (包括购买股票时预扣的交易资金+预扣手续费)
- [XTP_ACCOUNT_TYPE](#) [account_type](#)
账户类型
- uint64_t [unknown](#) [43]
(保留字段)

5.11.1 详细描述

账户资金查询响应结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.12 XTPQueryETFBASEReq结构体 参考

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

成员变量

- [XTP_MARKET_TYPE](#) [market](#)
交易市场
- char [ticker](#) [[XTP_TICKER_LEN](#)]
ETF买卖代码

5.12.1 详细描述

查询股票ETF合约基本情况-请求结构体, 请求参数为多条件参数:1,不填则返回所有市场的ETF合约信息。
2,只填写market,返回该交易市场下结果 3,填写market及ticker参数,只返回该etf信息。

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.13 XTPQueryETFBaseRsp结构体 参考

查询股票ETF合约基本情况-响应结构体

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

成员变量

- [XTP_MARKET_TYPE market](#)
交易市场
- [char etf \[XTP_TICKER_LEN\]](#)
*etf*代码,买卖,申赎统一使用该代码
- [char subscribe_redemption_ticker \[XTP_TICKER_LEN\]](#)
*etf*申购赎回代码
- [int32_t unit](#)
最小申购赎回单位对应的*ETF*份数,例如上证"50*ETF*"就是900000
- [int32_t subscribe_status](#)
是否允许申购,1-允许,0-禁止
- [int32_t redemption_status](#)
是否允许赎回,1-允许,0-禁止
- [double max_cash_ratio](#)
最大现金替代比例,小于1的数值 *TODO* 是否采用*double*
- [double estimate_amount](#)
*T*日预估金额
- [double cash_component](#)
*T-X*日现金差额
- [double net_value](#)
基金单位净值
- [double total_amount](#)
最小申赎单位净值总金额=*net_value***unit*

5.13.1 详细描述

查询股票ETF合约基本情况-响应结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.14 XTPQueryETFComponentReq结构体 参考

查询股票ETF合约成分股信息-请求结构体,请求参数为:交易市场+ETF买卖代码

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

成员变量

- [XTP_MARKET_TYPE market](#)
交易市场
- [char ticker \[XTP_TICKER_LEN\]](#)
*ETF*买卖代码

5.14.1 详细描述

查询股票ETF合约成分股信息-请求结构体,请求参数为:交易市场+ETF买卖代码

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.15 XTPQueryETFComponentRsp结构体 参考

查询股票ETF合约成分股信息-响应结构体

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

成员变量

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

5.15.1 详细描述

查询股票ETF合约成分股信息-响应结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.16 XTPQueryFundTransferLogReq结构体 参考

资金内转流水查询请求与响应

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

成员变量

- [uint64_t](#) [serial_id](#)
资金内转编号

5.16.1 详细描述

资金内转流水查询请求与响应

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.17 XTPQueryIPOQuotaRsp结构体 参考

查询用户申购额度

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

成员变量

- [XTP_MARKET_TYPE market](#)
交易市场
- [int32_t quantity](#)
可申购额度

5.17.1 详细描述

查询用户申购额度

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.18 XTPQueryIPOTickerRsp结构体 参考

查询当日可申购新股信息

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

成员变量

- [XTP_MARKET_TYPE market](#)
交易市场
- [char ticker \[XTP_TICKER_LEN\]](#)
申购代码
- [char ticker_name \[XTP_TICKER_NAME_LEN\]](#)
申购股票名称
- [double price](#)
申购价格
- [int32_t unit](#)
申购单元
- [int32_t qty_upper_limit](#)
最大允许申购数量

5.18.1 详细描述

查询当日可申购新股信息

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.19 XTPQueryOrderReq结构体 参考

报单查询 // 报单查询请求-条件查询

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

成员变量

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

5.19.1 详细描述

报单查询 // 报单查询请求-条件查询

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.20 XTPQueryReportByExecIdReq结构体 参考

成交回报查询 // 查询成交报告请求-根据执行编号查询 (保留字段)

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

成员变量

- uint64_t [order_xtp_id](#)
XTP 订单系统 ID.
- char [exec_id](#) [XTP_EXEC_ID_LEN]
成交执行编号

5.20.1 详细描述

成交回报查询 // 查询成交报告请求-根据执行编号查询 (保留字段)

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.21 XTPQueryStkPositionRsp结构体 参考

查询股票持仓情况

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

成员变量

- char [ticker](#) [XTP_TICKER_LEN]
证券代码
- char [ticker_name](#) [XTP_TICKER_NAME_LEN]
证券名称
- [XTP_MARKET_TYPE](#) market
交易市场
- int64_t [total_qty](#)
总持仓
- int64_t [sellable_qty](#)
可卖持仓
- double [avg_price](#)
持仓成本
- double [unrealized_pnl](#)
浮动盈亏（保留字段）
- int64_t [yesterday_position](#)
昨日持仓
- int64_t [purchase_redeemable_qty](#)
今日申购赎回数量（申购和赎回数量不可能同时存在，因此可以共用一个字段）
- uint64_t [unknown](#) [50]
(保留字段)

5.21.1 详细描述

查询股票持仓情况

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.22 XTPQueryStructuredFundInfoReq结构体 参考

查询分级基金信息结构体

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

成员变量

- [XTP_EXCHANGE_TYPE](#) exchange_id
交易所代码，不可为空
- char [sf_ticker](#) [XTP_TICKER_LEN]
分级基金母基金代码，可以为空，如果为空，则默认查询所有的分级基金

5.22.1 详细描述

查询分级基金信息结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.23 XTPQueryTraderReq结构体 参考

查询成交回报请求-查询条件

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

成员变量

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

5.23.1 详细描述

查询成交回报请求-查询条件

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.24 XTPQuoteStaticInfo结构体 参考

股票行情静态信息

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

成员变量

- [XTP_EXCHANGE_TYPE exchange_id](#)
交易所代码
- [char ticker \[XTP_TICKER_LEN\]](#)
合约代码（不包含交易所信息），不带空格，以'0'结尾
- [char ticker_name \[XTP_TICKER_NAME_LEN\]](#)
合约名称
- [XTP_TICKER_TYPE ticker_type](#)
合约类型
- [double pre_close_price](#)
昨收盘
- [double upper_limit_price](#)
涨停板价

- double [lower_limit_price](#)
跌停板价
- double [price_tick](#)
最小变动价位
- int32_t [buy_qty_unit](#)
合约最小交易量(买)
- int32_t [sell_qty_unit](#)
合约最小交易量(卖)

5.24.1 详细描述

股票行情静态信息

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.25 XTPRspInfoStruct结构体 参考

响应信息

```
#include <xtp_api_struct_common.h>
```

成员变量

- int32_t [error_id](#)
错误代码
- char [error_msg](#) [XTP_ERR_MSG_LEN]
错误信息

5.25.1 详细描述

响应信息

该结构体的文档由以下文件生成:

- [xtp_api_struct_common.h](#)

5.26 XTPSpecificTickerStruct结构体 参考

指定的合约

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

成员变量

- [XTP_EXCHANGE_TYPE](#) [exchange_id](#)
交易所代码
- char [ticker](#) [XTP_TICKER_LEN]
合约代码（不包含交易所信息）例如“600000”，不带空格，以‘0’结尾

5.26.1 详细描述

指定的合约

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.27 XTPStructuredFundInfo结构体 参考

