迅投Python版本文档说明v1.6

修改记录

文档版本	修订日期	修订作者	修订内容
1.5	2023/12/06	代鹏	新建api文档
1.6	2023/12/19	代鹏	新增市场状态查询，新增枚举名称查询

目录

1. 入门篇

1.1. 迅投Python版本XtTraderApi FAQ

1.1.1. XtTraderApi能提供哪些服务

XtTraderApi是基于迅投统一交易衍生出来的一套完善的Python策略交易框架，对外以Python库的形式提供策略交易所需要的API接口。

1.1.2. XtTraderApi运行依赖环境

XtTraderApi需要连接统一交易服务器上的XtApiService服务, 在运行XtTraderApi的程序前需要先保证迅投统一交易服务正常。

1.2. 常见问题

1.2.1. onUserLogin回调里的error包含错误信息: End of file

原因：是由于API本地的配置文件traderApi.ini里的isUseSSL和一体化服务器上/home/rzrk/server/config/xtapiservice.lua里的isUseSSL的值不匹配导致的。

解决办法：把两者都修改为0，然后重启API程序和一体化服务器上的XtApiService服务即可。把两者都修改为0，然后重启API程序和一体化服务器上的XtApiService服务即可。

1.2.2. onUserLogin回调里的error包含错误信息: md5不匹配禁止登录

原因：是由于一体化服务器上开启了API的md5校验导致的。

解决办法：修改一体化服务器上/home/rzrk/server/config/xtapiservice.lua里的isUseMD5Check的值修改为0，然后重启一体化服务器上的XtApiService服务即可。

1.2.3. onUserLogin回调里的error包含错误信息: 客户端mac不匹配

原因：是由于一体化服务器上开启了API的mac地址校验导致的。

解决办法：修改一体化服务器上/home/rzrk/server/config/xtapiservice.lua里的isUseMacCheck的值修改为0，然后重启一体化服务器上的XtApiService服务即可。

1.2.4. 周末要测试，但是账号休市

原因：周末因为股市休市，所以测试环境对于账号也是默认休市处理。

解决办法：在一体化管理端左侧菜单'系统监控->交易日设置'里，设置当天日期为交易日。如果调整之后，管理端上账号状态不是登录成功，可以在ET软件上对该账号关联的交易端执行'停用监控'操作后再'启用监控'。

1.2.5. 周一到周五盘后要测试，但是账号休市

原因：目前系统默认16:00账号休市，所以通常16:00以后账号会进入休市状态。

解决办法：在一体化管理端左侧菜单'基础管理->经纪公司'里，对要测试的经纪公司，点击设置来修改日盘时间即可。如果调整之后，管理端上账号状态不是登录成功，可以在ET软件上对该账号关联的交易端执行'停用监控'操作后再'启用监控'。

1.2.6. 运行python trade_demo.py的时候提示ImportError: DLL load failed: %1 不是有效的Win32应用程序。

原因：您使用的python是32位版本，需要使用32位版本的API。目前为了给客户最佳体验，迅投默认提供64位版本。

解决办法：

- 方法1：使用64位版本的Python即可。
- 方法2：如需使用32位版本可以把Win32.Release目录下的所有文件拷贝到和Win32.Release同一级目录，替换现有64位版本的库即可。

1.2.7. 近场交易说明

智能算法交易CIntelligentAlgorithmOrder和三方主动算法交易CExternAlgorithmOrder，采用近场交易，需要在m_eOrderStrategyType入参填写E_ORDER_STRATEGY_TYPE_APPROACH

