X1 柜台系统

<应用程序接口说明>

<v0.91>

<2016 年 8 月>

文档信息

项目名称	X1 交易系统项目
文档名称	<X1 交易系统应用程序接口说明文档>
版本号	<v0.91>
作者	X1 交易系统项目组

文档修订历史

版本	日期	修订内容	修订者	备注
0.1	2015.12.11	版本发布	X1 API 开发组	
0.2	2016.03.08	委托回报、撤单回报中添加 ErrorID	孟文	
0.3	2016.03.9	增加做市商相关接口	王浩然	
0.4	2016.03.17	OnRtnErrorMsg 接口描述	王浩然	
0.5	2016.03.18	增加交易所状态通知接口	王宣丹	
0.6	2016.03.30	下单接口支持期权期货	王宣丹	
0.7	2016.04.06	6.2.11 和 6.2.14 会话 id 不支持。	王宣丹	
0.8	2016.04.07	新增做市商相关错误码 做市商委托回报字段是否使用整理	王浩然	
0.81	2016.04.11	做市商委托回报和撤单请求补充说明 增加全部撤单错误码	王浩然	
0.82	2016.04.13	6.2.8 和 6.2.10 删除撤单成功和成交成功说明	王宣丹	
0.83	2016.04.19	6.1.5 增加订阅私有流的 UserAPI 接口：SubscribePrivateTopic	孟文	
0.84	2016.05.06	6.2.3 登录返回错误码为 35 说明	王宣丹	
0.85	2016.05.17	6.1.26/6.2.31 增加套利查询接口说明	王宣丹	
0.86	2016.05.17	详细解释 6.1.5 私有流订阅接口中参数	孟文	

		的取值范围		
0.87	2016.06.07	增加 init 接口的参数，输入输出线程绑定的 CPU core id	孟文	
0.88	2016.06.16	做市商成交回报不适用字段标注； 成交查询中接口描述中删除"版本不支持按柜台委托号 X1OrderID 或本地委托号 LocalOrderID 查询"； 交易所合约查询接口字段是否必填补充说明；	王浩然	
0.89	2016.07.26	委托查询中接口描述中删除"版本不支持按柜台委托号 X1OrderID 或本地委托号 LocalOrderID 查询"； 交易所合约查询接口字段是否必填补充说明.	王浩然	
0.90	2016.08.15	撤单请求接口参数描述补充说明，资金账号程序不校验。	王浩然	
0.91	2016.08.17	请求时增加-2,-3 返回值说明	齐国军	

目录

1. 简介

X1 交易系统应用程序接口（API）是一个基于 C++的类库，通过使用和扩展类库提供的功能来实现相关的交易功能，包括资金账户登陆，报单、撤单、等功能。

本文档的主要内容包括：

- 接口类库文件说明
- 接口的系统架构
- 接口开发规范
- 业务与接口的对照表
- 接口定义

2. 接口类库文件说明

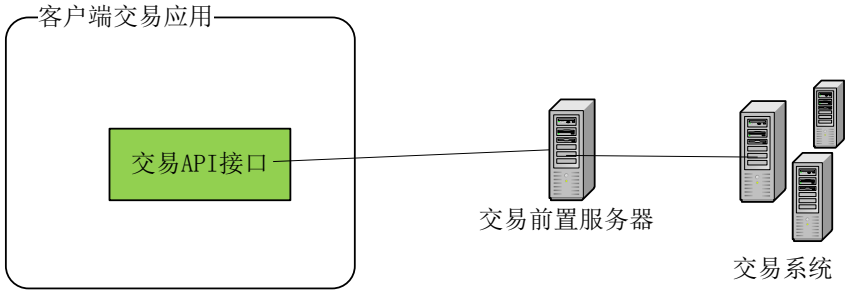
接口类库包含如下文件：

文件名	版本	文件描述
X1FtdcApiDataType.h	V0.1	定义接口所需的数据类型的头文件
X1FtdcApiStruct.h	V0.1	定义接口所需的数据接口的头文件
X1FtdcTraderApi.h	V0.1	定义交易接口的头文件

支持 Microsoft VisualC++6.0，VS2015 等集成开发环境。

3. 系统架构

3.1 接口与其他系统关系



说明：

- 本 API 接口在客户端交易应用（简称客户端）与交易系统的通信时使用；
- 客户端可以是人工交易客户端或者程序化交易系统；
- 交易 API 接口负责与交易前置连接；

3.2 通讯模式

API 与前置的通讯协议是建立在 TCP 协议上的通讯协议，一旦建立 TCP 连接，双方将保持该连接（长连接）

- 对话通讯模式：客户端主动发起请求，前置机接收请求，并立即将应答返回给客户端；

- 私有通讯模式：由前置机主动发起，通过 TCP 长连接向客户端发送特定信息，比如委托回报，成交回报等等；

3.3 接口模式

API 封装了两个接口，分别为 CX1FtdcTraderApi 和 CX1FtdcTraderSpi，两个接口对 API 与前置的通信及通信报文协议进行了封装，方便客户端应用程序的开发。客户端应用程序可以通过 CX1FtdcTraderApi 的接口发出操作请求，通过继承 CX1FtdcTraderSpi 并重载回调函数来处理后台服务的响应。

3.3.1 对话流和查询流编程接口通常如下：

请求：int CX1FtdcTraderApi::ReqXXX(CX1FtdcxxxField * pReqXXX)
响应：void CX1FtdcTraderSpi::OnRspXXX(CX1FtdcxxxField * pRspXXX,
CX1FtdcRspErrorField * pErrorInfo,
bool bIsLast);

其中请求接口的参数内容不能为空，每次请求时，需要检查接口的返回值是否为 0，每个结构体里包含了一个 RequestID 字段，当请求查询信息返回时，可以通过该字段将请求与响应对应起来。

当 API 收到后台服务应答时，CX1FtdcTraderSpi 的回调函数将被调用，即会调用客户端继承并实现的 Spi 函数。如果响应数据不止一个，则回调函数会被多次调用。回调函数的第一个参数为响应的具体数据，第二个参数为处理结果，表明本次请求的处理结果是成功还是失败。当失败时该值不为 NULL，所以当 Spi 函数被调用时，首先应该检测该值是否为 NULL，并从中获取错误 ID 和错误信息。

3.3.2 私有流编程接口

私有流中的数据中有客户的私有信息，如委托回报、成交回报等
通常私有流接收回报的编程如下：
void CX1FtdcTraderSpi::OnRtnXXX(CX1FtdcxxxField * pRtnXXX);
void CX1FtdcTraderSpi::OnRtnErrorMsg(CX1FtdcErrorRtnField * pErrorInfo);
当收到交易所的确认时，该类接口将被调用，如委托回报，回调参数为具体内容。

3.3.3 线程说明

与用户有关的 API 线程说明如下：

1. 发包线程：实例化 CX1FtdcTraderApi(用户在自己的逻辑线程中进行)

2. 收包线程：CX1FtdcTraderApi 对象初始化后(调用 Init 函数)，API 后台会自动创建回报处理线程，当收到用户请求对应的回报时，该线程会调用 CX1FtdcTraderSpi 子类对象中重载的回调函数。

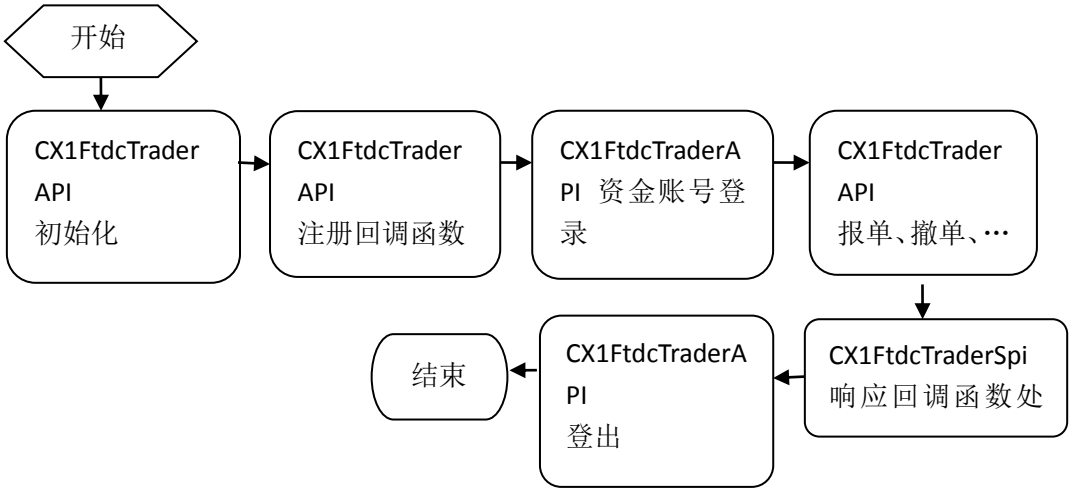
3. CX1FtdcTraderApi 对象中的接口是线程安全的。

4. 接口开发规范

4.1 命名空间

客户端交易 API 方法的命名空间为“x1ftdcapi”，使用该接口时，请添加命名空间的引用：
using namespace x1ftdcapi;

4.2 开发流程



4.3 CX1FtdcTraderSpi 接口

- CX1FtdcTraderSpi 接口定义了事件通知接口，开发人员必须正确继承并实现 CX1FtdcTraderSpi 接口，编写对应的事件处理方法。注意，在 Spi 函数接口中有 CX1FtdcRspErrorField 的参数时，若该参数不 NULL 表示有错误产生。其它参数在任何时候，均不会为 NULL。

4.4 RequestID 字段

- 接口定义了每次请求与响应报文的唯一标识：RequestID 字段，客户端应用通过该字段，将请求报文与响应报文对应起来，该 id 由客户应用产生并维护。

4.5 LocalOrderID 本地委托号字段

- 本地委托号唯一标识了一个会话中的每次报单，以及关联该报单的后续委托、成交、撤单等回报信息。在一个资金账号的一次登录会话范围内，本地委托号不能重复，且必须

是大于等于 1 的数字（推荐从 2 开始），可选的处理方式：每次报单在上一本地委托号基础上加 1。注意，本地委托号应该每次递增,如果输入负数会被强转为无符号 4 字节整数。

4.6 X1OrderID 柜台委托号字段

- 柜台委托号在 x1 系统里唯一标识了一笔报单，该笔报单在每个交易日里不会重复，且一般总是从 1 开始递增的（0 为无效的柜台委托号）。该委托号在委托响应成功时，会返回。

4.7 连接断开与重连

- 当 TCP 连接断开时，将使用 OnFrontDisconnected（）回调方法通知客户端；此时客户端 API 也会自动检测与前置机之间的连接，当网络可用，将自动建立连接，并使用 OnFrontConnected（）方法通知客户端，客户端可以在该方法中完成登录请求任务。

4.8 pErrorInfo 异常信息数据结构(错误码)

- pErrorInfo:返回异常信息的地址。在 Spi 函数进入时，若该指针不为 NULL，表示有错误，此时应处理错误信息，对于正确的调用，该变量值为 NULL。一般情况下，程序可先判断该变量是否为 NULL，再根据情况进行后续处理。

```
struct CX1FtdcRspErrorField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcSessionIDType        SessionID;          //会话标识
    TX1FtdcAccountIDType        AccountID;          //资金账号
    TX1FtdcErrorIDType          ErrorID;            //错误 ID
    TX1FtdcX1OrderIDType        X1OrderID;          //柜台委托号
    TX1FtdcLocalOrderIDType     LocalOrderID;       //本地委托号
    TX1FtdcErrorMsgInfoType     ErrorMsg;           //错误信息
    TX1FtdcInstrumentIDType     InstrumentID;       //合约代码
};
```

- 错误 ID 和错误信息对应关系

错误码	错误信息	说明	对应措施
0	success,	操作成功	
1	user name invalid,	登录账号非法	
2	passwd incorrect,	登录密码非法	
3	insert order failed,	委托响应返回失败	
4	insert order rtn failed,	交易所委托回报返回失败	