查询分级基金信息响应结构体

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

成员变量

- [XTP_EXCHANGE_TYPE](#) exchange_id
交易所代码
- [char sf_ticker](#) [XTP_TICKER_LEN]
分级基金母基金代码
- [char sf_ticker_name](#) [XTP_TICKER_NAME_LEN]
分级基金母基金名称
- [char ticker](#) [XTP_TICKER_LEN]
分级基金子基金代码
- [char ticker_name](#) [XTP_TICKER_NAME_LEN]
分级基金子基金名称
- [XTP_SPLIT_MERGE_STATUS](#) split_merge_status
基金允许拆分合并状态
- [uint32_t ratio](#)
拆分合并比例
- [uint32_t min_split_qty](#)
最小拆分数量
- [uint32_t min_merge_qty](#)
最小合并数量
- [double net_price](#)
基金净值

5.27.1 详细描述

查询分级基金信息响应结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

5.28 XTPTickByTickEntrust结构体 参考

逐笔委托(仅适用深交所)

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

成员变量

- `int32_t channel_no`
频道代码
- `int64_t seq`
委托序号(在同一个`channel_no`内唯一, 从1开始连续)
- `double price`
委托价格
- `int64_t qty`
委托数量
- `char side`
'1':买; '2':卖; 'G':借入; 'F':出借
- `char ord_type`
订单类别: '1': 市价; '2': 限价; '3': 本方最优

5.28.1 详细描述

逐笔委托(仅适用深交所)

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.29 XPTickByTickStruct结构体 参考

逐笔数据信息

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

成员变量

- `XTP_EXCHANGE_TYPE exchange_id`
交易所代码
- `char ticker[XTP_TICKER_LEN]`
合约代码(不包含交易所信息), 不带空格, 以'\0'结尾
- `int64_t seq`
预留
- `int64_t data_time`
委托时间 or 成交时间
- `XTP_TBT_TYPE type`
委托 or 成交
- `union {`
 `XPTickByTickEntrust entrust`
 `XPTickByTickTrade trade`
};

5.29.1 详细描述

逐笔数据信息

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.30 XTPTickByTickTrade结构体 参考

逐笔成交

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

成员变量

- `int32_t channel_no`
频道代码
- `int64_t seq`
委托序号(在同一个`channel_no`内唯一, 从1开始连续)
- `double price`
成交价格
- `int64_t qty`
成交量
- `double money`
成交金额(仅适用上交所)
- `int64_t bid_no`
买方订单号
- `int64_t ask_no`
卖方订单号
- `char trade_flag`

5.30.1 详细描述

逐笔成交

5.30.2 结构体成员变量说明

5.30.2.1 char trade_flag

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

该结构体的文档由以下文件生成:

- `xquote_api_struct.h`

5.31 XTPTickerPriceInfo结构体 参考

供查询的最新信息

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

成员变量

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

5.31.1 详细描述

供查询的最新信息

该结构体的文档由以下文件生成:

- [xquote_api_struct.h](#)

5.32 XTPTradeReport结构体 参考

报单成交结构体

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

成员变量

- `uint64_t order_xtp_id`
XTP系统订单ID, 此成交回报相关的订单ID, 在XTP系统中唯一
- `uint32_t order_client_id`
报单引用
- `char ticker [XTP_TICKER_LEN]`
合约代码
- `XTP_MARKET_TYPE market`
交易市场
- `uint64_t local_order_id`
订单号, 引入XTPID后, 该字段实际和order_xtp_id重复。接口中暂时保留。
- `char exec_id [XTP_EXEC_ID_LEN]`
成交编号, 深交所唯一, 上交所每笔交易唯一, 当发现2笔成交回报拥有相同的exec_id, 则可以认为此笔交易自成交
- `double price`
价格, 此次成交的价格
- `int64_t quantity`
数量, 此次成交的数量, 不是累计数量
- `int64_t trade_time`
成交时间, 格式为YYYYMMDDHHMMSSsss
- `double trade_amount`
成交金额, 此次成交的总金额 = price*quantity
- `uint64_t report_index`
成交序号 - 回报记录号, 每个交易所唯一, report_index+market字段可以组成唯一标识表示成交回报
- `char order_exch_id [XTP_ORDER_EXCH_LEN]`
报单编号 - 交易所单号, 上交所为空, 深交所所有此字段
- `TXTPTradeType trade_type`
成交类型 - 成交回报中的执行类型
- `XTP_SIDE_TYPE side`
买卖方向
- `XTP_BUSINESS_TYPE business_type`
业务类型
- `char branch_pbu [XTP_BRANCH_PBU_LEN]`
交易所交易员代码

5.32.1 详细描述

报单成交结构体

该结构体的文档由以下文件生成:

- [xoms_api_struct.h](#)

Chapter 6

文件说明

6.1 demo_test_quote_api.cpp 文件参考

定义控制台测试应用程序的入口点

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

函数

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

变量

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

6.1.1 详细描述

定义控制台测试应用程序的入口点

作者

中泰证券股份有限公司

6.1.2 函数说明

6.1.2.1 int main ()

int main() {

测试Demo入口函数

// 初始化UserApi

user_api_pointer = XTP::API::QuoteApi::CreateQuoteApi(client_id,filepath); // 创建UserApi

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

user_api_pointer->SetUDPBufferSize(128); //设置UDP接收缓冲区大小, 单位为MB

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

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

int loginResult = user_api_pointer->Login("127.0.0.1", 6666, username, password, XTP_PROTOCOL_UDP); //采用UDP方式登陆行情订阅服务器

if (loginResult == 0)

{ //登录成功

user_api_pointer->SubscribeMarketData(ppTicker, ticker_count, XTP_EXCHANGE_SZ); //订阅股票行情

}

return 0;

}

6.1.3 变量说明

6.1.3.1 int ticker_count = 1

行情订阅数量

6.2 demo_test_quote_spi.h 文件参考

Demo自定义客户端行情订阅响应接口类

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

结构体

- class DemoTestMdSpi

Demo自定义行情订阅接口响应类

6.2.1 详细描述

Demo自定义客户端行情订阅响应接口类

作者

中泰证券股份有限公司

6.3 xoms_api_fund_struct.h 文件参考

定义资金划拨相关结构体类型

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

结构体

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

宏定义

- #define [XTP_ACCOUNT_PASSWORD_LEN](#) 64
用户资金账户密码字符串数组长度

类型定义

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

6.3.1 详细描述

定义资金划拨相关结构体类型

作者

中泰证券股份有限公司

6.4 xoms_api_struct.h 文件参考

定义交易类相关数据结构

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

结构体

- struct [XTPOrderInsertInfo](#)
新订单请求
- struct [XTPOrderCancelInfo](#)
撤单失败响应消息
- struct [XTPOrderInfo](#)
报单响应结构体

- struct [XTPTradeReport](#)
报单成交结构体
- struct [XTPQueryOrderReq](#)
报单查询 // 报单查询请求-条件查询
- struct [XTPQueryReportByExeclIdReq](#)
成交回报查询 // 查询成交报告请求-根据执行编号查询（保留字段）
- struct [XTPQueryTraderReq](#)
查询成交回报请求-查询条件
- struct [XTPQueryAssetRsp](#)
账户资金查询响应结构体
- struct [XTPQueryStkPositionRsp](#)
查询股票持仓情况
- struct [XTPFundTransferNotice](#)
资金内转流水通知
- struct [XTPQueryFundTransferLogReq](#)
资金内转流水查询请求与响应
- struct [XTPQueryStructuredFundInfoReq](#)
查询分级基金信息结构体
- struct [XTPStructuredFundInfo](#)
查询分级基金信息响应结构体
- struct [XTPQueryETFBaseReq](#)
- struct [XTPQueryETFBaseRsp](#)
查询股票ETF合约基本情况-响应结构体
- struct [XTPQueryETFComponentReq](#)
查询股票ETF合约成分股信息-请求结构体,请求参数为:交易市场+ETF买卖代码
- struct [XTPQueryETFComponentRsp](#)
查询股票ETF合约成分股信息-响应结构体
- struct [XTPQueryIPOTickerRsp](#)
查询当日可申购新股信息
- struct [XTPQueryIPOQuotaRsp](#)
查询用户申购额度