1.3. 快速入门

1.3.1. 创建策略

```python
#coding=utf8

from XtTraderPyApi import *
import time

# CallBack类继承XtTraderApiCallback类，用于接收请求以及主推的回调
class CallBack(XtTraderApiCallback):
    def __init__(self, address, username, password, FundAccountId):
        super(CallBack, self).__init__()
        self.m_strAddress = address
        self.m_strUserName = username
        self.m_strPassword = password
        self.m_strFundAccountId = FundAccountId
        self.m_strAccountKey = ''
        self.m_nRequestId = 1
        self.m_client = None

    def init(self):
        if self.m_strAddress == None:
            return -1, u'server addr is empty'
        self.m_client = XtTraderApi.createXtTraderApi(self.m_strAddress)    #创建用户
        if self.m_client == None:
            return -1, u'failed to create traderapi client'
        if isinstance(self,XtTraderApiCallback):     #当前类是否为父类的实例
            self.m_client.setCallback(self)
            return self.m_client.init("../config")
        return 0, u''

    def join(self):
        self.m_client.join()

    #连接服务器的回调函数，参数success标识服务器连接是否成功，
    #参数errorMsg表示服务器连接失败的错误信息
    def onConnected(self, success, error_msg):
        print u'[onConnected] connect to server:', self.m_strAddress,\
         ', success: ', success, ', error_msg: ', error_msg
        if success:
            #machineInfo代表"IP地址|mac地址|磁盘序列号|cpu号|主机名|磁盘信息|
            #CPUID指令的ecx参数|主板信息"，如果相关字段无关，那就直接给个空值，
            #忽略相应字段就好，两个字段间用分隔符|隔开。
            machineInfo = "192.168.1.172|5C-F9-DD-73-7C-83|0682056C|BFEBFBFF000206A7\
            |DESKTOP-84OBV5E|C,NTFS,223|6VKJD3X"
```

```
        appid = "xt_api_2.0"
        authcode = "7f3c92e678f9ec77"
        self.m_client.userLogin(self.m_strUserName, self.m_strPassword, \
        self.m_nRequestId ,machineInfo, appid, authcode)
        self.m_nRequestId += 1

#用户登录的回调函数
def onUserLogin(self, userName, password, nRequestId, error):
    if error.isSuccess():
        print u'[onUserlogin] userName: ', username, ', 登录成功'
    else:
        print u'[onUserlogin] userName: ', username, ', 登录失败，失败原因: ',\
         error.errorMsg()

# 用户登出的回调函数 ,username用户名，password密码，error类
def onUserLogout(self, username, password, nRequestId, error):
    print u'[onUserLogout] success: ', error.isSuccess(), ', username: ',\
     username

# 账号key主推接口，账号登录成功后才可以执行下单等操作，可以根据这里的status字段做判断
# 参数: status 主推资金账号的登录状态
# 参数: account_type 主推资金账号的类型 1:期货账号，2:股票账号，3:信用账号
def onRtnLoginStatusWithActKey(self, accountID, status, account_type, accountKey,
error_msg):
    if status==EBrokerLoginStatus.BROKER_LOGIN_STATUS_OK:
        print u'[onRtnLoginStatusWithActKey] accountID: ',accountID,
        ', account_type: ', account_type, ",账号登录成功，初始化完成，可以交易"
        if account_type == 2:  # 股票账号
            self.m_strAccountKey = accountKey
            self.reqAccountDetail(accountID, self.m_strAccountKey)
            self.testStockOrdinaryOrder(accountID, accountKey)
        elif account_type == 1:  # 期货账号
            pass
        else:
            print u'[onRtnLoginStatusWithActKey] accountID: ', accountID, ',
            status: ', status, ', account_type: ', account_type
    else:
        print u'[onRtnLoginStatusWithActKey] accountID: ', accountID, \
        'status: ', status, ', account_type: ', account_type, ', \
        error_msg:', error_msg

# 股票普通单
```

```
def testStockOrdinaryOrder(self, accountID, accountKey):
    # 初始化数据，普通单结构体
    orderInfo = COrdinaryOrder()
    # 资金账号，必填参数。不填会被api打回，并且通过onOrder反馈失败
    orderInfo.m_strAccountID = accountID