5	cancel order failed,	撤单失败	
6	internal error,	内部错误	
7	client ptr is null,	交易编码为空	
8	account in client ptr is null,	资金指针非法	
9	client is invalid,	交易编码非法	
10	x1 is disabled,	X1 被 disable	
11	can not find contract id,	未找到合约号	
12	local number is invalid,	本地委托号非法	
13	investor id is invalid,	投资者号非法	
14	X1 internal error,	X1 内部错误	
15	check money error,	资金检查错误	
16	check open buy error,	开 买错误	
17	check open sell error,	开 卖错误	
18	check close sell error,	平 卖错误	
19	check close buy error,	平 买错误	
20	input bs error,	输入 买卖标志错误	
21	input oc error,	输入开平标志错误	
22	can not find contract info in risk control info or contract is forbidden	风控表中未找到该账号下对应的合约信息或合约是禁用的	
23	local order id invalid,	本地号非法	
24	local order info invalid,	本地信息非法	
25	check x1 state failed,	检查 x1 状态失败	
26	flow control failed,	流控错误	
27	cancel order failed	撤单失败	
28	insert order failed	下单失败	
29	strategy error,	策略错误	
30	login already,	已经登录	
31	vio put data error when order match,	Vio 存数据错误	
32	vio put data number error when order	Vio 存数据错误	

	match,		
33	the user is not login,	用户未登录	
34	exceed x1 max order	超过 x1 最大下单数量	
35	query server connect failed	查询服务连接失败	
36	query server login failed	查询服务登录失败	
37	query server logout failed	查询服务登出失败	
38	Invalid buy sell flag	买卖标志错	用户检查相关报单数据
39	Invalid open close flag	开平标志错	用户检查相关报单数据
40	Invalid speculate hedge flag	投保标志错	用户检查相关报单数据
41	No trade code	交易编码不存在	用户检查相关报单数据
42	The trade code has been frozen, cannot trade	交易编码被冻结，不能交易	用户检查相关报单数据
43	Invalid contract	无效合约	用户检查相关报单数据
44	The order price mismatches the float-price of the contract	价格最小变动单位不符合	用户检查相关报单数据
45	Invalid order amount	下单数量不在合理范围内	用户检查相关报单数据
46	The trade code was forbidden trading this kind of contract	交易编码交易权限检查，合约不可以交易	用户检查相关报单数据
47	Invalid order price is out of price cap or floor,cannot trade	价位不在涨跌幅度内	用户检查相关报单数据
48	Placed close-position order failed since no enough positions	平仓数量超过可平数量	用户检查相关报单数据
49	Not suport market price order in the exchange	该品种不支持市价单	用户检查相关报单数据
50	Invalid combination	套利合约不存在	用户检查相关报

	contract		单数据
51	The local order ID already used	本地报单号重复	用户检查相关报单数据
52	Invalid order type	无效订单类型 市价、限价等	用户检查相关报单数据
53	Invalid contract type	无效合约类型，期货、期权、互换等	用户检查相关报单数据
54	Invalid order property	无效的订单属性	用户检查相关报单数据
55	the profit(loss)price must be less than the principal price	买止盈(损)价格必须小于委托价格	用户检查相关报单数据
56	Sell the surplus(loss) than the entrusted price price	卖止盈(损)价格必须大于委托价格	用户检查相关报单数据
57	The order already canceled,cannot cancel it again	重复撤单	用户检查相关报单数据
58	The force-close order does not allow to cancel	强平单不可以撤单	用户检查相关报单数据
59	max_margin value is invalid	合约的每手最大保证金设置不合法	联系期货公司
60	Invalid exchange member_no or trade_no	交易所会员号或者席位号无效	联系期货公司,柜台配置错误
61	trader address invalid	柜台服务地址无效	用户检查地址信息
62	param error	参数输入错误	用户检查接口调用的实参
63	query api ret failed	查询接口返回错误	
64	order rtn, but not find contract id	委托回报或成交回报返回时没有找到合约信息	
65	order rtn, but not find exchange id	委托回报或成交回报返回时没有找到交易所 id	
66	open position amount reach upper limit	开仓数量超过客户限制	
67	open position amount reach proprietary trading upper limit	开仓超过会员自营限仓	
68	cancel order failed, x1_order_id and	撤单失败,柜台委托号和本地委托号均为无效值	

	local_order_id are invalid		
69	trader client check failed	会员客户检查不通过	
70	client without position	客户无持仓	
71	trader money deficiency	资金不够	
72	order status checked failed	订单状态检查不合法	
73	sorry, x1 no server	X1 柜台订单达到上限	
74	open position exceed member agent position limit	大商所：70004	
75	Invalid speculator type	投保类型不合法	
76	Invalid openclose type	开平标志不合法	
77	Invalid buysell type	买卖标志不合法	
78	Invalid order property	订单属性不合法	
79	Invalid min match cnt	最小成交手数不合法	
80	Invalid instrument type	合约类型不合法	
81	user account invalid	用户账号不合法	
82	order is wait to be canceled	撤单中	
83	session id invalid	连接 ID 不合法	
84	x1 order info invalid	订单信息非法	
85	order cnt less min open cnt.	少于订单最小开仓手数	
86	order cnt less min close cnt.	少于最小平仓手数	
87	contract type invalid.	订单类型非法	
88	limit order cnt max limit each	超出限价单每笔最多下单手数	
89	market order cnt max limit each	超出市价单每笔最多下单手数	
90	API flowmeter limit	API 限流	
91	Server flowmeter limit	服务器限流	
92	send cancel all	发送批量撤单请求给交易所失败	

	orders to exchange error		
93	the buy price must be less than the sell price	双边报单，卖价必须大于买价	
94	send for quote to exchange error	发送询价请求给交易所失败	
95	api version does not match.	API 版本不支持当前服务端	
96	Exchange non tradable	交易所不能交易	
97	not allow cancel order in this moment.	当前交易状态下不允许撤单	
98	have reasonable quotation, cannot for quote.	场上有合理报价，不能询价	
99	market maker not allow for quote	做市商不允许询价。	
100	server config error,member_no and trader_no are not match	服务配置错误，会员号和席位号不匹配	
101	variety status is not in trading state	交易品种不再交易状态	
102	cancel order failed, x1_order_id invalid	撤单失败，柜台号无效	
103	cancel order failed, spd_order_id invalid	撤单失败，柜台号无效	
104	Invalid quote interval.	无效的询价间隔	
105	Invalid quote contract or no market maker.	无效询价合约或者没有做市商做市	
106	full matched, can not cancel	完全成交，不能撤单。	
107	exchange is not supported.	交易所不支持该操作	
108	api duplicate init is forbidden.	API 禁止重复初始化	
109	mmap failed	Mmap 失败	
110	conf of cpu core id is invalid	Cpu 核心配置错误	
111	login too often, please try again	频繁登陆，稍后尝试	

	later...		
112	exceed x1 auto max order	超过自动单数量	
113	access control forbidden	权限访问禁止	
114	x1 order id invalid	X1 柜台委托号错误	
-1	Unknown error		
-2	The length of the account ID is longer than 12, please input again		
-3	%s contains invalid charactor		
-4	%s is invalid price value		
-9	Invalid command		
-11	Query frequency is too fast, this query operation is failed		
-12	The frequency of placing or canceling order is too fast, this order request operation is failed		
-13	Receive multiple invalid command requests, front will close this connection		
-14	The API version is too low, please upgrade		
-15	The version of the connected front is too low, please use correct API version		
-16	Try to connect to md front using trade api		
-17	Try to connect to trade front using md api		
-18	The md front doesnot configure multicasting market data		

-19	The connected md front does not support custom market data		
-20	The connected trader front does not support market maker		
-22	Not market maker user, prohibit the operation		
-30	Max session number limit by this account:%d		
-31	Not login		
-32	This user ID [%s] already login		
-33	This kind user ID [%s] has approached max login limit, please use other IDs		
-34	Query exchange time failed, please login again to get exchange time		
-35	Not login or already logout		
-36	Logout failed or already logout, please check		
-37	The trade server performance reaches to upper limit, please try again later		
-38	Invalid contract		
-39	Empty contract		
-40	Not subscribed or already unsubscribed the contract:%s		
-41	Not subscribed or already unsubscribed the exchange:%s		

-50	Cannot find the order ID (%d) corresponding to the account ID		
-51	The contract is incorrect to the order ID		
-52	Cannot find the order ID for the contract		
-53	Cannot find the order info by the local order ID		
-54	The local order ID is incorrect to the contract		
-55	Both order ID and local ID are empty		
-56	Cannot find any valid trade code		
-57	Please check the condition order configure file		
-58	Cannot find the order info by the algorithm order ID		
-59	The algorithm order ID is incorrect to the contract		
-60	The order is in placing, please try to cancel it again later		
-61	The exchange does not support batch cancellation orders		
-62	Invalid transaction time, banned mass cancellation		
-63	Cannot find the matched order info for the specified contract and order ID		

-80	The local order ID already used		
-81	Invalid order type		
-82	Invalid contract type		
-83	Arbitrage or roll over exchange order doesnot support the order within property		
-84	Invalid investment type		
-85	Invalid buy-sell flag		
-86	Invalid open-close flag		
-87	Invalid auto order flag		
-88	Invalid order property		
-89	The local order ID must be positive number		
-90	The order type does not match the given contract		
-91	The contract type is incorrect to this contract		
-92	During the continuous trading, banned under open automatically		
-93	Buy only the profit (loss) price must be less than the principal price		
-94	Sell the surplus (loss) than the entrusted price price		
-95	Option not surpport profit (loss) price order		
-96	Only the CFFEX market orders to support the order property		
-98	Inquiry instruction		

	only support options contract		
-99	Please use the common order insert interface for unilateral quotation		
-102	Send error data to trade front, please check		
-110	Not support trading order at this optimized-xspeed system		
-114	This account cannot open a new position		
-120	Invalid comparison flag		
-121	Invalid price reference		
-122	Invalid correlation property		
-123	Invalid association number		
-124	Invalid order amount		
-125	Invalid effective date		
-126	Invalid freezing type		
-127	The breakdown number is invalid		
-128	The order price mismatches the float-price of the contract		
-129	The breakdown type is invalid		
-130	The Conditional order type is invalid		
-131	Invalid limit number		
-200	Invalid exchange code		
-201	The contract is not from the specified		

	exchange		
-202	Cannot find the contract info		
-203	Invalid or empty trade code		
-204	Please check the order ID		
-205	The password length is longer than 16, and it must be digits or letters		
-206	Queried empty data		
-207	Empty date when request confirm bill		
-208	Please input correct confirm date, form:yyyy.mm.dd		
-209	Invalid confirm flag		
-210	Invalid delegate state		
-211	The query command does not support arbitrage contracts		
-212	Query frequency is too fast or the last querying was not finished		
-213	The X1OrderID is incorrect to the accountID		
-230	Invalid transfer type		
-231	Invalid transfer money amount:%lf		
-401	Front communication exception		

5. 业务与接口对照表

业务类型	业务	请求接口	响应接口	数据流
登录	登录	CX1FtdcTraderApi::ReqUserLogin	CX1FtdcTraderSpi::OnRspUserLogin	对话流
	登出	CX1FtdcTraderApi::ReqUserLogout	CX1FtdcTraderSpi::OnRspUserLogout	对话流
交易	下单	CX1FtdcTraderApi:: ReqInsertOrder	CX1FtdcTraderSpi::OnRspInsertOrder	对话流
	撤单	CX1FtdcTraderApi::ReqCancelOrder	CX1FtdcTraderSpi:: OnRspCancelOrder	对话流
私有回报	错误回报	N/A	CX1FtdcTraderSpi::OnRtnErrorMsg	私有流
	成交回报	N/A	CX1FtdcTraderSpi::OnRtnMatchedInfo	私有流
	委托回报	N/A	CX1FtdcTraderSpi:: OnRtnOrder	私有流
	撤单回报	N/A	CX1FtdcTraderSpi:: OnRtnCancelOrder	私有流
查询	委托查询	CX1FtdcTraderApi::ReqQryOrderInfo	CX1FtdcTraderSpi::OnRspQryOrderInfo	查询流
	成交查询	CX1FtdcTraderApi::ReqQryMatchInfo	CX1FtdcTraderSpi::OnRspQryMatchInfo	查询流
	持仓查询	CX1FtdcTraderApi:: ReqQryPosition	CX1FtdcTraderSpi::OnRspQryPosition	查询流
	持仓明细查询	CX1FtdcTraderApi::ReqQryPositionDetail	CX1FtdcTraderSpi::OnRspQryPositionDetail	查询流
	资金查询	CX1FtdcTraderApi:: ReqQryCustomerCapital	CX1FtdcTraderSpi::OnRspCustomerCapital	查询流
	交易所合约查询	CX1FtdcTraderApi:: ReqQryExchangeInstrument	CX1FtdcTraderSpi::OnRspQryExchangeInstrument	查询流
	指定合约查询	CX1FtdcTraderApi::ReqQrySpecifyInstrument	CX1FtdcTraderSpi::OnRspQrySpecifyInstrument	查询流
	交易所状态查询	CX1FtdcTraderApi::ReqQryExchangeStatus	CX1FtdcTraderSpi::OnRspQryExchangeStatus	查询流
	交易所状态通知	N/A	CX1FtdcXspeedTraderSpi::OnRtnExchangeStatus	对话流

6. CX1FtdcTraderAPI 使用参考手册

约定：

m/M	必填
NM	不需要必填
BLK/BLANK	填空格
ZERO	填 0
N/A	不关系，不修改，不使用

6.1. CX1FtdcTraderApi 接口

注意事项：

1）用户调用接口前，需要进行 memset 清空结构体内容，再填写需要的字段信息。
2）接口线程安全
3）买卖，开平、投保、订单类型、订单属性这些字段因为使用了位域处理，在 API 端不做数值有效性检查判断，按照头文件指定数值输入，输入其他值也会被转换为不确定数值。