类型定义

- typedef struct [XTPOrderInfo](#) [XTPQueryOrderRsp](#)
报单查询响应结构体
- typedef struct [XTPTradeReport](#) [XTPQueryTradeRsp](#)
成交回报查询响应结构体
- typedef struct [XTPFundTransferNotice](#) [XTPFundTransferLog](#)
资金内转流水记录结构体
- typedef struct [XTPQueryETFBaseRsp](#) [XTPQueryETFBaseRsp](#)
查询股票ETF合约基本情况-响应结构体
- typedef struct [XTPQueryETFComponentReq](#) [XTPQueryETFComponentReq](#)
查询股票ETF合约成分股信息-请求结构体,请求参数为:交易市场+ETF买卖代码

6.4.1 详细描述

定义交易类相关数据结构

作者

中泰证券股份有限公司

6.5 xquote_api_struct.h 文件参考

定义行情类相关数据结构

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

结构体

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

类型定义

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

6.5.1 详细描述

定义行情类相关数据结构

作者

中泰证券股份有限公司

6.6 xtp_api_data_type.h 文件参考

定义兼容数据基本类型

宏定义

- `#define XTP_VERSION_LEN 16`
存放版本号的字符串长度
- `#define XTP_TRADING_DAY_LEN 9`
可交易日字符串长度
- `#define XTP_TICKER_LEN 16`
存放证券代码的字符串长度
- `#define XTP_TICKER_NAME_LEN 64`
存放证券名称的字符串长度
- `#define XTP_LOCAL_ORDER_LEN 11`
本地报单编号的字符串长度
- `#define XTP_ORDER_EXCH_LEN 17`
交易所单号的字符串长度
- `#define XTP_EXEC_ID_LEN 18`
成交执行编号的字符串长度
- `#define XTP_BRANCH_PBU_LEN 7`
交易所交易员代码字符串长度
- `#define XTP_ACCOUNT_NAME_LEN 16`
用户资金账户的字符串长度
- `#define XTP_TRDT_COMMON '0'`
普通成交
- `#define XTP_TRDT_CASH '1'`
现金替代
- `#define XTP_TRDT_PRIMARY '2'`
一级市场成交
- `#define XTP_ORDT_Normal '0'`
正常
- `#define XTP_ORDT_DeriveFromQuote '1'`
报价衍生
- `#define XTP_ORDT_DeriveFromCombination '2'`
组合衍生
- `#define XTP_ORDT_Combination '3'`
组合报单
- `#define XTP_ORDT_ConditionalOrder '4'`
条件单
- `#define XTP_ORDT_Swap '5'`
互换单

类型定义

- `typedef char XTPVersionType[XTP_VERSION_LEN]`
版本号类型
- `typedef enum XTP_LOG_LEVEL XTP_LOG_LEVEL`
`XTP_LOG_LEVEL`是日志输出级别类型
- `typedef enum XTP_PROTOCOL_TYPE XTP_PROTOCOL_TYPE`

- XTP_PROTOCOL_TYPE*是通讯传输协议方式
- typedef enum [XTP_EXCHANGE_TYPE](#) [XTP_EXCHANGE_TYPE](#)
*XTP_EXCHANGE_TYPE*是交易所类型
- typedef enum [XTP_MARKET_TYPE](#) [XTP_MARKET_TYPE](#)
*XTP_MARKET_TYPE*市场类型
- typedef enum [XTP_PRICE_TYPE](#) [XTP_PRICE_TYPE](#)
*XTP_PRICE_TYPE*是价格类型
- typedef enum [XTP_SIDE_TYPE](#) [XTP_SIDE_TYPE](#)
*XTP_SIDE_TYPE*是买卖方向类型
- typedef enum [XTP_ORDER_ACTION_STATUS_TYPE](#) [XTP_ORDER_ACTION_STATUS_TYPE](#)
*XTP_ORDER_ACTION_STATUS_TYPE*是报单操作状态类型
- typedef enum [XTP_ORDER_STATUS_TYPE](#) [XTP_ORDER_STATUS_TYPE](#)
*XTP_ORDER_STATUS_TYPE*是报单状态类型
- typedef enum [XTP_ORDER_SUBMIT_STATUS_TYPE](#) [XTP_ORDER_SUBMIT_STATUS_TYPE](#)
*XTP_ORDER_SUBMIT_STATUS_TYPE*是报单提交状态类型
- typedef enum [XTP_TE_RESUME_TYPE](#) [XTP_TE_RESUME_TYPE](#)
*XTP_TE_RESUME_TYPE*是公有流（订单响应、成交回报）重传方式
- typedef enum [ETF_REPLACE_TYPE](#) [ETF_REPLACE_TYPE](#)
*ETF_REPLACE_TYPE*现金替代标识定义
- typedef enum [XTP_TICKER_TYPE](#) [XTP_TICKER_TYPE](#)
*XTP_TICKER_TYPE*证券类型
- typedef enum [XTP_BUSINESS_TYPE](#) [XTP_BUSINESS_TYPE](#)
*XTP_BUSINESS_TYPE*证券业务类型
- typedef enum [XTP_ACCOUNT_TYPE](#) [XTP_ACCOUNT_TYPE](#)
*XTP_ACCOUNT_TYPE*账户类型
- typedef enum [XTP_FUND_TRANSFER_TYPE](#) [XTP_FUND_TRANSFER_TYPE](#)
*XTP_FUND_TRANSFER_TYPE*是资金流转方向类型
- typedef enum [XTP_FUND_OPER_STATUS](#) [XTP_FUND_OPER_STATUS](#)
*XTP_FUND_OPER_STATUS*柜台资金操作结果
- typedef enum [XTP_SPLIT_MERGE_STATUS](#) [XTP_SPLIT_MERGE_STATUS](#)
*XTP_SPLIT_MERGE_STATUS*是一个基金当天拆分合并状态类型
- typedef enum [XTP_TBT_TYPE](#) [XTP_TBT_TYPE](#)
*XTP_TBT_TYPE*是一个逐笔回报类型
- typedef char [TXTPTradeTypeType](#)
*TXTPTradeTypeType*是成交类型类型
- typedef char [TXTPOrderTypeType](#)
*TXTPOrderTypeType*是报单类型类型

枚举

- enum [XTP_LOG_LEVEL](#) {
[XTP_LOG_LEVEL_FATAL](#), [XTP_LOG_LEVEL_ERROR](#), [XTP_LOG_LEVEL_WARNING](#), [XTP_LOG_LEVEL_INFO](#),
[XTP_LOG_LEVEL_DEBUG](#), [XTP_LOG_LEVEL_TRACE](#) }
- XTP_LOG_LEVEL*是日志输出级别类型
- enum [XTP_PROTOCOL_TYPE](#) { [XTP_PROTOCOL_TCP](#) = 1, [XTP_PROTOCOL_UDP](#) }
- XTP_PROTOCOL_TYPE*是通讯传输协议方式
- enum [XTP_EXCHANGE_TYPE](#) { [XTP_EXCHANGE_SH](#) = 1, [XTP_EXCHANGE_SZ](#), [XTP_EXCHANGE_UNKNOWN](#) }
- XTP_EXCHANGE_TYPE*是交易所类型