    # 报单价格，默认为double最大值
    orderInfo.m_dPrice = 12.3
    # 单笔超价百分比，选填字段。默认为0
    orderInfo.m_dSuperPriceRate = 0
    # 报单委托量，必填字段。默认int最大值，填0或不填会被api打回
    orderInfo.m_nVolume = 100
     # 报单市场。必填字段。股票市场有"SH"/"SZ"，如果填空或填错都会被api直接打回
    orderInfo.m_strMarket = "SH"
    # 报单合约代码，必填字段
    orderInfo.m_strInstrument = "600000"
    # 枚举类型，报单价格类型，必填字段
    orderInfo.m_ePriceType = EPriceType.PRTP_FIX
    # 报单委托类型。必填字段
    orderInfo.m_eOperationType = EOperationType.OPT_BUY
    orderInfo.m_strRemark = "alpha"
    self.m_nRequestId += 1
    self.m_client.order(orderInfo, self.m_nRequestId, accountKey)
    print u'[testStockOrdinaryOrder]股票普通单, accountId:',
    accountID, ', requestId:', self.m_nRequestId

#期货普通单
def testFutureOrdinaryOrder(self, accountID, accountKey):
    # 初始化COrdinaryOrder数据 ，普通单结构体
    orderInfo = COrdinaryOrder()
    # 资金账号，必填参数。不填会被api打回，并且通过onOrder反馈失败
    orderInfo.m_strAccountID = accountID
    # 报单价格，默认为double最大值
    orderInfo.m_dPrice = 15705
    # 单笔超价百分比，选填字段。默认为0
    orderInfo.m_dSuperPriceRate = 0
    # 报单委托量，必填字段。默认int最大值，填0或不填会被api打回
    orderInfo.m_nVolume = 1
    # 报单市场。必填字段
    orderInfo.m_strMarket = "CZCE"
    # 报单合约代码，必填字段。
    orderInfo.m_strInstrument = "CF107"
    # 报单价格类型，必填字段
```

```
        orderInfo.m_ePriceType = EPriceType.PRTP_FIX
        # 报单委托类型。必填字段
        orderInfo.m_eOperationType = EOperationType.OPT_OPEN_LONG
        orderInfo.m_strRemark = "cta"
        self.m_nRequestId += 1
        self.m_client.order(orderInfo, self.m_nRequestId, accountKey)  # 下单
        print u'[testFutureOrdinaryOrder]期货普通单, accountId:',
         accountID, ', requestId:', self.m_nRequestId

    # 下达算法委托
    def testAlgorithmOrder(self, accountID, accountKey):
        #股票为例
        orderInfo = CAlgorithmOrder()   # 算法单实例
        orderInfo.m_strAccountID = accountID    #资金账号
        orderInfo.m_strMarket = "SH"  # 市场
        orderInfo.m_strInstrument = "600000"  # 合约代码
        orderInfo.m_eOperationType = EOperationType.OPT_BUY  # 委托类型
        orderInfo.m_ePriceType = EPriceType.PRTP_FIX  # 报价类型
        orderInfo.m_dPrice = 12.3  # 委托价格
        orderInfo.m_nVolume = 1000  # 委托数量
        # 单笔基准量，默认为目标量
        orderInfo.m_eSingleVolumeType = EVolumeType.VOLUME_FIX
        orderInfo.m_dSingleVolumeRate = 0.1 # 基准量比例
        orderInfo.m_dPlaceOrderInterval = 10 # 下单间隔
        orderInfo.m_dWithdrawOrderInterval = 10 #撤单间隔
        # 价格波动区间，必填字段 0 <= orderInfo.m_dPriceRangeRate <= 1
        orderInfo.m_dPriceRangeRate = 0.1
        orderInfo.m_nSingleNumMin = 100  # 单笔最小量，股票最小为100，期货最小为1
        orderInfo.m_nSingleNumMax = 1000 # 单笔最大量
        # 指令有效时间
        orderInfo.m_nValidTimeStart = 0
        orderInfo.m_nValidTimeEnd = orderInfo.m_nValidTimeStart + 1800
        orderInfo.m_nMaxOrderCount = 100  # 最大下单笔数
        orderInfo.m_strRemark = 'algorithm' # 投资备注
        self.m_nRequestId += 1
        self.m_client.order(orderInfo,self.m_nRequestId, accountKey) #开始下单
        print u'[testAlgorithmOrder]算法下单, accountID:',
        accountID, ', requestID:', self.m_nRequestId

    # 下达智能算法委托
    def testIntelligentAlgorithmOrder(self, accountID, accountKey):
        #股票为例
```

```
        orderInfo = CIntelligentAlgorithmOrder()    # 算法单实例
        orderInfo.m_strAccountID = accountID     #资金账号
        orderInfo.m_strMarket = "SH"  # 市场
        orderInfo.m_strInstrument = "600000"  # 合约代码
        orderInfo.m_eOperationType = EOperationType.OPT_BUY  # 委托类型
        orderInfo.m_ePriceType = EPriceType.PRTP_FIX  # 报价类型
        orderInfo.m_dPrice = 12.3  # 委托价格