注：本节所示接口的委托响应均有可能晚于委托回报返回。

6.1.1. CreateCX1FtdcTraderApi 方法

产生一个 CX1FtdcTraderApi 实例

函数原型：

```
static CX1FtdcTraderApi *CreateCX1FtdcTraderApi();
```

返回值：

返回一个指向 CX1FtdcTraderApi 实例的指针。

6.1.2. Init 方法

该方法会和交易服务器建立连接，并启动一个接收线程，同时该方法注册一个回调函数集。

函数原型：

```
int Init( char *pszFrontAddr, CX1FtdcTraderSpi *pSpi, int output_core, int input_core);
```

参数：

pszFrontAddr：前置机地址。采用如下的格式：″tcp://172.16.0.31:10910 ″的形式
pSpi：指向回调函数集的指针。
output_core：输出线程绑定的 cpu core id，-1 表示不绑定
input_core：输入线程绑定的 cpu core id，-1 表示不绑定
Input_core output_core 不支持仅有一个参数为-1 的情形

返回值：

0: 初始化成功
-1：初始化失败，此时需要检查 OnRtnErrorMsg 给出的错误信息

Init 失败原因可能：

1) 填写的 addr 格式不正确或者 addr 中的 ip 地址及端口不正确，如使用交易 API 连接到了行情前置的端口
2) 网络问题，可 telnet 连接 ip 及 port，检查是否畅通

6.1.3. Join 方法

等待接口线程结束运行

函数原型：

```
int Join(void);
```

返回值：

0：线程退出成功

-1：线程退出失败

6.1.4. Release 方法

退出 API 各线程，释放 API 的各项资源

函数原型：

```
void Release(void);
```

返回值：无

当调用 CreateCX1FtdcTraderApi 生成一个 API 实例后，退出时必须调用 Release 接口，否则会造成资源泄漏。就像 C 中 malloc 了一块内存，需要 free 一样。

6.1.5. SubscribePrivateTopic 方法

订阅私有流，在 Init 方法调用之后，ReqUserLogin 方法调用之前调用。若不调用该接口，则收不到私有流

函数原型：

```
void SubscribePrivateTopic(int priflow_req_flag, unsigned int pri_no);
```

返回值：无

参数：

priflow_req_flag：私有流拉取方式，取值范围如下：

X1_PrivateFlow_Req_Quick	0：只传送登录后私有流的内容
X1_PrivateFlow_Req_Restart	1：从本交易日开始重传
X1_PrivateFlow_Req_Resume	2：从上次收到的续传
X1_PrivateFlow_Req_Specify	3：从指定编号开始续传

pri_no：指定的私有流编号。当 priflow_req_flag 取值 3 时，该参数才有效，其他情况，传入 0 值即可。

6.1.6. ReqUserLogin 方法

用户发出请求登录。

函数原型：

```
int ReqUserLogin(struct CX1FtdcUserLoginField *pUserLoginData);
```

参数：

pUserLoginData：指向用户登录请求结构的地址。用户请求登录结构：

```
struct CX1FtdcUserLoginField
{
    TX1FtdcRequestIDType        RequestID;      //请求 ID
    TX1FtdcAccountIDType        AccountID;      //资金账户 ID
    TX1FtdcPasswdType           Passwd;         //密码
    TX1FtdcCompanyIDType        CompanyID;      //厂商 ID  (预留)

};
```

字段名称	是否必填	取值范围	字段描述
RequestID	NM		请求 ID
AccountID	M		资金帐号 ID
Passwd	M		密码
CompanyID	BLK		厂商 ID

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3: 请求包含非法字符。

注：AccountID 为用户的登陆号. 登录账号长度为 12 字节。

6.1.7. ReqUserLogout 方法

用户发出退出请求。

函数原型：

```
int ReqUserLogout( struct CX1FtdcUserLogoutField *pUserLogoutData );
```

参数：

pUserLogoutData:指向用户退出请求结构的地址。用户请求退出结构：

```
struct CX1FtdcUserLogoutField
{
    TX1FtdcRequestIDType        RequestID;      //请求 ID
    TX1FtdcAccountIDType        AccountID;      //资金帐号 ID
    TX1FtdcSessionIDType        SessionID;      //会话 ID    (预留)
};
```

字段名称	是否必填	取值范围	字段描述
RequestID	NM		请求 ID
AccountID	M		资金帐号 ID

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3: 请求包含非法字符。

注：AccountID 为用户的登陆号:用户可以传空或传入正确的登录号；RequestID，SessionID 两个字段目前版本无用

6.1.8. ReqInsertOrder 方法

用户发出下单请求。

函数原型：

```
int ReqInsertOrder( struct CX1FtdcInsertOrderField *pInsertOrderData );
```

参数：

pInsertOrderData:指向用户请求报单结构地址。用户请求报单结构：

```
struct CX1FtdcInsertOrderField
{
    TX1FtdcAccountIDType          AccountID;          //资金账户
    TX1FtdcLocalOrderIDType       LocalOrderID;       //本地委托号
    TX1FtdcInstrumentIDType       InstrumentID;       //合约代码,
    TX1FtdcPriceType              InsertPrice;        //报单价格,
    TX1FtdcAmountType             OrderAmount;        //报单数量
    TX1FtdcBuySellTypeType        BuySellType;        //买卖标志
    TX1FtdcOpenCloseTypeType      OpenCloseType;      //开平标志
    TX1FtdcSpeculatorType         Speculator;         //投保类型
    TX1FtdcInsertType             InsertType;         //自动单类别
    TX1FtdcOrderTypeType          OrderType;          //报单类型
    TX1FtdcOrderPropertyType      OrderProperty;      //报单附加属性
    TX1FtdcInstrumentTypeType     InstrumentType;     //合约类型
    TX1FtdcAmountType             MinMatchAmount;     //最小成交量
    TX1FtdcReservedType           ReservedType2;      //预留字段 2
    TX1FtdcRequestIDType          RequestID;          //请求 ID
    TX1FtdcCustomCategoryType     CustomCategory;     //自定义类别      (预留)
    TX1FtdcPriceType              ProfitLossPrice;    //止盈止损价格    (预留)
};
```

字段名称	是否必填	取值范围	字段描述
AccountID	M		资金帐号 ID
LocalOrderID	M		本地委托号
InstrumentID	M		合约代码

InsertPrice	M		报单价格
OrderAmount	M		报单数量
BuySellType	M	**1 买；2 卖**	买卖标志
OpenCloseType	M	**1 开仓；2 平仓；4 平今**	开平标志
Speculator	M		投保类型
InsertType	M	**1 普通单**	委托类别
OrderType	M		报单类型
OrderProperty	M		报单附加属性
InstrumentType	M	**0 期货；1 期权**	合约类型
MinMatchAmount	M		最小成交量
ReservedType2	BLK		预留字段
RequestID	NM		
CustomCategory	BLK		
ProfitLossPrice	BLK		

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3: 请求包含非法字符。

说明 1：OrderType 目前只支持输入：

```
#define DFITC_LIMITORDER                1 ///< 限价委托
#define DFITC_MKORDER                   2 ///< 市价委托
#define DFITC_ARBITRAGE                 4 ///< 套利委托
#define DFITC_EXTENSION                 8 ///< 展期互换委托
```

说明 2：买卖，开平、投保、订单类型、订单属性这些字段不做判断，按照头文件指定数值输入，输入其他值也会被转不确定数值引起错误。

目前期权期货的开平标志大商所为开或平

注意：用户输入的资金账号必须保证为 12 字节。

6.1.9. ReqCancelOrder 方法

用户发出撤单请求。

函数原型：

```
int ReqCancelOrder( struct CX1FtdcCancelOrderField *pCancelOrderData );
```

参数：

pCancelOrderData:指向请求撤单结构地址。用户请求撤单结构：

```
struct CX1FtdcCancelOrderField
{
    TX1FtdcAccountIDType        AccountID;          //登陆账户 ID  (预留)
    TX1FtdcX1OrderIDType        X1OrderID;          //柜台委托号
    TX1FtdcLocalOrderIDType     LocalOrderID;       //本地委托号
    TX1FtdcInstrumentIDType     InstrumentID;       //合约代码     (预留)
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcSessionIDType        SessionID;          //会话 ID
};
```

字段名称	是否必填	取值范围	字段描述
AccountID	NM		登陆账户 ID,程序不校验，可以不填。
X1OrderID	NM		柜台委托号
LocalOrderID	M		本地委托号
InstrumentID	NM		合约号，程序不校验，可以不填
RequestID	M		请求 ID
SessionID	BLK		会话 ID

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3: 请求包含非法字符。

说明：

当发起撤单时，X1OrderID 和 LocalOrderID 任意输入其中一个值即可，如果两个值都输入了，程序将优先使用 X1OrderID（两者同时小于等于 0 时，则无法撤单成功）。

注意，当客户端使用 API 的登出后，再登陆到前置后，此时需要用 LocalOrderID 加 SessionID 进行撤单。

当柜台重启后，需要使用 X1OrderID 进行撤单。

撤单不检查账户 id 和合约代码。

6.1.10. ReqQryOrderInfo 方法

用户发出当日委托查询请求。

函数原型：

```
int ReqQryOrderInfo (struct CX1FtdcQryOrderField *pOrderData);
```

参数：

pOrderData：指向请求当日委托查询结构地址。用户请求当日委托查询结构：

```
struct CX1FtdcQryOrderField
{
    TX1FtdcRequestIDType            RequestID;          //请求 ID
    TX1FtdcAccountIDType            AccountID;          //资金账户 ID
    TX1FtdcInstrumentTypeType       InstrumentType;     //合约类型
    TX1FtdcCustomCategoryType       CustomCategory;     //自定义类别
    TX1FtdcOrderAnswerStatusType    OrderStatus;        //委托状态
    TX1FtdcOrderTypeType            OrderType;          //报单类型
    TX1FtdcX1OrderIDType            X1OrderID;          //柜台委托号  (预留)
    TX1FtdcLocalOrderIDType         LocalOrderID;       //本地委托号  (预留)
    TX1FtdcInstrumentIDType         InstrumentID;       //合约代码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3: 请求包含非法字符。

说明：这里有丰富的查询条件设置，如果想查询所有委托记录，则只需要输入 AccountID 和 InstrumentType 即可。

查询处于某状态的报单，则中需要将 OrderStatus 设置为相应报单状态的值即可，如 3 表示未成交在队列的单子。**（注意 OrderStatus 和 OrderType 查询条件该版本暂不支持）**

查询指定报单类型的单子，如限价单或市价单等。

废单仅支持下单的委托查询。（V2.1 版本）

废单的价格均显示为 0.

6.1.11. ReqQryPosition 方法

用户发出持仓查询请求。

函数原型：

```
int ReqQryPosition( struct CX1FtdcQryPositionField*pPositionData );
```

参数：

pPositionData：指向请求持仓查询结构地址。用户请求持仓查询结构：

```
struct CX1FtdcQryPositionField
{
    TX1FtdcRequestIDType            RequestID;          //请求 ID
    TX1FtdcAccountIDType            AccountID;          //资金账户 ID
    TX1FtdcInstrumentIDType         InstrumentID;       //合约代码
    TX1FtdcInstrumentTypeType       InstrumentType;     //合约类型
};
```

说明：如果没有提供合约代码，则查询全部持仓信息

返回值：

0：请求发送成功。
-1：请求发送失败。
-2：字符串溢出
-3: 请求包含非法字符。

6.1.12. ReqQryPositionDetail 方法

用户发出持仓明细查询请求。

函数原型：

```
int ReqQryPositionDetail ( struct CX1FtdcQryPositionDetailField*pPositionDetailData );
```

参数：

pPositionDetailData：指向请求持仓明细查询结构地址。用户请求持仓明细查询结构：

```
struct CX1FtdcQryPositionDetailField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcAccountIDType        AccountID;          //资金账户 ID
    TX1FtdcInstrumentIDType     InstrumentID;       //合约代码
    TX1FtdcInstrumentTypeType   InstrumentType;     //合约类型
};
```

说明：如果没有提供合约代码，则查询所有合约的持仓明细信息

返回值：

0：请求发送成功。
-1：请求发送失败。
-2：字符串溢出
-3: 请求包含非法字符。

6.1.13. ReqQryMatchInfo 方法

用户发出当日成交查询请求。

函数原型：

```
int ReqQryMatchInfo (struct CX1FtdcQryMatchField* pMatchData);
```

参数：

pMatchData：指向请求当日成交查询结构地址。用户请求当日成交查询结构：

```
struct CX1FtdcQryMatchField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcAccountIDType        AccountID;          //资金账户 ID
    TX1FtdcInstrumentTypeType   InstrumentType;     //合约类型
    TX1FtdcCustomCategoryType   CustomCategory;     //自定义类别
    TX1FtdcOrderTypeType        OrderType;          //报单类型
```