- enum `XTP_MARKET_TYPE` { `XTP_MKT_INIT` = 0, `XTP_MKT_SZ_A` = 1, `XTP_MKT_SH_A`, `XTP_MKT_U`↵
`NKNOWN` }
`XTP_MARKET_TYPE`市场类型
- enum `XTP_PRICE_TYPE` {
`XTP_PRICE_LIMIT` = 1, `XTP_PRICE_BEST_OR_CANCEL`, `XTP_PRICE_BEST5_OR_LIMIT`, `XTP_PRIC`↵
`E_BEST5_OR_CANCEL`,
`XTP_PRICE_ALL_OR_CANCEL`, `XTP_PRICE_FORWARD_BEST`, `XTP_PRICE_REVERSE_BEST_LIMIT`,
`XTP_PRICE_TYPE_UNKNOWN` }
`XTP_PRICE_TYPE`是价格类型
- enum `XTP_SIDE_TYPE` {
`XTP_SIDE_BUY` = 1, `XTP_SIDE_SELL`, `XTP_SIDE_BUY_OPEN`, `XTP_SIDE_SELL_OPEN`,
`XTP_SIDE_BUY_CLOSE`, `XTP_SIDE_SELL_CLOSE`, `XTP_SIDE_PURCHASE`, `XTP_SIDE_REDEMPTI`↵
`ON`,
`XTP_SIDE_SPLIT`, `XTP_SIDE_MERGE`, `XTP_SIDE_UNKNOWN` }
`XTP_SIDE_TYPE`是买卖方向类型
- enum `XTP_ORDER_ACTION_STATUS_TYPE` { `XTP_ORDER_ACTION_STATUS_SUBMITTED` = 1, `XT`↵
`P_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_P`↵
`ARTTRADEDQUEUEING`, `XTP_ORDER_STATUS_PARTTRADEDNOTQUEUEING`,
`XTP_ORDER_STATUS_NOTRADEQUEUEING`, `XTP_ORDER_STATUS_CANCELED`, `XTP_ORDER_S`↵
`TATUS_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_STATU`↵
`S_CANCEL_SUBMITTED`,
`XTP_ORDER_SUBMIT_STATUS_CANCEL_REJECTED`, `XTP_ORDER_SUBMIT_STATUS_CANCEL_A`↵
`CCEPTED` }
`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_MU`↵
`ST`, `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_TIC`↵
`KER_TYPE_BOND`,
`XTP_TICKER_TYPE_UNKNOWN` }
`XTP_TICKER_TYPE`证券类型
- enum `XTP_BUSINESS_TYPE` {
`XTP_BUSINESS_TYPE_CASH` = 0, `XTP_BUSINESS_TYPE_IPOS`, `XTP_BUSINESS_TYPE_REPO`, `XT`↵
`P_BUSINESS_TYPE ETF`,
`XTP_BUSINESS_TYPE_MARGIN`, `XTP_BUSINESS_TYPE_DESIGNATION`, `XTP_BUSINESS_TYPE_A`↵
`LLOTMENT`, `XTP_BUSINESS_TYPE_STRUCTURED_FUND_PURCHASE_REDEMPTION`,
`XTP_BUSINESS_TYPE_STRUCTURED_FUND_SPLIT_MERGE`, `XTP_BUSINESS_TYPE_MONEY_FU`↵
`ND`, `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_ACCO`↵
`UNT_DERIVE`, `XTP_ACCOUNT_UNKNOWN` }
`XTP_ACCOUNT_TYPE`账户类型
- enum `XTP_FUND_TRANSFER_TYPE` { `XTP_FUND_TRANSFER_OUT` = 0, `XTP_FUND_TRANSFER_IN`,
`XTP_FUND_TRANSFER_UNKNOWN` }
`XTP_FUND_TRANSFER_TYPE`是资金流转方向类型

- enum `XTP_FUND_OPER_STATUS` {
`XTP_FUND_OPER_PROCESSING` = 0, `XTP_FUND_OPER_SUCCESS`, `XTP_FUND_OPER_FAILED`, `XTP_FUND_OPER_SUBMITTED`,
`XTP_FUND_OPER_UNKNOWN` }
`XTP_FUND_OPER_STATUS`柜台资金操作结果
- enum `XTP_SPLIT_MERGE_STATUS` { `XTP_SPLIT_MERGE_STATUS_ALLOW` = 0, `XTP_SPLIT_MERGE_STATUS_ONLY_SPLIT`, `XTP_SPLIT_MERGE_STATUS_ONLY_MERGE`, `XTP_SPLIT_MERGE_STATUS_FORBIDDEN` }
`XTP_SPLIT_MERGE_STATUS`是一个基金当天拆分合并状态类型
- enum `XTP_TBT_TYPE` { `XTP_TBT_ENTRUST` = 1, `XTP_TBT_TRADE` = 2 }
`XTP_TBT_TYPE`是一个逐笔回报类型

6.6.1 详细描述

定义兼容数据基本类型

作者

中泰证券股份有限公司

6.6.2 枚举类型说明

6.6.2.1 enum ETF_REPLACE_TYPE

ETF_REPLACE_TYPE现金替代标识定义

枚举值

ERT_CASH_FORBIDDEN 禁止现金替代

ERT_CASH_OPTIONAL 可以现金替代

ERT_CASH_MUST 必须现金替代

EPT_INVALID 无效值

6.6.2.2 enum XTP_ACCOUNT_TYPE

XTP_ACCOUNT_TYPE账户类型

枚举值

XTP_ACCOUNT_NORMAL 普通账户

XTP_ACCOUNT_CREDIT 信用账户

XTP_ACCOUNT_DERIVE 衍生品账户

XTP_ACCOUNT_UNKNOWN 未知账户类型

6.6.2.3 enum XTP_BUSINESS_TYPE

XTP_BUSINESS_TYPE证券业务类型

枚举值

XTP_BUSINESS_TYPE_CASH 普通股票业务（股票买卖，ETF买卖等）

XTP_BUSINESS_TYPE_IPOS 新股申购业务（对应的price type需选择限价类型）
XTP_BUSINESS_TYPE_REPO 回购业务（对应的price type填为限价，side填为卖）
XTP_BUSINESS_TYPE ETF ETF申赎业务（暂未支持）
XTP_BUSINESS_TYPE_MARGIN 融资融券业务（暂未支持）
XTP_BUSINESS_TYPE_DESIGNATION 转托管（未支持）
XTP_BUSINESS_TYPE_ALLOTMENT 配股业务（对应的price type需选择限价类型,side填为买）
XTP_BUSINESS_TYPE_STRUCTURED_FUND_PURCHASE_REDEMPTION 分级基金申赎业务
XTP_BUSINESS_TYPE_STRUCTURED_FUND_SPLIT_MERGE 分级基金拆分合并业务
XTP_BUSINESS_TYPE_MONEY_FUND 货币基金业务（暂未支持）
XTP_BUSINESS_TYPE_UNKNOWN 未知类型