        orderInfo.m_nVolume = 1000  # 委托数量
        orderInfo.m_strOrderType = "VWAP"
        # 指令有效时间
        orderInfo.m_nValidTimeStart = int(time.time())
        orderInfo.m_nValidTimeEnd = orderInfo.m_nValidTimeStart + 1800
        orderInfo.m_dMaxPartRate = 1  # 量比比例
        orderInfo.m_dMinAmountPerOrder = 100  # 委托最小金额
        orderInfo.m_strRemark = 'intelligent' # 投资备注
        self.m_nRequestId += 1
        self.m_client.order(orderInfo,self.m_nRequestId, accountKey)     #开始下单
        print u'[testIntelligentAlgorithmOrder]算法下单, accountID:'
        , accountID, ', requestID:', self.m_nRequestId

    #下达指令后的回调
    def onOrder(self, requestID, order_id, error):
        if error.isSuccess():
            print u'[onOrder] success:True',u', order_id:',order_id, u',
             requestID: ',requestID, u', error_msg: ',error.errorMsg()
            #self.cancel_cmd(orderId)
        else:
            print u'[onOrder] failed',u', order_id:',order_id, u',
             requestID: ',requestID, u', error_msg: ',error.errorMsg()

    # 获取主推的指令状态, data对应COrderInfo结构
    def onRtnOrder(self, data):
        if data.m_eStatus == EOrderCommandStatus.OCS_CHECKING:
            orderStatus = u'风控检查中'
        elif data.m_eStatus == EOrderCommandStatus.OCS_RUNNING:
            orderStatus = u'运行中'
        elif data.m_eStatus == EOrderCommandStatus.OCS_APPROVING:
            orderStatus = u'审批中'
        elif data.m_eStatus == EOrderCommandStatus.OCS_REJECTED:
            orderStatus = u'已驳回'
        elif data.m_eStatus == EOrderCommandStatus.OCS_RUNNING:
            orderStatus = u'运行中'
```

```
        elif data.m_eStatus == EOrderCommandStatus.OCS_CANCELING:
            orderStatus = u'撤销中'
        elif data.m_eStatus == EOrderCommandStatus.OCS_FINISHED:
            orderStatus = u'已完成'
        elif data.m_eStatus == EOrderCommandStatus.OCS_STOPPED:
            orderStatus = u'已撤销'
        else:
            orderStatus = u'默认状态:', data.m_eStatus
        print u'[onRtnOrder] 指令编号:', data.m_nOrderID, u', 账号:',\
         data.m_strAccountID, u', m_strRemark:', data.m_strRemark, \
         u',指令状态:', orderStatus, u', 成交量:', data.m_dTradedVolume, \
         u', 撤销者:', data.m_canceler, u', 指令执行信息:', data.m_strMsg

    # 获得主推的委托明细, data对应COrderDetail结构
    def onRtnOrderDetail(self, data):
        if data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_UNREPORTED:
            entrust_status = u'未报'
        elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_WAIT_REPORTING:
            entrust_status = u'待报'
        elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_REPORTED:
            entrust_status = u'已报'
        elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_REPORTED_CANCEL:
            entrust_status = u'已报待撤'
        elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_PARTSUCC_CANCEL:
            entrust_status = u'部成待撤'
        elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_PART_CANCEL:
            entrust_status = u'部撤'
        elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_CANCELED:
            entrust_status = u'已撤'
        elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_PART_SUCC:
            entrust_status = u'部成'
        elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_SUCCEEDED:
            entrust_status = u'已成'
        elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_JUNK:
            entrust_status = u'废单'
        else:
            entrust_status = u'默认状态->', data.m_eOrderStatus