```
    TX1FtdcX1OrderIDType        X1OrderID;          //柜台委托号
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3: 请求包含非法字符。

说明：这里有丰富的查询条件设置，如果想查询所有委托记录，则只需要输入 AccountID 和 InstrumentType 即可。（**注意 OrderType 查询条件该版本暂不支持**）

6.1.14. ReqQrySpecifyInstrument 方法

用户发出指定合约信息查询请求。

函数原型：

```
int ReqQrySpecifyInstrument (struct CX1FtdcQrySpecificInstrumentField* pInstrument);
```

参数：

pInstrument：指向请求指定合约信息查询结构地址。用户请求指定合约信息查询结构：

```
struct CX1FtdcQrySpecificInstrumentField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcAccountIDType        AccountID;          //资金账户 ID
    TX1FtdcInstrumentIDType     InstrumentID;       //合约代码
    TX1FtdcExchangeIDType       ExchangeID;         //交易所 ID
    TX1FtdcInstrumentTypeType   InstrumentType;     //合约类型
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3: 请求包含非法字符。

说明：ExchangeID 并非必须输入选项，因为目前期货市场上，一个合约代码在各交易所是唯一存在的。

6.1.15. ReqQryCustomerCapital 方法

用户发出资金查询请求。

函数原型：

```
int ReqQryCustomerCapital( struct CX1FtdcQryCapitalField*pCapitalData );
```

参数：

pCapitalData：指向请求资金查询结构地址。用户请求资金查询结构：

```
struct CX1FtdcQryCapitalField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcAccountIDType        AccountID;          //资金账户 ID
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3： 请求包含非法字符。

6.1.16. ReqQryExchangeStatus 方法

用户发出交易所状态查询请求

函数原型：

```
int ReqQryExchangeStatus(struct CX1FtdcQryExchangeStatusField*pQryExchangeStatusData)
```

参数：

pQryExchangeStatusData：指向交易所状态查询请求结构的地址。

```
struct CX1FtdcQryExchangeStatusField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcExchangeIDType       ExchangeID;         //交易所编码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3： 请求包含非法字符。

6.1.17. ReqQryExchangeInstrument 方法

查询交易所合约列表。

函数原型：

```
Int ReqQryExchangeInstrument(struct CX1FtdcQryExchangeInstrumentField* pExchangeInstrumentData );
```

参数：

pExchangeInstrumentData：指向交易所合约查询结构的地址。

```
struct CX1FtdcQryExchangeInstrumentField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcAccountIDType        AccountID;          //资金账户 ID
    TX1FtdcExchangeIDType       ExchangeID;         //交易所编码
    TX1FtdcInstrumentTypeType   InstrumentType;     //合约类型
};
```

字段名称	是否必填	取值范围	字段描述
RequestID	NM		请求 ID
AccountID	NM		资金帐号 ID
ExchangeID	NM		交易所编码
InstrumentType	NM	0 期货 1 期权	合约类型

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3: 请求包含非法字符。

说明：当 ExchangeID 为空时，表示查询各交易所的所有合约代码，也可指定查询某交易所的所有合约代码信息，但该接口不能查询到套利合约代码。

6.1.18. ReqQuoteInsert 方法

用户发出做市商报单请求

函数原型：

```
int ReqQuoteInsert(struct CX1FtdcQuoteInsertField * pQuoteInsertOrderData)
```

参数：

pQuoteInsertOrderData：指向做市商报单请求结构的地址。

```
struct CX1FtdcQuoteInsertField
{
    TX1FtdcAccountIDType          AccountID;          //资金账号     (预留)
    TX1FtdcRequestIDType          RequestID;          //请求 ID
    TX1FtdcLocalOrderIDType       LocalOrderID;       //本地委托号
    TX1FtdcInsertType             InsertType;         //自动单类别   (预留)
    TX1FtdcInstrumentIDType       InstrumentID;       //合约代码
    TX1FtdcQuoteIDType            QuoteID;            //询价编号
    TX1FtdcInstrumentTypeType     InstrumentType;     //合约类型
    TX1FtdcAmountType             BuyOrderAmount;     //报单数量（买）
    TX1FtdcAmountType             SellOrderAmount;    //报单数量（卖）
    TX1FtdcPriceType              BuyInsertPrice;     //委托价格（买）
    TX1FtdcPriceType              SellInsertPrice;    //委托价格（卖）
    TX1FtdcOpenCloseTypeType      BuyOpenCloseType;   //开平标志（买）
    TX1FtdcOpenCloseTypeType      SellOpenCloseType;  //开平标志（卖）
    TX1FtdcSpeculatorType         BuySpeculator;      //投资类别（买）
    TX1FtdcSpeculatorType         SellSpeculator;     //投资类别（卖）
    TX1FtdcStayTimeType           StayTime;           //停留时间
    TX1FtdcCustomCategoryType     CustomCategory;     //自定义类别   (预留)
};
```

字段名称	是否必填	取值范围	字段描述
RequestID	NM		请求 ID
AccountID	M		资金帐号 ID
Passwd	M		密码
CompanyID	M		厂商 ID

备注：

stayTime 停留时间字段：仅支持郑州。其它情况可设置为 0

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：检测异常。

6.1.19. ReqQuoteCancel 方法

用户发出做市商撤单请求

函数原型：

```
int ReqQuoteCancel(struct CX1FtdcCancelOrderField * pQuoteCancelOrderData)
```

参数：

pQuoteCancelOrderData：指向做市商撤单请求结构的地址。

```
struct APISTRUCT CX1FtdcCancelOrderField
{
    TX1FtdcAccountIDType        AccountID;      //资金账户 ID（预留)
    TX1FtdcX1OrderIDType        X1OrderID;      //柜台委托号
    TX1FtdcLocalOrderIDType     LocalOrderID;   //本地委托号
    TX1FtdcInstrumentIDType     InstrumentID;   //合约代码
    TX1FtdcRequestIDType        RequestID;      //请求 ID
    TX1FtdcSessionIDType        SessionID;      //会话 ID
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3： 请求包含非法字符。

备注：

对于双边应价，本接口只支持撤双边，不支持只撤销一边的操作。

6.1.20. ReqCancelAllOrder 方法

用户发出做市商全部撤单请求

函数原型：

```
int ReqCancelAllOrder(struct CX1FtdcCancelAllOrderField * pCancelAllOrderData)
```

参数：

pCancelAllOrderData：指向做市商全部撤单请求结构的地址。

```
struct CX1FtdcCancelAllOrderField
{
    TX1FtdcRequestIDType        RequestID;      //请求 ID
    TX1FtdcAccountIDType        AccountID;      //资金账户 ID（预留)
    TX1FtdcExchangeIDType       ExchangeID;     //交易所编码（预留)
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3： 请求包含非法字符。

说明：

该指令目前只支持大商所的做市商报价订单，其它交易所暂不支持，普通期货期权订单也不支持

6.1.21. ReqForQuote 方法

用户发出询价请求

函数原型：

```
int ReqForQuote(struct CX1FtdcForQuoteField * pForQuoteData)
```

参数：

pForQuoteData：指向用户发出询价请求结构的地址。

```
struct CX1FtdcForQuoteField
{
    TX1FtdcRequestIDType          RequestID;      //请求 ID
    TX1FtdcAccountIDType          AccountID;      //资金账户 ID（预留)
    TX1FtdcInstrumentIDType       InstrumentID;   //合约代码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3： 请求包含非法字符。

6.1.22. ReqQryQuoteOrderInfo 方法

用户发出应/报价查询请求

函数原型：

```
int ReqQryQuoteOrderInfo(struct CX1FtdcQuoteOrderField * pQuoteOrderData)
```

参数：

pQuoteOrderData：指向用户发出应/报价查询请求结构的地址。

```
struct CX1FtdcQuoteOrderField
{
    TX1FtdcRequestIDType          RequestID;      //请求 ID
    TX1FtdcExchangeIDType         ExchangeID;     //交易所
    TX1FtdcAccountIDType          AccountID;      //资金账户
    TX1FtdcInstrumentIDType       InstrumentID;   //合约代码
    TX1FtdcLocalOrderIDType       LocalOrderID;   //本地委托号
    TX1FtdcX1OrderIDType          X1OrderID;      //柜台委托号
    TX1FtdcOrderAnswerStatusType  OrderStatus;    //委托状态
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3： 请求包含非法字符。

6.1.23. ReqQryQuoteNotice 方法

用户发出询价通知查询请求

函数原型：

```
int ReqQryQuoteNotice(struct CX1FtdcQryQuoteNoticeField * pQryQuoteNoticeData)
```

参数：

pQryQuoteNoticeData：指向用户发出查询询价通知请求结构地址。

```
struct CX1FtdcQryQuoteNoticeField
{
    TX1FtdcAccountIDType        AccountID;      //资金账号
    TX1FtdcRequestIDType        RequestID;      //请求 ID
    TX1FtdcExchangeIDType       ExchangeID;     //交易所
    TX1FtdcInstrumentIDType     InstrumentID;   //合约代码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3： 请求包含非法字符。

6.1.24. ReqQryForQuote 方法

用户发出询价委托查询请求

函数原型：

```
int ReqQryForQuote(struct CX1FtdcQryForQuoteField * pQryForQuoteData)
```

参数：

pQryForQuoteData：指向用户发出查询询价委托请求结构地址。

```
struct CX1FtdcQryForQuoteField
{
    TX1FtdcRequestIDType        RequestID;      //请求 ID
    TX1FtdcAccountIDType        AccountID;      //资金账户 ID
    TX1FtdcInstrumentIDType     InstrumentID;   //合约代码
    TX1FtdcExchangeIDType       ExchangeID;     //交易所
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3： 请求包含非法字符。

6.1.25. ReqQryArbitrageInstrument 方法

查询交易所套利合约列表。

函数原型：

```
int CX1FtdcTraderApiImp::ReqQryArbitrageInstrument(struct CX1FtdcAbiInstrumentField * pAbtriInstrumentData)
```

参数：

pAbtriInstrumentData：指向套利合约结构的地址。

```
struct APISTRUCT CX1FtdcAbiInstrumentField
{
    TX1FtdcRequestIDType        lRequestID;     //请求 ID
    TX1FtdcAccountIDType        accountID;      //资金账户 ID
    TX1FtdcExchangeIDType       exchangeID;     //交易所代码
};
```

返回值：

0：请求发送成功。

-1：请求发送失败。

-2：字符串溢出

-3: 请求包含非法字符。

说明：当 exchangeID 为空时，表示查询各交易所的所有合约代码，也可指定查询某交易所的所有合约代码信息。

6.2. CX1FtdcTraderSpi 接口

CX1FtdcTraderSpi 实现了事件通知接口，用户需要实现此类接口，编写事件处理方法来处理用户感兴趣的事件。

6.2.1. OnFrontConnected 方法

该方法是在 Api 和前置机建立连接后被调用，该调用仅仅是说明 tcp 连接已经建立成功。用户需要自行登录才能进行后续的业务操作，当然也可以在该函数内进行登录请求。连接失败则此方法不会被调用。

函数原型：

```
void OnFrontConnected();
```

6.2.2. OnFrontDisconnected 方法

该方法是在 Api 和前置机连接断开后被调用。

函数原型：

```
void OnFrontDisconnected(int nReason);
```

6.2.3. OnRspUserLogin 方法

当用户发出登录请求后，前置机返回响应时此方法会被调用，通知用户登录是否成功。

函数原型：

```
Void OnRspUserLogin(struct CX1FtdcRspUserLoginField*pUserLoginInfoRtn,struct
 CX1FtdcRspErrorField *pErrorInfo )
```

参数：

pUserLoginInfoRtn：返回用户登录信息结构地址：

```
struct CX1FtdcRspUserLoginField
{
    TX1FtdcRequestIDType            RequestID;        //请求 ID
    TX1FtdcAccountIDType            AccountID;        //资金帐号 ID
    TX1FtdcAccountLoginResultType   LoginResult;      //登录结果
    TX1FtdcLocalOrderIDType         InitLocalOrderID; //初始本地委托号  (预留)
    TX1FtdcSessionIDType            SessionID;        //sessionID(会话 ID)
    TX1FtdcErrorIDType              ErrorID;          //错误 ID
    TX1FtdcErrorMsgInfoType         ErrorMsg;         //错误信息
    TX1FtdcTimeType                 DCEtime;          //大商所时间      (预留)
    TX1FtdcTimeType                 SHFETime;         //上期所时间      (预留)
    TX1FtdcTimeType                 CFFEXTime;        //中金所时间      (预留)
    TX1FtdcTimeType                 CZCETime;         //郑商所时间      (预留)
    TX1FtdcTimeType                 INETime;          //上能所时间      (预留)
};
```

当 loginResult 为 0 时表示登录成功，且登录成功时，pErrorInfo 为 NULL，否则 pErrorInfo 中将包含错误 ID 和错误信息。成功时，用户将获取一个会话 ID,当 SessionID 为 1 的时候，默认为用户第一次登陆.

注：当用户登录返回错误码 35,用户需 release API 或 重启 API 后重新登录。

6.2.4. OnRspUserLogout 方法

当用户发出退出请求后，前置机返回响应此方法会被调用，通知用户退出状态。

函数原型：