6.6.2.4 enum XTP_EXCHANGE_TYPE

XTP_EXCHANGE_TYPE是交易所类型

枚举值

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

6.6.2.5 enum XTP_FUND_OPER_STATUS

XTP_FUND_OPER_STATUS柜台资金操作结果

枚举值

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

6.6.2.6 enum XTP_FUND_TRANSFER_TYPE

XTP_FUND_TRANSFER_TYPE是资金流转方向类型

枚举值

XTP_FUND_TRANSFER_OUT 转出 从XTP转出到柜台
XTP_FUND_TRANSFER_IN 转入 从柜台转入XTP
XTP_FUND_TRANSFER_UNKNOWN 未知类型

6.6.2.7 enum XTP_LOG_LEVEL

XTP_LOG_LEVEL是日志输出级别类型

枚举值

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

6.6.2.8 enum XTP_MARKET_TYPE

XTP_MARKET_TYPE市场类型

枚举值

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

6.6.2.9 enum XTP_ORDER_ACTION_STATUS_TYPE

XTP_ORDER_ACTION_STATUS_TYPE是报单操作状态类型

枚举值

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

6.6.2.10 enum XTP_ORDER_STATUS_TYPE

XTP_ORDER_STATUS_TYPE是报单状态类型

枚举值

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

6.6.2.11 enum XTP_ORDER_SUBMIT_STATUS_TYPE

XTP_ORDER_SUBMIT_STATUS_TYPE是报单提交状态类型

枚举值

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

6.6.2.12 enum XTP_PRICE_TYPE

XTP_PRICE_TYPE是价格类型

枚举值

XTP_PRICE_LIMIT 限价单-沪深（除普通股票业务外，其余业务均使用此种类型）
XTP_PRICE_BEST_OR_CANCEL 即时成交剩余转撤销，市价单-深
XTP_PRICE_BEST5_OR_LIMIT 最优五档即时成交剩余转限价，市价单-沪
XTP_PRICE_BEST5_OR_CANCEL 最优5档即时成交剩余转撤销，市价单-沪深
XTP_PRICE_ALL_OR_CANCEL 全部成交或撤销,市价单-深
XTP_PRICE_FORWARD_BEST 本方最优，市价单-深
XTP_PRICE_REVERSE_BEST_LIMIT 对方最优剩余转限价，市价单-深
XTP_PRICE_TYPE_UNKNOWN 未知或者无效价格类型

6.6.2.13 enum XTP_PROTOCOL_TYPE

XTP_PROTOCOL_TYPE是通讯传输协议方式

枚举值

XTP_PROTOCOL_TCP 采用TCP方式传输
XTP_PROTOCOL_UDP 采用UDP方式传输（目前暂未支持）

6.6.2.14 enum XTP_SIDE_TYPE

XTP_SIDE_TYPE是买卖方向类型

枚举值

XTP_SIDE_BUY 买（新股申购、ETF买等）
XTP_SIDE_SELL 卖（逆回购）
XTP_SIDE_BUY_OPEN 买开（暂未支持）
XTP_SIDE_SELL_OPEN 卖开（暂未支持）
XTP_SIDE_BUY_CLOSE 买平（暂未支持）
XTP_SIDE_SELL_CLOSE 卖平（暂未支持）

XTP_SIDE_PURCHASE 申购
XTP_SIDE_REDEMPTION 赎回
XTP_SIDE_SPLIT 拆分
XTP_SIDE_MERGE 合并
XTP_SIDE_UNKNOWN 未知或者无效买卖方向

6.6.2.15 enum XTP_SPLIT_MERGE_STATUS

XTP_SPLIT_MERGE_STATUS是一个基金当天拆分合并状态类型

枚举值

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

6.6.2.16 enum XTP_TBT_TYPE

XTP_TBT_TYPE是一个逐笔回报类型

枚举值

XTP_TBT_ENTRUST 逐笔委托
XTP_TBT_TRADE 逐笔成交

6.6.2.17 enum XTP_TE_RESUME_TYPE

XTP_TE_RESUME_TYPE是公有流（订单响应、成交回报）重传方式

枚举值

XTP_TERT_RESTART 从本交易日开始重传
XTP_TERT_RESUME 从上次收到的续传（暂未支持）
XTP_TERT_QUICK 只传送登录后公有流（订单响应、成交回报）的内容

6.6.2.18 enum XTP_TICKER_TYPE

XTP_TICKER_TYPE证券类型

枚举值

XTP_TICKER_TYPE_STOCK 普通股票
XTP_TICKER_TYPE_INDEX 指数
XTP_TICKER_TYPE_FUND 基金
XTP_TICKER_TYPE_BOND 债券
XTP_TICKER_TYPE_UNKNOWN 未知类型

6.7 xtp_api_struct.h 文件参考

定义业务数据结构

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

6.7.1 详细描述

定义业务数据结构

作者

中泰证券股份有限公司

6.8 xtp_api_struct_common.h 文件参考

定义业务公共数据结构

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

结构体

- struct [XTPRspInfoStruct](#)

响应信息

宏定义

- #define [XTP_ERR_MSG_LEN](#) 124

错误信息的字符串长度

类型定义

- typedef struct [XTPRspInfoStruct](#) [XTPRI](#)

响应信息

6.8.1 详细描述

定义业务公共数据结构

作者

中泰证券股份有限公司

6.9 xtp_quote_api.h 文件参考

定义行情订阅客户端接口

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

结构体

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

6.9.1 详细描述

定义行情订阅客户端接口

作者

中泰证券股份有限公司

Index

- CreateQuoteApi
 - XTP::API::QuoteApi, [12](#)
- demo_test_quote_api.cpp, [47](#)
 - main, [48](#)
 - ticker_count, [48](#)
- demo_test_quote_spi.h, [48](#)
- DemoTestMdSpi, [9](#)
 - OnTickByTick, [10](#)
- EPT_INVALID
 - xtp_api_data_type.h, [55](#)
- ERT_CASH_FORBIDDEN
 - xtp_api_data_type.h, [55](#)
- ERT_CASH_MUST
 - xtp_api_data_type.h, [55](#)
- ERT_CASH_OPTIONAL
 - xtp_api_data_type.h, [55](#)
- ETF_REPLACE_TYPE
 - xtp_api_data_type.h, [55](#)
- GetApiLastError
 - XTP::API::QuoteApi, [12](#)
- GetApiVersion
 - XTP::API::QuoteApi, [13](#)
- GetTradingDay
 - XTP::API::QuoteApi, [13](#)
- Login
 - XTP::API::QuoteApi, [13](#)
- Logout
 - XTP::API::QuoteApi, [14](#)
- main
 - demo_test_quote_api.cpp, [48](#)
- OnDepthMarketData
 - XTP::API::QuoteSpi, [20](#)
- OnDisconnected
 - XTP::API::QuoteSpi, [20](#)
- OnError
 - XTP::API::QuoteSpi, [20](#)
- OnOrderBook
 - XTP::API::QuoteSpi, [21](#)
- OnQueryAllTickers
 - XTP::API::QuoteSpi, [21](#)
- OnQueryTickersPriceInfo
 - XTP::API::QuoteSpi, [21](#)
- OnSubMarketData
 - XTP::API::QuoteSpi, [21](#)
- OnSubOrderBook
 - XTP::API::QuoteSpi, [22](#)
- OnSubTickByTick
 - XTP::API::QuoteSpi, [23](#)
- OnSubscribeAllMarketData
 - XTP::API::QuoteSpi, [22](#)
- OnSubscribeAllOrderBook
 - XTP::API::QuoteSpi, [22](#)
- OnSubscribeAllTickByTick
 - XTP::API::QuoteSpi, [23](#)
- OnTickByTick
 - DemoTestMdSpi, [10](#)
 - XTP::API::QuoteSpi, [23](#)
- OnUnSubMarketData
 - XTP::API::QuoteSpi, [23](#)
- OnUnSubOrderBook
 - XTP::API::QuoteSpi, [25](#)
- OnUnSubTickByTick
 - XTP::API::QuoteSpi, [26](#)
- OnUnSubscribeAllMarketData
 - XTP::API::QuoteSpi, [25](#)
- OnUnSubscribeAllOrderBook
 - XTP::API::QuoteSpi, [25](#)
- OnUnSubscribeAllTickByTick
 - XTP::API::QuoteSpi, [26](#)
- OrderBookStruct, [10](#)
- QueryAllTickers
 - XTP::API::QuoteApi, [14](#)
- QueryAllTickersPriceInfo
 - XTP::API::QuoteApi, [14](#)
- QueryTickersPriceInfo
 - XTP::API::QuoteApi, [14](#)
- QuoteApi, [11](#)
- QuoteSpi, [19](#)
- RegisterSpi
 - XTP::API::QuoteApi, [15](#)
- Release
 - XTP::API::QuoteApi, [15](#)
- SetHeartBeatInterval
 - XTP::API::QuoteApi, [15](#)
- SetUDPBufferSize
 - XTP::API::QuoteApi, [15](#)
- SubscribeAllMarketData
 - XTP::API::QuoteApi, [15](#)
- SubscribeAllOrderBook
 - XTP::API::QuoteApi, [15](#)
- SubscribeAllTickByTick
 - XTP::API::QuoteApi, [16](#)