        print u'[onRtnOrderDetail] 指令编号:', data.m_nOrderID, u', 投资备注:',
        data.m_strRemark, u', 合同编号:', data.m_strOrderSysID, \
            u', 委托状态:', entrust_status, u', 已成量:',data.m_nTradedVolume, \
            u', 成交均价:', data.m_dAveragePrice, u', 代码:', data.m_strInstrumentID, \
            u', ErrorID:', data.m_nErrorID,u', ErrorMsg:', data.m_strErrorMsg
```

```
# 获得主推的成交明细，data对应CDealDetail
def onRtnDealDetail(self, data):
    print u'[onRtnDealDetail] 指令编号:', data.m_nOrderID,  u'，投资备注:',\
    data.m_strRemark, data.m_nOrderID, u'，成交量:', data.m_nVolume, \
     u'，成交均价:', data.m_dAveragePrice

# 获得主推的委托错误信息，data对应COrderError结构，
# 在委托被迅投风控或者柜台驳回都会产生这个回调
def onRtnOrderError(self, data):
    if data == None:
        print u'[onRtnOrderError], data is None'
    else:
        print u'[onRtnOrderError] orderId: ', data.m_nOrderID, \
        u'm_nErrorID: ', data.m_nErrorID, u'm_strErrorMsg: ', \
        data.m_strErrorMsg, u'm_nRequestID:', data.m_nRequestID, \
        u'm_nOrderID:', data.m_nOrderID

#撤销指令
def cancel_cmd(self, order_id):
    print u'[cancel_cmd] 撤指令，指令编号:', order_id
    self.m_nRequestId += 1
    self.m_client.cancel(order_id,self.m_nRequestId)

#撤销指令回调
def onCancel(self, requestID, error):
    if error.isSuccess():
        print u'[onCancel] success'
    else:
        print u'[onCancel] failed, error_msg: ', error.errorMsg()

#撤销委托
def cancel_order(self, accountID, order_sysid, market, code, accountKey):
    print u'[cancel_order] 撤委托，合同编号:', accountID, ', ', order_sysid,\
    ', ', market,  ', ', code,  ', ', accountKey,  ', ', order_sysid
    self.m_nRequestId += 1
    self.m_client.cancelOrder(accountID, order_sysid, market, '',
    self.m_nRequestId, accountKey)

#撤销委托回调
def onCancelOrder(self, requestID, error):
```

```
    if error.isSuccess():
        print u'[onCancelOrder] success'
    else:
        print u'[onCancelOrder] failed, error_msg: ', error.errorMsg()

# 获得主推的撤单错误信息，data对应CCancelError结构
# 撤委托请求被迅投系统风控打回，或者撤销请求被柜台打回，均通过这个接口推送
def onRtnCancelError(self, data):
    print u'[onRtnCancelError]orderId:', data.m_nOrderID, u'm_nErrorID:',
     data.m_nErrorID, u'm_strErrorMsg:', data.m_strErrorMsg

# 请求账号资产数据
def reqAccountDetail(self, accountID, accountKey):
    print u'[reqAccountDetail]查询账号:', accountID, u"的资产数据"
    self.m_nRequestId += 1
    self.m_client.reqAccountDetail(accountID, self.m_nRequestId, accountKey)

#请求资金数据的回调函数，data对应CAccountDetail结构
def onReqAccountDetail(self,accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onReqAccountDetail] success, accountID:',accountID, \
        ', isLast:', isLast
    else:
        print u'[onReqAccountDetail] failed, accountID:', accountID, \
        ', error_msg:',error.errorMsg()

# 请求信用账号资产数据
def reqCreditDetail(self, accountID, accountKey):
    print u'[reqCreditAccountDetail]查询信用账号:', accountID, u"的资产数据"
    self.m_nRequestId += 1
    self.m_client.reqCreditDetail(accountID, self.m_nRequestId, accountKey)

#请求信用资产数据的回调函数，data对应CCreditDetail结构
def onReqCreditDetail(self,accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onReqCreditAccountDetail] success, accountID:', \
        accountID, 'isLast:', isLast
    else:
        print u'[onReqCreditAccountDetail] failed, accountID:', \
        accountID, ', error_msg:',error.errorMsg()

# 请求委托明细
```

```
def reqOrderDetail(self, accountID, accountKey):
    print u'[reqOrderDetail]查询账号:', accountID, u"的当日委托明细"
    self.m_nRequestId += 1
    self.m_client.reqOrderDetail(accountID, self.m_nRequestId, accountKey)

#委托明细查询的回调函数，data对应COrderDetail结构