```
void OnRspUserLogout( struct CX1FtdcRspUserLogoutInfoField*pUserLogoutInfoRtn,struct
 CX1FtdcRspErrorField *pErrorInfo );
```

参数：

pUserLogoutInfoRtn:返回用户退出信息结构地址：

```
struct CX1FtdcRspUserLogoutInfoField
{
    TX1FtdcRequestIDType            RequestID;      //请求 ID
    TX1FtdcAccountIDType            AccountID;      //资金账户 ID
    TX1FtdcAccountLogoutResultType  LogoutResult;   //退出结果
    TX1FtdcErrorIDType              ErrorID;        //错误 ID
    TX1FtdcErrorMsgInfoType         ErrorMsg;       //错误信息
};
```

6.2.5. OnRspInsertOrder 方法

下单应答。当用户录入报单后，前置返回响应时，该方法会被调用。

函数原型：

```
void OnRspInsertOrder(struct CX1FtdcRspOperOrderField*pOrderRtn,struct
 CX1FtdcRspErrorField *pErrorInfo );
```

参数：

pOrderRtn：返回下单响应信息结构地址。下单响应信息结构：

```
struct CX1FtdcRspOperOrderField
{
    TX1FtdcLocalOrderIDType         LocalOrderID;       //本地委托号
    TX1FtdcOrderAnswerStatusType    OrderStatus;        //委托状态
    TX1FtdcRequestIDType            RequestID;          //请求 ID
    TX1FtdcPriceType                Margin;             //冻结保证金
    TX1FtdcSessionIDType            SessionID;          //会话 ID
    TX1FtdcX1OrderIDType            X1OrderID;          //柜台委托号      (预留)
    TX1FtdcPriceType                Fee;                //手续费          (预留)
    TX1FtdcCustomCategoryType       CustomCategory;     //自定义类别      (预留)
    TX1FtdcAccountIDType            AccountID;          //资金账户 ID     (预留)
    TX1FtdcInstrumentIDType         InstrumentID;       //合约代码        (预留)
    TX1FtdcExchangeIDType           ExchangeID;         //交易所          (预留)
    TX1FtdcBuySellTypeType          BuySellType;        //买卖            (预留)
    TX1FtdcOpenCloseTypeType        OpenCloseType;      //开平            (预留)
    TX1FtdcInstrumentTypeType       InstrumentType;     //合约类型        (预留)
    TX1FtdcSpeculatorType           Speculator;         //投资类别        (预留)
    TX1FtdcPriceType                InsertPrice;        //委托价          (预留)
    TX1FtdcPriceType                ProfitLossPrice;    //止盈止损价格    (预留)
    TX1FtdcAmountType               MinMatchAmount;     //最小成交量      (预留)
    TX1FtdcAmountType               OrderAmount;        //委托数量        (预留)
    TX1FtdcInsertType               InsertType;         //自动单类别      (预留)
    TX1FtdcOrderTypeType            OrderType;          //订单类型        (预留)
```

```
    TX1FtdcOrderPropertyType        OrderProperty;      //订单属性          (预留)
    TX1FtdcClientIDType             ClientID;           //交易编码          (预留)
};
```

当报单发生错误时，pErrorInfo 不为 NULL，并在其中包含了错误 ID 及错误信息，和请求报单时的 LocalOrderID，用于对应客户程序的报单。报单成功时，是表示 X1 台系统确认了该笔报单，该报单也同时报到了交易所，但交易所还未确认。

6.2.6. OnRspCancelOrder 方法

撤单应答。当用户撤单后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspCancelOrder(struct CX1FtdcRspOperOrderField* pOrderCanceledRtn,struct
CX1FtdcRspErrorField *pErrorInfo );
```

参数：

pOrderCanceledRtn：返回撤单响应信息结构。撤单响应信息结构：

```
struct CX1FtdcRspOperOrderField
{
    TX1FtdcLocalOrderIDType         LocalOrderID;       //本地委托号
    TX1FtdcX1OrderIDType            X1OrderID;          //柜台委托号
    TX1FtdcOrderAnswerStatusType    OrderStatus;        //委托状态
    TX1FtdcRequestIDType            RequestID;          //请求 ID
    TX1FtdcPriceType                Margin;             //冻结保证金
    TX1FtdcSessionIDType            SessionID;          //会话 ID
    TX1FtdcPriceType                Fee;                //手续费            (预留)
    TX1FtdcCustomCategoryType       CustomCategory;     //自定义类别        (预留)
    TX1FtdcAccountIDType            AccountID;          //资金账户 ID       (预留)
    TX1FtdcInstrumentIDType         InstrumentID;       //合约代码          (预留)
    TX1FtdcExchangeIDType           ExchangeID;         //交易所            (预留)
    TX1FtdcBuySellTypeType          BuySellType;        //买卖              (预留)
    TX1FtdcOpenCloseTypeType        OpenCloseType;      //开平              (预留)
    TX1FtdcInstrumentTypeType       InstrumentType;     //合约类型          (预留)
    TX1FtdcSpeculatorType           Speculator;         //投资类别          (预留)
    TX1FtdcPriceType                InsertPrice;        //委托价            (预留)
    TX1FtdcPriceType                ProfitLossPrice;    //止盈止损价格      (预留)
    TX1FtdcAmountType               MinMatchAmount;     //最小成交量        (预留)
    TX1FtdcAmountType               OrderAmount;        //委托数量          (预留)
    TX1FtdcInsertType               InsertType;         //自动单类别        (预留)
    TX1FtdcOrderTypeType            OrderType;          //订单类型          (预留)
    TX1FtdcOrderPropertyType        OrderProperty;      //订单属性          (预留)
    TX1FtdcClientIDType             ClientID;           //交易编码          (预留)
};
```

撤单发生错误时，需要检查 pErrorInfo 是否为 NULL。且客户端收到该响应时，只表示柜台确认了这笔撤单请求。

6.2.7. OnRtnErrorMsg 方法

API 内部或者柜台内部发生异常错误，通过此方法通知用户。
API 内部错误：包含 API 内存数据异常、系统资源申请异常等。
柜台内存错误：包含交易所 API 内部抛出异常，柜台内部数据异常，系统资源申请异常等。

函数原型：

```
void OnRtnErrorMsg( struct CX1FtdcRspErrorField*pErrorInfo );
```

6.2.8. OnRtnMatchedInfo 方法

成交回报，当委托成功交易后此方法会被调用。

函数原型：

```
void OnRtnMatchedInfo( struct CX1FtdcRspPriMatchInfoField*pRtnMatchData )；
```

参数：

CX1FtdcRspPriMatchInfoField：指向成交回报的结构。成交回报数据结构：

```
struct CX1FtdcMatchRtnField
{
    TX1FtdcLocalOrderIDType          LocalOrderID;     //本地委托号
    TX1FtdcOrderSysIDType            OrderSysID;       //报单编号
    TX1FtdcMatchIDType               MatchID;          //成交编号
    TX1FtdcInstrumentIDType          InstrumentID;     //合约代码
    TX1FtdcBuySellTypeType           BuySellType;      //买卖
    TX1FtdcOpenCloseTypeType         OpenCloseType;    //开平标志
    TX1FtdcPriceType                 MatchedPrice;     //成交价格
    TX1FtdcAmountType                OrderAmount;      //委托数量        (预留)
    TX1FtdcAmountType                MatchedAmount;    //成交数量
    TX1FtdcDateType                  MatchedTime;      //成交时间
    TX1FtdcPriceType                 InsertPrice;      //报价            (预留)
    TX1FtdcX1OrderIDType             X1OrderID;        //柜台委托号
    TX1FtdcMatchType                 MatchType;        //成交类型
    TX1FtdcSpeculatorType            Speculator;       //投保
    TX1FtdcExchangeIDType            ExchangeID;       //交易所 ID
    TX1FtdcFeeType                   Fee;              //手续费          (预留)
    TX1FtdcSessionIDType             SessionID;        //会话标识
    TX1FtdcInstrumentTypeType        InstrumentType;   //合约类型        (预留)
    TX1FtdcAccountIDType             AccountID;        //资金账号        (预留)
    TX1FtdcOrderAnswerStatusType     OrderStatus;      //申报结果        (预留)
```

```
    TX1FtdcPriceType            Margin;                          (预留)
    //开仓为保证金,平仓为解冻保证金
    TX1FtdcPriceType            FrozenCapita;                    (预留)
    //成交解冻委托冻结的资金
    TX1FtdcAdjustmentInfoType   AdjustmentInfo;                  (预留)
    //组合或对锁的保证金调整信息,格式:[合约代码,买卖标志,投资类别,调整金额;]
    TX1FtdcCustomCategoryType   CustomCategory;  //自定义类别    (预留)
    TX1FtdcPriceType            Turnover;        //成交金额      (预留)
    TX1FtdcOrderTypeType        OrderType;       //报单类型      (预留)
    TX1FtdcInsertType           InsertType;      //自动单类别    (预留)
    TX1FtdcClientIDType         ClientID;        //交易编码      (预留)
}
```

6.2.9. OnRtnOrder 方法

下单委托成功后，或交易所拒绝报单时此方法会被调用。

函数原型：

```
void OnRtnOrder( struct CX1FtdcRspPriOrderField*pRtnOrderData )；
```

参数：

pRtnOrderData：指向委托回报地址。委托回报数据结构：

```
struct CX1FtdcRspPriOrderField
{
    TX1FtdcLocalOrderIDType     LocalOrderID;        //本地委托号
    TX1FtdcX1OrderIDType        X1OrderID;           //柜台委托号
    TX1FtdcOrderSysIDType       OrderSysID;          //报单编号
    TX1FtdcOrderAnswerStatusType OrderStatus;        //委托状态
    TX1FtdcSessionIDType        SessionID;           //会话 ID
    TX1FtdcDateType             SuspendTime;         //挂起时间      (预留)
    TX1FtdcInstrumentIDType     InstrumentID;        //合约代码
    TX1FtdcExchangeIDType       ExchangeID;          //交易所
    TX1FtdcBuySellTypeType      BuySellType;         //买卖
    TX1FtdcOpenCloseTypeType    OpenCloseType;       //开平
    TX1FtdcInstrumentTypeType   InstrumentType;      //合约类型
    TX1FtdcSpeculatorType       Speculator;          //投资类别
    TX1FtdcPriceType            InsertPrice;         //委托价
    TX1FtdcPriceType            ProfitLossPrice;     //止盈止损价格  (预留)
    TX1FtdcAccountIDType        AccountID;           //资金账号      (预留)
    TX1FtdcAmountType           CancelAmount;        //撤单数量
    TX1FtdcAmountType           OrderAmount;         //委托数量
    TX1FtdcInsertType           InsertType;          //自动单类别    (预留)
    TX1FtdcOrderTypeType        OrderType;           //报单类型      (预留)
```

```
    TX1FtdcX1OrderIDType        ExtX1OrderID;       //算法单编号        (预留)
    TX1FtdcReservedType         ReservedType2;      //预留字段 2        (预留)
    TX1FtdcCustomCategoryType   CustomCategory;     //自定义类别        (预留)
    TX1FtdcOrderPropertyType    OrderProperty;      //订单属性          (预留)
    TX1FtdcAmountType           MinMatchAmount;     //最小成交量        (预留)
    TX1FtdcClientIDType         ClientID;           //交易编码          (预留)
    TX1FtdcErrorIDType          ErrorID;            //错误 ID
    TX1FtdcErrorMsgInfoType     StatusMsg;          //状态信息
    TX1FtdcExtOrderType         ExtOrderType;       //条件单类型        (预留)
};
```

注: LocalOrderID 在系统重启后，尽量恢复，不确保准确。

OrderStatus 在 FAK FOK 情况返回交易所已经接受，但尚未成交 8，失败返回 7，其他情况一律返回未成交 3

6.2.10. OnRtnCancelOrder 方法

当撤单成功后或撤单被交易所拒绝时该方法会被调用。

函数原型：

```
void OnRtnCancelOrder(struct CX1FtdcRspPriCancelOrderField* pCancelOrderData)
```