- SubscribeMarketData
 - XTP::API::QuoteApi, [16](#)
- SubscribeOrderBook
 - XTP::API::QuoteApi, [16](#)
- SubscribeTickByTick
 - XTP::API::QuoteApi, [17](#)
- ticker_count
 - demo_test_quote_api.cpp, [48](#)
- trade_flag
 - XTPTickByTickTrade, [43](#)
- UnSubscribeAllMarketData
 - XTP::API::QuoteApi, [17](#)
- UnSubscribeAllOrderBook
 - XTP::API::QuoteApi, [17](#)
- UnSubscribeAllTickByTick
 - XTP::API::QuoteApi, [17](#)
- UnSubscribeMarketData
 - XTP::API::QuoteApi, [18](#)
- UnSubscribeOrderBook
 - XTP::API::QuoteApi, [18](#)
- UnSubscribeTickByTick
 - XTP::API::QuoteApi, [18](#)
- XTP::API::QuoteApi
 - CreateQuoteApi, [12](#)
 - GetApiLastError, [12](#)
 - GetApiVersion, [13](#)
 - GetTradingDay, [13](#)
 - Login, [13](#)
 - Logout, [14](#)
 - QueryAllTickers, [14](#)
 - QueryAllTickersPricelInfo, [14](#)
 - QueryTickersPricelInfo, [14](#)
 - RegisterSpi, [15](#)
 - Release, [15](#)
 - SetHeartBeatInterval, [15](#)
 - SetUDPBufferSize, [15](#)
 - SubscribeAllMarketData, [15](#)
 - SubscribeAllOrderBook, [15](#)
 - SubscribeAllTickByTick, [16](#)
 - SubscribeMarketData, [16](#)
 - SubscribeOrderBook, [16](#)
 - SubscribeTickByTick, [17](#)
 - UnSubscribeAllMarketData, [17](#)
 - UnSubscribeAllOrderBook, [17](#)
 - UnSubscribeAllTickByTick, [17](#)
 - UnSubscribeMarketData, [18](#)
 - UnSubscribeOrderBook, [18](#)
 - UnSubscribeTickByTick, [18](#)
- XTP::API::QuoteSpi
 - OnDepthMarketData, [20](#)
 - OnDisconnected, [20](#)
 - OnError, [20](#)
 - OnOrderBook, [21](#)
 - OnQueryAllTickers, [21](#)
 - OnQueryTickersPricelInfo, [21](#)
 - OnSubMarketData, [21](#)
 - OnSubOrderBook, [22](#)
 - OnSubTickByTick, [23](#)
 - OnSubscribeAllMarketData, [22](#)
 - OnSubscribeAllOrderBook, [22](#)
 - OnSubscribeAllTickByTick, [23](#)
 - OnTickByTick, [23](#)
 - OnUnSubMarketData, [23](#)
 - OnUnSubOrderBook, [25](#)
 - OnUnSubTickByTick, [26](#)
 - OnUnSubscribeAllMarketData, [25](#)
 - OnUnSubscribeAllOrderBook, [25](#)
 - OnUnSubscribeAllTickByTick, [26](#)
- XTP_ACCOUNT_CREDIT
 - xtp_api_data_type.h, [55](#)
- XTP_ACCOUNT_DERIVE
 - xtp_api_data_type.h, [55](#)
- XTP_ACCOUNT_NORMAL
 - xtp_api_data_type.h, [55](#)
- XTP_ACCOUNT_TYPE
 - xtp_api_data_type.h, [55](#)
- XTP_ACCOUNT_UNKNOWN
 - xtp_api_data_type.h, [55](#)
- XTP_BUSINESS_TYPE
 - xtp_api_data_type.h, [55](#)
- XTP_BUSINESS_TYPE_ALLOTMENT
 - xtp_api_data_type.h, [56](#)
- XTP_BUSINESS_TYPE_CASH
 - xtp_api_data_type.h, [55](#)
- XTP_BUSINESS_TYPE_DESIGNATION
 - xtp_api_data_type.h, [56](#)
- XTP_BUSINESS_TYPE ETF
 - xtp_api_data_type.h, [56](#)
- XTP_BUSINESS_TYPE_IPOS
 - xtp_api_data_type.h, [55](#)
- XTP_BUSINESS_TYPE_MARGIN
 - xtp_api_data_type.h, [56](#)
- XTP_BUSINESS_TYPE MONEY_FUND
 - xtp_api_data_type.h, [56](#)
- XTP_BUSINESS_TYPE_REPO
 - xtp_api_data_type.h, [56](#)
- XTP_BUSINESS_TYPE_STRUCTURED_FUND_PURCHASE_REDEMPTION
 - xtp_api_data_type.h, [56](#)
- XTP_BUSINESS_TYPE_STRUCTURED_FUND_SPLIT_MERGE
 - xtp_api_data_type.h, [56](#)
- XTP_BUSINESS_TYPE_UNKNOWN
 - xtp_api_data_type.h, [56](#)
- XTP_EXCHANGE_SH
 - xtp_api_data_type.h, [56](#)
- XTP_EXCHANGE_SZ
 - xtp_api_data_type.h, [56](#)
- XTP_EXCHANGE_TYPE
 - xtp_api_data_type.h, [56](#)
- XTP_EXCHANGE_UNKNOWN
 - xtp_api_data_type.h, [56](#)
- XTP_FUND_OPER_FAILED
 - xtp_api_data_type.h, [56](#)