def onReqOrderDetail(self,accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onReqOrderDetail] success, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onReqOrderDetail] failed, accountID:', \
        accountID, ', error_msg:',error.errorMsg()

# 请求成交明细
def reqDealDetail(self, accountID, accountKey):
    print u'[reqDealDetail]查询账号:', accountID, u"的当日成交明细"
    self.m_nRequestId += 1
    self.m_client.reqDealDetail(accountID, self.m_nRequestId, accountKey)

#成交明细查询的回调函数，data对应CDetalDetail结构
def onReqDealDetail(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onReqDealDetail] success, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onReqDealDetail] failed, accountID:', \
        accountID, ', error_msg:',error.errorMsg()

# 请求持仓明细
def reqPositionDetail(self, accountID, accountKey):
    print u'[reqPositionDetail]查询账号:', accountID, u"的持仓明细"
    self.m_nRequestId += 1
    self.m_client.reqPositionDetail(accountID, self.m_nRequestId, accountKey)

#请求持仓明细的回调函数，data对应CPositionDetail结构
def onReqPositionDetail(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onReqPositionDetail] success, accountID:', accountID, \
        ', isLast:', isLast
    else:
        print u'[onReqPositionDetail] failed, accountID:', accountID, \
```

```
        ', error_msg:',error.errorMsg()

    # 请求持仓统计
    def reqPositionStatics(self, accountID, accountKey):
        print u'[reqPositionStatics]查询账号:', accountID, u"的持仓明细"
        self.m_nRequestId += 1
        self.m_client.reqPositionStatics(accountID, self.m_nRequestId, accountKey)

    #请求持仓统计的回调函数，data对应CPositionStatics结构
    def onReqPositionStatics(self,accountID, requestID, data, isLast,error):
        if error.isSuccess():
            print u'[onReqPositionStatics] success, accountID:', accountID, \
            ', isLast:', isLast
        else:
            print u'[onReqPositionStatics] failed, accountID:', accountID, \
            ', error_msg:',error.errorMsg()

    #请求负债合约的回调函数
    def onReqStksubjects(self, accountID, requestID, data, isLast, error):
        if error.isSuccess():
            print u'[onReqStksubjects] success, accountID:', accountID, \
            ', isLast:', isLast
        else:
            print u'[onReqStksubjects] failed, accountID:', accountID, \
            ', error_msg:',error.errorMsg()

    #请求融资融券标的接口后的回调函数
    def onReqStkcompacts(self, accountID, requestID, data, isLast, error):
        if error.isSuccess():
            print u'[onReqStkcompacts] success, accountID:', accountID, \
            ', isLast:', isLast
        else:
            print u'[onReqStkcompacts] failed, accountID:', accountID, \
            ', error_msg:',error.errorMsg()

     # 请求行情数据的回调，data对应CPriceData结构
    def onReqPriceData(self, requestID, data, error):
        if error.isSuccess():
            print u'[onReqPriceData] success'
        else:
            print u'[onReqPriceData] failed, error_msg:', error.errorMsg()
```

```
    #订阅行情后的订阅回调，data对应CSubscribData结构
    def onSubscribQuote(self, requestID, data, error):
        print u'[onSubscribQuote] m_strInstrumentID:', data.m_strInstrumentID

    #行情订阅成功后的行情主推，data对应CPriceData结构
    def onRtnPriceData(self, data):
        print u'[onRtnPriceData] m_strInstrumentID:', data.m_strInstrumentID

if __name__ == '__main__':
    server_addr = "210.14.143.252:65300" # 统一交易服务器的地址
    username = "API优化独立投顾3" # 用户
    password = "@a123456" # 用户密码
    fund_account = "6000000083"    # 资金账号
    cb = CallBack(server_addr, username, password, fund_account)
    cb.init()
    cb.join()
```