参数：

pCancelOrderData：指向撤单回报结构。撤单回报结构

```
struct CX1FtdcRspPriCancelOrderField
{
    TX1FtdcLocalOrderIDType         LocalOrderID;       //本地委托号
    TX1FtdcOrderSysIDType           OrderSysID;         //报单编号
    TX1FtdcInstrumentIDType         InstrumentID;       //合约代码          (预留)
    TX1FtdcPriceType                InsertPrice;        //报单价格          (预留)
    TX1FtdcBuySellTypeType          BuySellType;        //买卖类型          (预留)
    TX1FtdcOpenCloseTypeType        OpenCloseType;      //开平标志          (预留)
    TX1FtdcAmountType               CancelAmount;       //撤单数量
    TX1FtdcX1OrderIDType            X1OrderID;          //柜台委托号
    TX1FtdcSpeculatorType           Speculator;         //投保(预留)
    TX1FtdcExchangeIDType           ExchangeID;         //交易所 ID         (预留)
    TX1FtdcDateType                 CanceledTime;       //撤单时间
    TX1FtdcSessionIDType            SessionID;          //会话标识
    TX1FtdcOrderAnswerStatusType    OrderStatus;        //申报结果
    TX1FtdcInstrumentTypeType       InstrumentType;     //合约类型          (预留)
    TX1FtdcAccountIDType            AccountID;          //资金账号          (预留)
    TX1FtdcAmountType               OrderAmount;        //委托数量          (预留)
    TX1FtdcPriceType                Margin;             //保证金            (预留)
```

```
    TX1FtdcPriceType            Fee;              //手续费          (预留)
    TX1FtdcCustomCategoryType   CustomCategory;   //自定义类别      (预留)
    TX1FtdcPriceType            ProfitLossPrice;  //止盈止损价格    (预留)
    TX1FtdcAmountType           MinMatchAmount;   //最小成交量      (预留)
    TX1FtdcInsertType           InsertType;       //自动单类别      (预留)
    TX1FtdcClientIDType         ClientID;         //交易编码        (预留)
    TX1FtdcErrorIDType          ErrorID;          //错误 ID
    TX1FtdcErrorMsgInfoType     StatusMsg;        //状态信息
    TX1FtdcOrderPropertyType    OrderProperty;    //报单附加属性    (预留)
};
```

6.2.11. OnRspQryOrderInfo 方法

查询当日委托响应，当用户发出委托查询后，该方法会被调用。

函数原型：

```
void OnRspQryOrderInfo(struct CX1FtdcRspOrderField* pRtnOrderData, struct
CX1FtdcRspErrorField * pErrorInfo, bool bIsLast)
```

参数：

bIsLast：表明是否是最后一条响应信息

pRtnOrderData：指向委托回报结构。委托回报数据结构

```
struct CX1FtdcRspOrderField
{
    TX1FtdcRequestIDType        RequestID;        //请求 ID
    TX1FtdcX1OrderIDType        X1OrderID;        //柜台委托号
    TX1FtdcOrderAnswerStatusType OrderStatus;     //委托状态
    TX1FtdcInstrumentIDType     InstrumentID;     //合约代码
    TX1FtdcBuySellTypeType      BuySellType;      //买卖
    TX1FtdcOpenCloseTypeType    OpenClose;        //开平标志
    TX1FtdcPriceType            InsertPrice;      //委托价
    TX1FtdcAmountType           OrderAmount;      //委托数量
    TX1FtdcPriceType            MatchedPrice;     //成交价格
    TX1FtdcAmountType           MatchedAmount;    //成交数量
    TX1FtdcAmountType           CancelAmount;     //撤单数量
    TX1FtdcInsertType           InsertType;       //自动单类别
    TX1FtdcSpeculatorType       Speculator;       //投保
    TX1FtdcDateType             CommTime;         //委托时间
    TX1FtdcDateType             SubmitTime;       //申报时间
    TX1FtdcClientIDType         ClientID;         //交易编码
    TX1FtdcExchangeIDType       ExchangeID;       //交易所 ID
    TX1FtdcFrontAddrType        OperStation;      //委托地址
    TX1FtdcAccountIDType        AccountID;        //客户号
```

```
    TX1FtdcInstrumentTypeType       InstrumentType;         //合约类型
    TX1FtdcSessionIDType            SessionId;              //会话 ID          (预留)
    TX1FtdcReservedType             ReservedType2;          //预留字段 2
    TX1FtdcOrderSysIDType           OrderSysID;             //报单编号
    TX1FtdcCustomCategoryType       CustomCategory;         //自定义类别
    TX1FtdcPriceType                Margin;                 //保证金
    TX1FtdcPriceType                Fee;                    //手续费
    TX1FtdcLocalOrderIDType         LocalOrderID;           //本地委托号
    TX1FtdcOrderTypeType            OrderType;              //报单类型
    TX1FtdcOrderPropertyType        OrderProperty;          //订单属性
};
```

6.2.12. OnRspQryPosition 方法

查询持仓响应。当用户发出持仓查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspQryPosition( struct CX1FtdcRspPositionField*pPositionInfoRtn,struct
CX1FtdcRspErrorField *pErrorInfo, bool bIsLast );
```

参数：

bIsLast:表示是否是最后一条消息

pPositionInfoRtn：返回持仓信息结构地址。持仓信息结构：

```
struct CX1FtdcRspPositionField
{
    TX1FtdcRequestIDType            RequestID;              //请求 ID
    TX1FtdcAccountIDType            AccountID;              //资金帐号 ID
    TX1FtdcExchangeIDType           ExchangeID;             //交易所代码
    TX1FtdcInstrumentIDType         InstrumentID;           //合约号
    TX1FtdcBuySellTypeType          BuySellType;            //买卖
    TX1FtdcPriceType                OpenAvgPrice;           //开仓均价
    TX1FtdcPriceType                PositionAvgPrice;       //持仓均价
    TX1FtdcAmountType               PositionAmount;         //持仓量
    TX1FtdcAmountType               TotalAvaiAmount;        //总可用
    TX1FtdcAmountType               TodayAvaiAmount;        //今可用
    TX1FtdcAmountType               LastAvaiAmount;         //昨可用
    TX1FtdcAmountType               TodayAmount;            //今仓
    TX1FtdcAmountType               LastAmount;             //昨仓
    TX1FtdcAmountType               TradingAmount;          //平今挂单量
    TX1FtdcProfitLossType           DatePositionProfitLoss; //盯市持仓盈亏
    TX1FtdcProfitLossType           DateCloseProfitLoss;    //盯市平仓盈亏
    TX1FtdcProfitLossType           Premium;                //权利金
    TX1FtdcProfitLossType           ProfitLoss;             //浮动盈亏
```

```
    TX1FtdcProfitLossType          Margin;                  //占用保证金
    TX1FtdcSpeculatorType          Speculator;              //投保类别
    TX1FtdcClientIDType            ClientID;                //交易编码
    TX1FtdcPriceType               PreSettlementPrice;      //昨结算价
    TX1FtdcInstrumentTypeType      InstrumentType;          //合约类型
    TX1FtdcAmountType              YesterdayTradingAmount;  //平昨挂单量
};
```

6.2.13. OnRspQryPositionDetail 方法

查询持仓明细响应。当用户发出持仓明细查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspQryPositionDetail( struct CX1FtdcRspPositionDetailField* pPositionDetailRtn,
struct CX1FtdcRspErrorField * pErrorInfo, bool bIsLast );
```

参数：

bIsLast:表示是否是最后一条消息

pPositionDetailRtn：返回持仓明细信息结构地址。持仓明细信息结构：

```
struct CX1FtdcRspPositionDetailField
{
    TX1FtdcRequestIDType           RequestID;               //请求 ID
    TX1FtdcAccountIDType           AccountID;               //资金帐号 ID
    TX1FtdcExchangeIDType          ExchangeID;              //交易所代码
    TX1FtdcInstrumentIDType        InstrumentID;            //合约号
    TX1FtdcBuySellTypeType         BuySellType;             //买卖
    TX1FtdcPriceType               OpenPrice;               //开仓价
    TX1FtdcAmountType              Volume;                  //手数
    TX1FtdcMatchIDType             MatchID;                 //成交编号
    TX1FtdcDateType                MatchedDate;             //成交日期
    TX1FtdcProfitLossType          DatePositionProfitLoss;  //盯市持仓盈亏
    TX1FtdcProfitLossType          DateCloseProfitLoss;     //盯市平仓盈亏
    TX1FtdcProfitLossType          ProfitLoss;              //浮动盈亏
    TX1FtdcProfitLossType          Margin;                  //占用保证金
    TX1FtdcSpeculatorType          Speculator;              //投保类别
    TX1FtdcClientIDType            ClientID;                //交易编码
    TX1FtdcPriceType               PreSettlementPrice;      //昨结算价
    TX1FtdcInstrumentTypeType      InstrumentType;          //合约类型
    TX1FtdcX1OrderIDType           X1OrderID;               //柜台委托号
    TX1FtdcCustomCategoryType      CustomCategory;          //自定义类别
    TX1FtdcAmountType              CloseOrderVol;           //平仓委托数量
    TX1FtdcAmountType              CloseMatchVol;           //平仓成交数量
    TX1FtdcPositionDateType        PositionDateType;        //持仓日期类型
};
```

6.2.14. OnRspQryMatchInfo 方法

查询当日成交信息响应，当用户发出成交查询后该方法会被调用。

函数原型：

```
void OnRspQryMatchInfo(struct CX1FtdcRspMatchField* pRtnMatchData, struct
CX1FtdcRspErrorField * pErrorInfo, bool bIsLast){};
```

参数：

bIsLast：表明是否是最后一条响应信息。

pRtnMatchData：指向成交回报地址。成交回报结构：

SessionId 此版本不支持

```
struct CX1FtdcRspMatchField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcX1OrderIDType        X1OrderID;          //柜台委托号
    TX1FtdcAccountIDType        AccountID;          //资金账号
    TX1FtdcExchangeIDType       ExchangeID;         //交易所 ID
    TX1FtdcInstrumentIDType     InstrumentID;       //合约代码
    TX1FtdcBuySellTypeType      BuySellType;        //买卖
    TX1FtdcOpenCloseTypeType    OpenClose;          //开平
    TX1FtdcPriceType            MatchedPrice;       //成交价格
    TX1FtdcAmountType           MatchedAmount;      //成交数量
    TX1FtdcPriceType            MatchedMort;        //成交金额
    TX1FtdcSpeculatorType       Speculator;         //投保类别
    TX1FtdcDateType             MatchedTime;        //成交时间
    TX1FtdcMatchIDType          MatchedID;          //成交编号
    TX1FtdcLocalOrderIDType     LocalOrderID;       //本地委托号
    TX1FtdcClientIDType         ClientID;           //交易编码
    TX1FtdcMatchType            MatchType;          //成交类型
    TX1FtdcInstrumentTypeType   InstrumentType;     //合约类型
    TX1FtdcSessionIDType        SessionId;          //会话 ID      (预留)
    TX1FtdcReservedType         ReservedType2;      //预留字段 2
    TX1FtdcCustomCategoryType   CustomCategory;     //自定义类别
    TX1FtdcPriceType            Fee;                //手续费
    TX1FtdcOrderTypeType        OrderType;          //报单类型
    TX1FtdcOrderSysIDType       OrderSysID;         //报单编号
};
```

注：成交信息响应的会话号和本地号均为 0，自定义类别为空。

6.2.15. OnRspQrySpecifyInstrument 方法

查询指定合约信息响应，当用户发出查询制定合约指令后该方法会被调用。

函数参数：

```
void OnRspQrySpecifyInstrument(struct CX1FtdcRspSpecificInstrumentField* pInstrument,struct
CX1FtdcRspErrorField    * pErrorInfo, bool bIsLast)
```

参数：

pInstrument：指向返回的合约信息结构的地址，该结构如下：

```
Struct CX1FtdcRspSpecificInstrumentField
{
    TX1FtdcRequestIDType            RequestID;              //请求 ID
    TX1FtdcInstrumentIDType         InstrumentID;           //合约代码
    TX1FtdcRatioType                LongMarginRatio;        //多头保证金率
    TX1FtdcRatioType                ShortMarginRatio;       //空头保证金率
    TX1FtdcPriceType                LongMarginRatioByVolume;  //多头保证金费
    TX1FtdcPriceType                ShortMarginRatioByVolume; //空头保证金费
    TX1FtdcRatioType                OpenFeeVolRatio;        //开仓手续费
    TX1FtdcRatioType                CloseFeeVolRatio;       //平仓手续费
    TX1FtdcRatioType                CloseTodayFeeVolRatio;  //平今手续费
    TX1FtdcRatioType                OpenFeeAmtRatio;        //开仓手续费率
    TX1FtdcRatioType                CloseFeeAmtRatio;       //平仓手续费率
    TX1FtdcRatioType                CloseTodayFeeAmtRatio;  //平今手续费率
    TX1FtdcInstrumentTypeType       OrderTopLimit;          //委托上限
    TX1FtdcPriceType                ContractMultiplier;     //合约乘数
    TX1FtdcPriceType                MinimumPriceChange;     //最小变动价位
    TX1FtdcInstrumentTypeType       InstrumentType;         //合约类型
    TX1FtdcInstrumentMaturityType   InstrumentMaturity;     //合约最后交易日
    TX1FtdcComputeModeType          ComputeMode;            //计算方式
    TX1FtdcPriceType                AtMoneyNorm;            //平值按定额(卖期权)
    TX1FtdcPriceType                UpperLimitPrice;        //涨停板价
    TX1FtdcPriceType                LowerLimitPrice;        //跌停板价
    TX1FtdcPriceType                PreClosePrice;          //昨收盘
    TX1FtdcPriceType                PreSettlementPrice;     //昨结算价
    TX1FtdcPriceType                SettlementPrice;        //结算价
    TX1FtdcAmountType               PreOpenInterest;        //昨持仓量
    TX1FtdcRatioType                OptExecRatio;           //期权：行权按比例
                                                            //期货：交割按比例
    TX1FtdcRatioType                OptExecRatioPerVol;     //期权：行权按定额
                                                            //期货：交割按定额
};
```