XTP_FUND_OPER_PROCESSING
xtp_api_data_type.h, 56

XTP_FUND_OPER_STATUS
xtp_api_data_type.h, 56

XTP_FUND_OPER_SUBMITTED
xtp_api_data_type.h, 56

XTP_FUND_OPER_SUCCESS
xtp_api_data_type.h, 56

XTP_FUND_OPER_UNKNOWN
xtp_api_data_type.h, 56

XTP_FUND_TRANSFER_IN
xtp_api_data_type.h, 56

XTP_FUND_TRANSFER_OUT
xtp_api_data_type.h, 56

XTP_FUND_TRANSFER_TYPE
xtp_api_data_type.h, 56

XTP_FUND_TRANSFER_UNKNOWN
xtp_api_data_type.h, 56

XTP_LOG_LEVEL
xtp_api_data_type.h, 56

XTP_LOG_LEVEL_DEBUG
xtp_api_data_type.h, 57

XTP_LOG_LEVEL_ERROR
xtp_api_data_type.h, 57

XTP_LOG_LEVEL_FATAL
xtp_api_data_type.h, 57

XTP_LOG_LEVEL_INFO
xtp_api_data_type.h, 57

XTP_LOG_LEVEL_TRACE
xtp_api_data_type.h, 57

XTP_LOG_LEVEL_WARNING
xtp_api_data_type.h, 57

XTP_MARKET_TYPE
xtp_api_data_type.h, 57

XTP_MKT_INIT
xtp_api_data_type.h, 57

XTP_MKT_SH_A
xtp_api_data_type.h, 57

XTP_MKT_SZ_A
xtp_api_data_type.h, 57

XTP_MKT_UNKNOWN
xtp_api_data_type.h, 57

XTP_ORDER_ACTION_STATUS_ACCEPTED
xtp_api_data_type.h, 57

XTP_ORDER_ACTION_STATUS_REJECTED
xtp_api_data_type.h, 57

XTP_ORDER_ACTION_STATUS_SUBMITTED
xtp_api_data_type.h, 57

XTP_ORDER_ACTION_STATUS_TYPE
xtp_api_data_type.h, 57

XTP_ORDER_STATUS_ALLTRADED
xtp_api_data_type.h, 57

XTP_ORDER_STATUS_CANCELED
xtp_api_data_type.h, 57

XTP_ORDER_STATUS_INIT
xtp_api_data_type.h, 57

XTP_ORDER_STATUS_NOTRADEQUEUEING
xtp_api_data_type.h, 57

XTP_ORDER_STATUS_PARTTRADEDNOTQUEUEING
xtp_api_data_type.h, 57

XTP_ORDER_STATUS_PARTTRADEDQUEUEING
xtp_api_data_type.h, 57

XTP_ORDER_STATUS_REJECTED
xtp_api_data_type.h, 57

XTP_ORDER_STATUS_TYPE
xtp_api_data_type.h, 57

XTP_ORDER_STATUS_UNKNOWN
xtp_api_data_type.h, 57

XTP_ORDER_SUBMIT_STATUS_CANCEL_ACCEPTED
xtp_api_data_type.h, 58

XTP_ORDER_SUBMIT_STATUS_CANCEL_REJECTED
xtp_api_data_type.h, 58

XTP_ORDER_SUBMIT_STATUS_CANCEL_SUBMITTED
xtp_api_data_type.h, 58

XTP_ORDER_SUBMIT_STATUS_INSERT_ACCEPTED
xtp_api_data_type.h, 58

XTP_ORDER_SUBMIT_STATUS_INSERT_REJECTED
xtp_api_data_type.h, 58

XTP_ORDER_SUBMIT_STATUS_INSERT_SUBMITTED
xtp_api_data_type.h, 58

XTP_ORDER_SUBMIT_STATUS_TYPE
xtp_api_data_type.h, 57

XTP_PRICE_ALL_OR_CANCEL
xtp_api_data_type.h, 58

XTP_PRICE_BEST5_OR_CANCEL
xtp_api_data_type.h, 58

XTP_PRICE_BEST5_OR_LIMIT
xtp_api_data_type.h, 58

XTP_PRICE_BEST_OR_CANCEL
xtp_api_data_type.h, 58

XTP_PRICE_FORWARD_BEST
xtp_api_data_type.h, 58

XTP_PRICE_LIMIT
xtp_api_data_type.h, 58

XTP_PRICE_REVERSE_BEST_LIMIT
xtp_api_data_type.h, 58

XTP_PRICE_TYPE
xtp_api_data_type.h, 58

XTP_PRICE_TYPE_UNKNOWN
xtp_api_data_type.h, 58

XTP_PROTOCOL_TCP
xtp_api_data_type.h, 58

XTP_PROTOCOL_TYPE
xtp_api_data_type.h, 58

XTP_PROTOCOL_UDP
xtp_api_data_type.h, 58

XTP_SIDE_BUY
xtp_api_data_type.h, 58

XTP_SIDE_BUY_CLOSE

- xtp_api_data_type.h, 58
- XTP_SIDE_BUY_OPEN
 - xtp_api_data_type.h, 58
- XTP_SIDE_MERGE
 - xtp_api_data_type.h, 59
- XTP_SIDE_PURCHASE
 - xtp_api_data_type.h, 58
- XTP_SIDE_REDEMPTION
 - xtp_api_data_type.h, 59
- XTP_SIDE_SELL
 - xtp_api_data_type.h, 58
- XTP_SIDE_SELL_CLOSE
 - xtp_api_data_type.h, 58
- XTP_SIDE_SELL_OPEN
 - xtp_api_data_type.h, 58
- XTP_SIDE_SPLIT
 - xtp_api_data_type.h, 59
- XTP_SIDE_TYPE
 - xtp_api_data_type.h, 58
- XTP_SIDE_UNKNOWN
 - xtp_api_data_type.h, 59
- XTP_SPLIT_MERGE_STATUS
 - xtp_api_data_type.h, 59
- XTP_SPLIT_MERGE_STATUS_ALLOW
 - xtp_api_data_type.h, 59
- XTP_SPLIT_MERGE_STATUS_FORBIDDEN
 - xtp_api_data_type.h, 59
- XTP_SPLIT_MERGE_STATUS_ONLY_MERGE
 - xtp_api_data_type.h, 59
- XTP_SPLIT_MERGE_STATUS_ONLY_SPLIT
 - xtp_api_data_type.h, 59
- XTP_TBT_ENTRUST
 - xtp_api_data_type.h, 59
- XTP_TBT_TRADE
 - xtp_api_data_type.h, 59
- XTP_TBT_TYPE
 - xtp_api_data_type.h, 59
- XTP_TE_RESUME_TYPE
 - xtp_api_data_type.h, 59
- XTP_TERT_QUICK
 - xtp_api_data_type.h, 59
- XTP_TERT_RESTART
 - xtp_api_data_type.h, 59
- XTP_TERT_RESUME
 - xtp_api_data_type.h, 59
- XTP_TICKER_TYPE
 - xtp_api_data_type.h, 59
- XTP_TICKER_TYPE_BOND
 - xtp_api_data_type.h, 59
- XTP_TICKER_TYPE_FUND
 - xtp_api_data_type.h, 59
- XTP_TICKER_TYPE_INDEX
 - xtp_api_data_type.h, 59
- XTP_TICKER_TYPE_STOCK
 - xtp_api_data_type.h, 59
- XTP_TICKER_TYPE_UNKNOWN
 - xtp_api_data_type.h, 59
- XTPFundTransferNotice, 26
- XTPFundTransferReq, 27
- XTPMarketDataStruct, 27
- XTPOrderCancelInfo, 30
- XTPOrderInfo, 31
- XTPOrderInsertInfo, 32
- XTPQueryAssetRsp, 32
- XTPQueryETFBaseReq, 33
- XTPQueryETFBaseRsp, 34
- XTPQueryETFComponentReq, 34
- XTPQueryETFComponentRsp, 35
- XTPQueryFundTransferLogReq, 35
- XTPQueryIPOQuotaRsp, 36
- XTPQueryIPOTickerRsp, 36
- XTPQueryOrderReq, 37
- XTPQueryReportByExecIdReq, 37
- XTPQueryStkPositionRsp, 38
- XTPQueryStructuredFundInfoReq, 38
- XTPQueryTraderReq, 39
- XTPQuoteStaticInfo, 39
- XTPRspInfoStruct, 40
- XTPSpecificTickerStruct, 40
- XTPStructuredFundInfo, 41
- XTPTickByTickEntrust, 41
- XTPTickByTickStruct, 42
- XTPTickByTickTrade, 43
 - trade_flag, 43
- XTPTickerPriceInfo, 43
- XTPTradeReport, 44
- xoms_api_fund_struct.h, 49
- xoms_api_struct.h, 49
- xquote_api_struct.h, 51
- xtp_api_data_type.h, 52
 - EPT_INVALID, 55
 - ERT_CASH_FORBIDDEN, 55
 - ERT_CASH_MUST, 55
 - ERT_CASH_OPTIONAL, 55
 - ETF_REPLACE_TYPE, 55
 - XTP_ACCOUNT_CREDIT, 55
 - XTP_ACCOUNT_DERIVE, 55
 - XTP_ACCOUNT_NORMAL, 55
 - XTP_ACCOUNT_TYPE, 55
 - XTP_ACCOUNT_UNKNOWN, 55
 - XTP_BUSINESS_TYPE, 55
 - XTP_BUSINESS_TYPE_ALLOTMENT, 56
 - XTP_BUSINESS_TYPE_CASH, 55
 - XTP_BUSINESS_TYPE_DESIGNATION, 56
 - XTP_BUSINESS_TYPE ETF, 56
 - XTP_BUSINESS_TYPE_IPOS, 55
 - XTP_BUSINESS_TYPE_MARGIN, 56
 - XTP_BUSINESS_TYPE_MONEY_FUND, 56
 - XTP_BUSINESS_TYPE_REPO, 56
 - XTP_BUSINESS_TYPE_STRUCTURED_FUND↵
 - _PURCHASE_REDEMPTION, 56
 - XTP_BUSINESS_TYPE_STRUCTURED_FUND↵
 - _SPLIT_MERGE, 56
 - XTP_BUSINESS_TYPE_UNKNOWN, 56
 - XTP_EXCHANGE_SH, 56
 - XTP_EXCHANGE_SZ, 56