6.2.16. OnRspCustomerCapital 方法

资金查询应答。当用户发出资金查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspCustomerCapital(struct CX1FtdcRspCapitalField*pCapitalInfoRtn,struct
CX1FtdcRspErrorField *pErrorInfo );
```

参数：

pCapitalInfoRtn：返回资金信息结构地址。资金信息结构：

```
struct CX1FtdcRspCapitalField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcAccountIDType        AccountID;          //资金帐号
    TX1FtdcEquityType           PreEquity;          //上日权益
    TX1FtdcEquityType           TodayEquity;        //当日客户权益
    TX1FtdcProfitLossType       CloseProfitLoss;    //平仓盈亏
    TX1FtdcProfitLossType       PositionProfitLoss; //持仓盈亏
    TX1FtdcProfitLossType       FrozenMargin;       //委托冻结保证金
    TX1FtdcProfitLossType       Margin;             //持仓保证金
    TX1FtdcProfitLossType       Fee;                //当日手续费
    TX1FtdcProfitLossType       Available;          //可用资金
    TX1FtdcProfitLossType       Withdraw;           //可取资金
    TX1FtdcRiskDegreeType       RiskDegree;         //风险度
    TX1FtdcPremiumType          TodayPremiumIncome; //本日权利金收入
    TX1FtdcPremiumType          TodayPremiumPay;    //本日权利金付出
    TX1FtdcPremiumType          YesterdayPremium;   //昨权利金收付
    TX1FtdcMarketValueType      OptMarketValue;     //期权市值
    TX1FtdcProfitLossType       ProfitLoss;         //浮动盈亏
    TX1FtdcProfitLossType       TotalFundOut;       //总出金
    TX1FtdcProfitLossType       TotalFundIn;        //总入金
};
```

6.2.17. OnRspQryExchangeStatus 方法

交易所状态查询响应，当用户发出交易所状态查询指令后该方法会被调用。

函数参数：

```
void OnRspQryExchangeStatus(struct CX1FtdcRspExchangeStatusField* pRspExchangeStatusData)
```

参数：

pRspExchangeStatusData：返回交易所状态结构的地址，该结构如下：

```
struct CX1FtdcRspExchangeStatusField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcExchangeStatusType   ExchangeStatus;     //交易所状态
```

```
    TX1FtdcExchangeIDType          ExchangeID;          //交易所编码
};
```

备注：

若收不到开闭市信号，可先咨询期货公司运维同事，是否在报盘的配置中，打开了<PushJYZT Value="YES"/>的选项

6.2.18. OnRspQryExchangeInstrument 方法

合约查询请求应答。当用户发出合约查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void OnRspQryExchangeInstrument( struct
                        CX1FtdcRspExchangeInstrumentField*pInstrumentData,
struct CX1FtdcRspErrorField    *pErrorInfo, bool bIsLast );
```

参数：

bIsLast:表示是否是最后一条消息

pInstrumentData：返回合约信息结构地址。合约信息结构：

```
struct CX1FtdcRspExchangeInstrumentField
{
    TX1FtdcRequestIDType          RequestID;          //请求 ID
    TX1FtdcExchangeIDType         ExchangeID;         //交易所编码
    TX1FtdcInstrumentIDType       InstrumentID;       //合约代码
    TX1FtdcVarietyNameType        VarietyName;        //品种名称
    TX1FtdcInstrumentTypeType     InstrumentType;     //合约类型
    TX1FtdcInstrumentTypeType     OrderTopLimit;      //委托上限
    TX1FtdcAmountType             MktOrderTopLimit;   //市价委托上限
    TX1FtdcPriceType              ContractMultiplier; //合约乘数
    TX1FtdcPriceType              MinPriceFluctuation;//最小变动价位
    TX1FtdcInstrumentMaturityType InstrumentMaturity; //合约最后交易日
    TX1FtdcPriceType              UpperLimitPrice;    //涨停板价
    TX1FtdcPriceType              LowerLimitPrice;    //跌停板价
    TX1FtdcPriceType              PreClosePrice;      //昨收盘
    TX1FtdcPriceType              PreSettlementPrice; //昨结算价
    TX1FtdcPriceType              SettlementPrice;    //结算价
    TX1FtdcAmountType             PreOpenInterest;    //昨持仓量
    TX1FtdcInstrumentPrefixType   InstrumentPrefix;   //合约前缀
    TX1FtdcInstrumenExpirationDateType  InstrumenExpiration; //合约到期日
    TX1FtdcOptionTypeType         OptionType;         //期权类型
    TX1FtdcInstrumentIDType       Underlying;         //标的合约代码
    TX1FtdcOptionTypeType         OptionType;         //期权类型
    TX1FtdcPriceType              StrikePrice;        //执行价格
    TX1FtdcRiskDegreeType         ExchangeRiskDegree; //最低保障风险系数
    TX1FtdcPriceType              MinMargin;          //最小保证金
    TX1FtdcAmountType             TradeSize;          //期权开仓单位
```

```
};
```

6.2.19. OnRspQuoteInsert 方法

做市商报单响应，当用户发出做市商报单请求指令后该方法会被调用

函数参数：

```
void OnRspQuoteInsert(struct CX1FtdcQuoteRspField * pRspQuoteData, struct
CX1FtdcRspErrorField * pErrorInfo)
```

参数：

pRspQuoteData：返回做市商报单响应，该结构如下：

```
struct CX1FtdcQuoteRspField
{
    TX1FtdcLocalOrderIDType        LocalOrderID;      //本地委托号
    TX1FtdcX1OrderIDType           X1OrderID;         //柜台委托号
    TX1FtdcRequestIDType           RequestID;         //请求 ID
    TX1FtdcPriceType               Fee;               //手续费        (预留)
    TX1FtdcPriceType               Margin;            //保证金(仅报价使用)
    TX1FtdcDateType                OrderTime;         //委托时间      (预留)
    TX1FtdcOrderAnswerStatusType   OrderStatus;       //委托状态
    TX1FtdcCustomCategoryType      CustomCategory;    //自定义类别    (预留)
    TX1FtdcInstrumentIDType        InstrumentID;      //合约代码      (预留)
    TX1FtdcAccountIDType           AccountID;         //资金账号      (预留)
    TX1FtdcQuoteIDType             QuoteID;           //询价编号
    TX1FtdcSessionIDType           SessionID;         //会话 ID
    TX1FtdcClientIDType            ClientID;          //交易编码      (预留)
};
```

6.2.20. OnRtnQuoteInsert 方法

做市商报单回报，当用户发出做市商报单请求指令后该方法会被调用

函数参数：

```
void OnRtnQuoteInsert(struct CX1FtdcQuoteRtnField * pRtnQuoteData)
```

参数：

pRtnQuoteData：返回做市商报单回报，该结构如下：

```
struct CX1FtdcQuoteRtnField
{
    TX1FtdcExchangeIDType          ExchangeID;        //交易所
    TX1FtdcClientIDType            ClientID;          //交易编码
    TX1FtdcOrderSysIDType          OrderSysID;        //报单编号
```

```
    TX1FtdcInstrumentIDType          InstrumentID;          //合约代码
    TX1FtdcLocalOrderIDType          LocalOrderID;          //本地委托号
    TX1FtdcSeatCodeType              SeatCode;              //席位代码
    TX1FtdcOpenCloseTypeType         BuyOpenCloseType;      //开平标志（买）
    TX1FtdcOpenCloseTypeType         SellOpenCloseType;     //开平标志（卖）
    TX1FtdcSpeculatorType            Speculator;            //投资类别
    TX1FtdcAmountType                BuyOrderAmount;        //委托数量（买）
    TX1FtdcAmountType                SellOrderAmount;       //委托数量（卖）
    TX1FtdcPriceType                 BuyInsertPrice;        //委托价   （买）
    TX1FtdcPriceType                 SellInsertPrice;       //委托价   （卖）
    TX1FtdcX1OrderIDType             X1OrderID;             //柜台委托号
    TX1FtdcAccountIDType             AccountID;             //资金账号     (预留)
    TX1FtdcInstrumentTypeType        InstrumentType;        //合约类型
    TX1FtdcDateType                  SuspendTime;           //挂单时间     (预留)
    TX1FtdcEntrusTellerType          EntrusTeller;          //委托柜员
    TX1FtdcOrderAnswerStatusType     OrderStatus;           //委托状态
    TX1FtdcSessionIDType             SessionID;             //会话 ID
    TX1FtdcQuoteIDType               QuoteID;               //询价编号     (预留)
    TX1FtdcErrorMsgInfoType          ErrorMsg;              //错误信息
    TX1FtdcCustomCategoryType        CustomCategory;        //自定义类别   (预留)
};
```

备注：

返回的柜台委托号为第一边的柜台委托号，第二边的柜台委托号实际上是第一遍柜台委托号加 1，不需要用户关心。后续撤单操作指定这个柜台委托号来进行双边撤单，不支持只撤销其中一边的操作。

6.2.21. OnRtnQuoteMatchedInfo 方法

成交回报，当委托成功交易后此方法会被调用。

函数原型：

```
void OnRtnQuoteMatchedInfo(struct CX1FtdcQuoteMatchRtnField * pRtnQuoteMatchedData)；
```

参数：

CX1FtdcQuoteMatchRtnField：指向成交回报的结构。成交回报数据结构：

```
struct CX1FtdcQuoteMatchRtnField
{
        TX1FtdcExchangeIDType        ExchangeID;            //交易所 ID
        TX1FtdcClientIDType          ClientID;              //交易编码          (预留)
        TX1FtdcInstrumentIDType      InstrumentID;          //合约代码
        TX1FtdcSeatCodeType          SeatCode;              //席位代码
        TX1FtdcLocalOrderIDType      LocalOrderID;          //本地委托号
        TX1FtdcOpenCloseTypeType     OpenCloseType;         //开平标志
        TX1FtdcSpeculatorType        Speculator;            //投资类别
```

```
    TX1FtdcX1OrderIDType            X1OrderID;          //柜台委托号
    TX1FtdcOrderSysIDType           OrderSysID;         //报单编号
    TX1FtdcMatchIDType              MatchID;            //成交编号
    TX1FtdcAmountType               MatchedAmount;      //成交数量
    TX1FtdcPriceType                MatchedPrice;       //成交价格
    TX1FtdcAccountIDType            AccountID;          //资金账号          (预留)
    TX1FtdcPriceType                Turnover;           //成交金额          (预留)
    TX1FtdcEntrusTellerType         EntrusTeller;       //委托柜员          (预留)
    TX1FtdcDateType                 MatchedTime;        //成交时间
    TX1FtdcFeeType                  Fee;                //手续费            (预留)
    TX1FtdcPriceType                InsertPrice;        //委托价格          (预留)
    TX1FtdcAmountType               OrderAmount;        //委托数量          (预留)
    TX1FtdcOrderAnswerStatusType    OrderStatus;        //申报结果          (预留)
    TX1FtdcPriceType                Margin;                                 (预留)
    //开仓为保证金,平仓为解冻保证金
    TX1FtdcBuySellTypeType          BuySellType;        //买卖
    TX1FtdcAmountType               CloseTodayAmount;   //平今数量          (预留)
    TX1FtdcPriceType                ClosePrice;         //平仓金额          (预留)
    TX1FtdcPriceType                CloseTodayPrice;    //平今金额          (预留)
    TX1FtdcAdjustmentInfoType       AdjustmentInfo;                         (预留)
    //组合或对锁的保证金调整信息,格式:[合约代码,买卖标志,投资类别,调整金额;]
    TX1FtdcPriceType                FrozenCapita;                           (预留)
    //成交解冻委托冻结的资金
    TX1FtdcProfitLossType           DateCloseProfitLoss;//盯市平仓盈亏    (预留)
    TX1FtdcInstrumentTypeType       InstrumentType;     //合约类型
    TX1FtdcSessionIDType            SessionID;          //会话标识
    TX1FtdcLargeMarginDirectType    LargeMarginDirect;  //大边保证金方向  (预留)
    TX1FtdcQuoteIDType              QuoteID;            //询价编号          (预留)
    TX1FtdcCustomCategoryType       CustomCategory;     //自定义类别        (预留)
    TX1FtdcLocalOrderIDType         PriFlowNo;          //  私有流编号}
```