XTP_EXCHANGE_TYPE, 56
 XTP_EXCHANGE_UNKNOWN, 56
 XTP_FUND_OPER_FAILED, 56
 XTP_FUND_OPER_PROCESSING, 56
 XTP_FUND_OPER_STATUS, 56
 XTP_FUND_OPER_SUBMITTED, 56
 XTP_FUND_OPER_SUCCESS, 56
 XTP_FUND_OPER_UNKNOWN, 56
 XTP_FUND_TRANSFER_IN, 56
 XTP_FUND_TRANSFER_OUT, 56
 XTP_FUND_TRANSFER_TYPE, 56
 XTP_FUND_TRANSFER_UNKNOWN, 56
 XTP_LOG_LEVEL, 56
 XTP_LOG_LEVEL_DEBUG, 57
 XTP_LOG_LEVEL_ERROR, 57
 XTP_LOG_LEVEL_FATAL, 57
 XTP_LOG_LEVEL_INFO, 57
 XTP_LOG_LEVEL_TRACE, 57
 XTP_LOG_LEVEL_WARNING, 57
 XTP_MARKET_TYPE, 57
 XTP_MKT_INIT, 57
 XTP_MKT_SH_A, 57
 XTP_MKT_SZ_A, 57
 XTP_MKT_UNKNOWN, 57
 XTP_ORDER_ACTION_STATUS_ACCEPTED, 57
 XTP_ORDER_ACTION_STATUS_REJECTED, 57
 XTP_ORDER_ACTION_STATUS_SUBMITTED, 57
 XTP_ORDER_ACTION_STATUS_TYPE, 57
 XTP_ORDER_STATUS_ALLTRADED, 57
 XTP_ORDER_STATUS_CANCELED, 57
 XTP_ORDER_STATUS_INIT, 57
 XTP_ORDER_STATUS_NOTRADEQUEUEING, 57
 XTP_ORDER_STATUS_PARTTRADEDNOTQUEUING, 57
 XTP_ORDER_STATUS_PARTTRADEDQUEUEING, 57
 XTP_ORDER_STATUS_REJECTED, 57
 XTP_ORDER_STATUS_TYPE, 57
 XTP_ORDER_STATUS_UNKNOWN, 57
 XTP_ORDER_SUBMIT_STATUS_CANCEL_ACCEPTED, 58
 XTP_ORDER_SUBMIT_STATUS_CANCEL_REJECTED, 58
 XTP_ORDER_SUBMIT_STATUS_CANCEL_SUBMITTED, 58
 XTP_ORDER_SUBMIT_STATUS_INSERT_ACCEPTED, 58
 XTP_ORDER_SUBMIT_STATUS_INSERT_REJECTED, 58
 XTP_ORDER_SUBMIT_STATUS_INSERT_SUBMITTED, 58
 XTP_ORDER_SUBMIT_STATUS_TYPE, 57
 XTP_PRICE_ALL_OR_CANCEL, 58
 XTP_PRICE_BEST5_OR_CANCEL, 58
 XTP_PRICE_BEST5_OR_LIMIT, 58
 XTP_PRICE_BEST_OR_CANCEL, 58
 XTP_PRICE_FORWARD_BEST, 58
 XTP_PRICE_LIMIT, 58
 XTP_PRICE_REVERSE_BEST_LIMIT, 58
 XTP_PRICE_TYPE, 58
 XTP_PRICE_TYPE_UNKNOWN, 58
 XTP_PROTOCOL_TCP, 58
 XTP_PROTOCOL_TYPE, 58
 XTP_PROTOCOL_UDP, 58
 XTP_SIDE_BUY, 58
 XTP_SIDE_BUY_CLOSE, 58
 XTP_SIDE_BUY_OPEN, 58
 XTP_SIDE_MERGE, 59
 XTP_SIDE_PURCHASE, 58
 XTP_SIDE_REDEMPTION, 59
 XTP_SIDE_SELL, 58
 XTP_SIDE_SELL_CLOSE, 58
 XTP_SIDE_SELL_OPEN, 58
 XTP_SIDE_SPLIT, 59
 XTP_SIDE_TYPE, 58
 XTP_SIDE_UNKNOWN, 59
 XTP_SPLIT_MERGE_STATUS, 59
 XTP_SPLIT_MERGE_STATUS_ALLOW, 59
 XTP_SPLIT_MERGE_STATUS_FORBIDDEN, 59
 XTP_SPLIT_MERGE_STATUS_ONLY_MERGE, 59
 XTP_SPLIT_MERGE_STATUS_ONLY_SPLIT, 59
 XTP_TBT_ENTRUST, 59
 XTP_TBT_TRADE, 59
 XTP_TBT_TYPE, 59
 XTP_TE_RESUME_TYPE, 59
 XTP_TERT_QUICK, 59
 XTP_TERT_RESTART, 59
 XTP_TERT_RESUME, 59
 XTP_TICKER_TYPE, 59
 XTP_TICKER_TYPE_BOND, 59
 XTP_TICKER_TYPE_FUND, 59
 XTP_TICKER_TYPE_INDEX, 59
 XTP_TICKER_TYPE_STOCK, 59
 XTP_TICKER_TYPE_UNKNOWN, 59
 xtp_api_struct.h, 60
 xtp_api_struct_common.h, 60
 xtp_quote_api.h, 61