注：OrderStatus 字段为成交成功 12，全部成交与否用户自行计算。

6.2.22. OnRspQuoteCancel 方法

做市商撤单响应，当用户发出做市商撤单请求指令后该方法会被调用

函数参数：

```
void OnRspQuoteCancel(struct CX1FtdcQuoteRspField * pRspQuoteCanceledData, struct
CX1FtdcRspErrorField * pErrorInfo)
```

参数：

pRspQuoteCanceledData：返回做市商撤单响应，该结构如下：

```
struct CX1FtdcQuoteRspField
{

        TX1FtdcLocalOrderIDType         LocalOrderID;       //本地委托号
        TX1FtdcX1OrderIDType            X1OrderID;          //柜台委托号
        TX1FtdcRequestIDType            RequestID;          //请求 ID
        TX1FtdcPriceType                Fee;                //手续费        (预留)
        TX1FtdcPriceType                Margin;             //保证金(仅报价使用)
        TX1FtdcDateType                 OrderTime;          //委托时间      (预留)
        TX1FtdcOrderAnswerStatusType    OrderStatus;        //委托状态
        TX1FtdcCustomCategoryType       CustomCategory;     //自定义类别    (预留)
        TX1FtdcInstrumentIDType         InstrumentID;       //合约代码      (预留)
        TX1FtdcAccountIDType            AccountID;          //资金账号      (预留)
        TX1FtdcQuoteIDType              QuoteID;            //询价编号
        TX1FtdcSessionIDType            SessionID;          //会话 ID
        TX1FtdcClientIDType             ClientID;           //交易编码      (预留)

};
```

6.2.23. OnRtnQuoteCancelOrder 方法`

做市商撤单回报，当用户发出做市商撤单请求指令后该方法会被调用

函数参数：

```
void OnRtnQuoteCancel(struct CX1FtdcQuoteCanceledRtnField * pRtnQuoteCanceledData)
```

参数：

pRtnQuoteCanceledData：返回做市商撤单回报，该结构如下：

```
struct APISTRUCT CX1FtdcQuoteCanceledRtnField
{
    TX1FtdcExchangeIDType           ExchangeID;         //交易所            (预留)
    TX1FtdcClientIDType             ClientID;           //交易编码          (预留)
    TX1FtdcOrderSysIDType           OrderSysID;         //报单编号
    TX1FtdcInstrumentIDType         InstrumentID;       //合约代码          (预留)
    TX1FtdcLocalOrderIDType         LocalOrderID;       //本地委托号
    TX1FtdcSeatCodeType             SeatCode;           //席位代码
    TX1FtdcOpenCloseTypeType        BuyOpenCloseType;   //开平标志（买）    (预留)
    TX1FtdcOpenCloseTypeType        SellOpenCloseType;  //开平标志（卖）    (预留)
    TX1FtdcSpeculatorType           Speculator;         //投资类别          (预留)
    TX1FtdcX1OrderIDType            X1OrderID;          //柜台委托号
    TX1FtdcAccountIDType            AccountID;          //资金账号          (预留)
    TX1FtdcEntrusTellerType         EntrusTeller;       //委托柜员          (预留)
    TX1FtdcOrderAnswerStatusType    OrderStatus;        //委托状态
```

```
    TX1FtdcAmountType           CancelAmount;       //撤单数量
    TX1FtdcPriceType            Fee;                //解冻手续费        (预留)
    TX1FtdcPriceType            Margin;             //解冻保证金        (预留)
    TX1FtdcSessionIDType        SessionID;          //会话ID
    TX1FtdcBuySellTypeType      BuySellType;        //买卖标志          (预留)
    TX1FtdcQuoteIDType          QuoteID;            //询价编号
    TX1FtdcDateType             CanceledTime;       //撤单时间
    TX1FtdcCustomCategoryType   CustomCategory;     //自定义类别        (预留)
};
```

6.2.24. OnRspCancelAllOrder 方法

做市商批量撤单响应，当用户发出做市商批量撤单请求指令后该方法会被调用

函数参数：

```
void OnRspCancelAllOrder(struct CX1FtdcCancelAllOrderRspField *pRspCancelAllOrderData,
struct CX1FtdcRspErrorField * pErrorInfo)
```

参数：

pRspCancelAllOrderData：返回做市商批量撤单响应，该结构如下：

```
struct CX1FtdcCancelAllOrderRspField
{
    DFITCRequestIDType          lRequestID;         //请求 ID
    DFITCAccountIDType          accountID;          //资金账号          (预留)
    DFITCOrderAnswerStatusType  orderStatus;        //委托状态
};
```

6.2.25. OnRspForQuote 方法

询价响应，用户调用询价请求后该接口会被调用

函数参数：

```
void OnRspForQuote(struct CX1FtdcForQuoteRspField * pRspForQuoteData, struct
CX1FtdcRspErrorField * pErrorInfo)
```

参数：

pRspForQuoteData：返回用户询价响应，该结构如下：

```
struct APISTRUCT CX1FtdcForQuoteRspField
{
    TX1FtdcRequestIDType        RequestID;          //请求 ID
    TX1FtdcX1OrderIDType        X1OrderID;          //柜台委托号
    TX1FtdcDateType             CommTime;           //委托时间      (预留)
};
```

6.2.26. OnRtnForQuote 方法

询价回报，当交易所收到请求后该接口会调用

函数参数：

```
void OnRtnForQuote(struct CX1FtdcForQuoteRtnField * pRtnForQuoteData)
```

参数：

pRtnForQuoteData：返回用户询价回报，该结构如下：

```
struct APISTRUCT CX1FtdcForQuoteRtnField
{
    TX1FtdcX1OrderIDType            X1OrderID;      //柜台委托号
    TX1FtdcSessionIDType            sessionID;      //会话 ID
    TX1FtdcInstrumentIDType         instrumentID;   //合约代码
    TX1FtdcExchangeIDType           exchangeID;     //交易所
    TX1FtdcAccountIDType            accountID;      //资金账号
    TX1FtdcOrderAnswerStatusType    orderStatus;    //委托状态
};
```

6.2.27. OnRspQryQuoteOrderInfo 方法

查询应/报价响应，当用户发出查询应/报价委托的请求时该接口会调用

函数参数：

```
void OnRspQryQuoteOrderInfo(struct CX1FtdcQuoteOrderRtnField * pRtnQuoteOrderData, struct CX1FtdcRspErrorField * pErrorInfo, bool bIsLast)
```

参数：

pRtnQuoteOrderData：返回用户应/报价信息，该结构如下：

```
struct APISTRUCT CX1FtdcQuoteOrderRtnField
{
    TX1FtdcRequestIDType            RequestID;          //请求 ID
    TX1FtdcX1OrderIDType            X1OrderID;          //柜台委托号
    TX1FtdcOrderAnswerStatusType    OrderStatus;        //委托状态
    TX1FtdcInstrumentIDType         InstrumentID;       //合约代码
    TX1FtdcPriceType                Margin;             //保证金
    TX1FtdcPriceType                Fee;                //手续费
    TX1FtdcLocalOrderIDType         LocalOrderID;       //本地委托号
    TX1FtdcAccountIDType            AccountID;          //客户号
    TX1FtdcDateType                 CommTime;           //委托时间
    TX1FtdcDateType                 SubmitTime;         //申报时间
    TX1FtdcExchangeIDType           ExchangeID;         //交易所 ID
    TX1FtdcAmountType               BuyOrderAmount;     //委托数量（买）
    TX1FtdcAmountType               BuyMatchedAmount;   //成交数量（买）
    TX1FtdcAmountType               BuyCancelAmount;    //撤单数量（买）
    TX1FtdcPriceType                BuyInsertPrice;     //委托价格（买）
```

```
    TX1FtdcPriceType            BuyMatchedPrice;     //成交价格（买）
    TX1FtdcOpenCloseTypeType    BuyOpenCloseType;    //开平标志（买）
    TX1FtdcAmountType           SellOrderAmount;     //委托数量（卖）
    TX1FtdcAmountType           SellMatchedAmount;   //成交数量（卖）
    TX1FtdcAmountType           SellCancelAmount;    //撤单数量（卖）
    TX1FtdcPriceType            SellInsertPrice;     //委托价格（卖）
    TX1FtdcPriceType            SellMatchedPrice;    //成交价格（卖）
    TX1FtdcOpenCloseTypeType    SellOpenCloseType;   //开平标志（卖）
    TX1FtdcFrontAddrType        OperStation;         //操作站点
    TX1FtdcSessionIDType        SessionID;           //会话 ID
    TX1FtdcQuoteIDType          QuoteID;             //询价编号
    TX1FtdcCustomCategoryType   CustomCategory;      //自定义类别
};
```

6.2.28. OnRspQryQuoteNotice 方法

询价通知查询响应，当用户发出询价通知查询请求时该接口会调用

函数参数：

```
void OnRspQryQuoteNotice(struct CX1FtdcQryQuoteNoticeRtnField * pRtnQryQuoteNoticeData,
struct CX1FtdcRspErrorField * pErrorInfo, bool bIsLast)
```

参数：

pRtnQryQuoteNoticeData：返回询价通知信息，该结构如下：

```
struct APISTRUCT CX1FtdcQryQuoteNoticeRtnField
{
    TX1FtdcRequestIDType        RequestID;           //请求 ID
    TX1FtdcQuoteIDType          QuoteID;             //询价编号
    TX1FtdcExchangeIDType       ExchangeID;          //交易所
    TX1FtdcInstrumentIDType     InstrumentID;        //合约代码
    TX1FtdcSourceType           Source;              //来源
    TX1FtdcDateType             QuoteTime;           //询价时间
};
```

6.2.29. OnRspQryForQuote 方法

询价委托查询响应，当用户发出询价委托查询请求时该接口会调用

函数参数：

```
void OnRspQryForQuote(struct CX1FtdcQryForQuoteRtnField * pRtnQryForQuoteData, struct
CX1FtdcRspErrorField * pErrorInfo, bool bIsLast)
```

参数：

pRtnQryForQuoteData：返回询价委托信息，该结构如下：

```
struct APISTRUCT CX1FtdcQryForQuoteRtnField
{
```

```
    TX1FtdcRequestIDType            RequestID;          //请求 ID
    TX1FtdcAccountIDType            AccountID;          //资金账号
    TX1FtdcX1OrderIDType            X1OrderID;          //柜台委托号
    TX1FtdcInstrumentIDType         instrumentID;       //合约代码
    TX1FtdcExchangeIDType           exchangeID;         //交易所
    TX1FtdcDateType                 SuspendTime;        //挂起时间
    TX1FtdcOrderAnswerStatusType    orderStatus;        //委托状态
};
```

6.2.30. OnRtnExchangeStatus 方法

交易所状态通知，当交易所有交易状态变化后该方法会被调用。

函数参数：

```
void OnRtnExchangeStatus(struct CX1FtdcExchangeStatusRtnField * pRtnExchangeStatusData)
```

参数：

pRtnExchangeStatusData：返回交易所状态通知结构的地址，该结构如下：

```
struct APISTRUCT CX1FtdcExchangeStatusRtnField
{
    TX1FtdcExchangeIDType           exchangeID;         //交易所
    TX1FtdcInstrumentIDType         instrumentID;       //合约代码
    TX1FtdcExchangeStatusType       exchangeStatus;     //交易所状态
};
```

备注：支持交易所状态通知和品种状态通知，在 x1 柜台可配。

6.2.31. OnRspArbitrageInstrument 方法

套利合约查询应答。当用户发出套利合约查询指令后，前置返回响应时该方法会被调用。

函数原型：

```
void CX1FtdcXspeedTraderSpi::OnRspArbitrageInstrument(struct DFITCAbiInstrumentRtnField *
pAbiInstrumentData, struct DFITCErrorRtnField * pErrorInfo, bool bIsLast)
```

参数：

bIsLast:表示是否是最后一条消息

pAbiInstrumentData：返回合约信息结构地址。合约信息结构：

```
struct APISTRUCT CX1FtdcAbiInstrumentRtnField
{
    TX1FtdcRequestIDType            lRequestID;         //请求 ID
    TX1FtdcExchangeIDType           exchangeID;         //交易所编码
    TX1FtdcInstrumentIDType         InstrumentID;       //合约代码
```

```
    TX1FtdcInstrumentNameType    instrumentName;     //合约名称
    TX1FtdcPriceType             upperLimitPrice;    //涨停板价
    TX1FtdcPriceType             lowerLimitPrice;    //跌停板价
    TX1FtdcPriceType             priceTick;          //最小变动价位
};
```

7. 开发样例

x1_dfitc_api_test_demo.cpp

具体样例请查看发布包中 demo 工程。