2. 进阶篇

2.1. XtTraderApi运行逻辑

XtTraderApi封装了策略交易所需要的Python API接口, 可以和迅投统一交易服务交互，进行报单、撤单、查询资产、查询委托、 查询成交、查询持仓以及接收委托和成交的实时主推数据。

2.2. XtTraderApi数据类型和数据结构字典

2.2.1. 交易市场

枚举值	数据类型	含义
"SH"	str	上交所
"SZ"	str	深交所
"CFFEX"	str	股指期货

枚举值	数据类型	含义
"SHFE"	str	上海商品期货
"DCE"	str	大连商品期货
"CZCE"	str	郑州商品期货
"SHO"	str	上海期权
"SZO"	str	深圳期权
"HGT"	str	沪港通
"SGT"	str	深港通
"INE"	str	国际能源中心期货
"GFEX"	str	广州商品期货
"BJ"	str	北京证券交易所

2.2.2. 账号类型EXTBrokerType

枚举变量名	值	含义
AT_FUTURE	1	期货账号
AT_STOCK	2	股票账号
AT_CREDIT	3	信用账号
AT_GOLD	4	贵金属账号
AT_FUTURE_OPTION	5	期货期权账号
AT_STOCK_OPTION	6	股票期权账号
AT_HUGANGTONG	7	沪港通账号
AT_INCOME_SWAP	8	美股收益互换账号
AT_NEW3BOARD	10	全国股转账号
AT_SHENGANGTONG	11	深港通账号
AT_FICC_COMMODITY	14	期货电子盘账号

枚举变量名	值	含义
AT_FICC_INTEREST	15	利率电子盘账号
AT_ABROAD_FUTURE	16	外盘期货账号

2.2.3. 委托类型EOperationType

变量名	值	含义
OPT_INVALID	-1	无效值
OPT_OPEN_LONG	0	开多
OPT_CLOSE_LONG_HISTORY	1	平昨多
OPT_CLOSE_LONG_TODAY	2	平今多
OPT_OPEN_SHORT	3	开空
OPT_CLOSE_SHORT_HISTORY	4	平昨空
OPT_CLOSE_SHORT_TODAY	5	平今空
OPT_CLOSE_LONG_TODAY_FIRST	6	平多，优先平今
OPT_CLOSE_LONG_HISTORY_FIRST	7	平多，优先平昨
OPT_CLOSE_SHORT_TODAY_FIRST	8	平空，优先平今
OPT_CLOSE_SHORT_HISTORY_FIRST	9	平空，优先平昨
OPT_CLOSE_LONG_TODAY_HISTORY_THEN_OPEN_SHORT	10	开空，优先平今多
OPT_CLOSE_LONG_HISTORY_TODAY_THEN_OPEN_SHORT	11	开空，优先平昨多
OPT_CLOSE_SHORT_TODAY_HISTORY_THEN_OPEN_LONG	12	开多，优先平今空

变量名	值	含义
OPT_CLOSE_SHORT_HISTORY_TODAY_THEN_OPEN_LONG	13	开多，优先平昨空
OPT_SELL_PRIORITY_OPEN	14	卖出，优先开仓
OPT_BUY_PRIORITY_OPEN	15	买入，优先开仓
OPT_SELL_OPTIMAL_COMSSION	16	卖出，最优手续费
OPT_BUY_OPTIMAL_COMSSION	17	买入，最优手续费
OPT_BUY	18	买入，针对股票
OPT_SELL	19	卖出，针对股票
OPT_FIN_BUY	20	融资买入
OPT_SLO_SELL	21	融券卖出
OPT_BUY_SECU_REPAY	22	买券还券
OPT_DIRECT_SECU_REPAY	23	直接还券
OPT_SELL_CASH_REPAY	24	卖券还款
OPT_DIRECT_CASH_REPAY	25	直接还款
OPT_FUND_SUBSCRIBE	26	基金申购
OPT_FUND_REDEMPTION	27	基金赎回
OPT_FUND_MERGE	28	基金合并
OPT_FUND_SPLIT	29	基金分拆
OPT_PLEDGE_IN	30	质押入库
OPT_PLEDGE_OUT	31	质押出库

变量名	值	含义
OPT_OPTION_BUY_OPEN	32	买入开仓
OPT_OPTION_SELL_CLOSE	33	卖出平仓
OPT_OPTION_SELL_OPEN	34	卖出开仓
OPT_OPTION_BUY_CLOSE	35	买入平仓
OPT_OPTION_COVERED_OPEN	36	备兑开仓
OPT_OPTION_COVERED_CLOSE	37	备兑平仓
OPT_OPTION_CALL_EXERCISE	38	认购行权
OPT_OPTION_PUT_EXERCISE	39	认沽行权
OPT_OPTION_SECU_LOCK	40	证券锁定
OPT_OPTION_SECU_UNLOCK	41	证券解锁
OPT_FUTURE_OPTION_EXERCISE	50	期货期权行权
OPT_CONVERT_BONDS	51	可转债转股
OPT_SELL_BACK_BONDS	52	可转债回售
OPT_COLLATERAL_TRANSFER_IN	55	担保品划入
OPT_COLLATERAL_TRANSFER_OUT	56	担保品划出
OPT_ETF_PURCHASE	1004	ETF申购
OPT_ETF_REDEMPTION	1005	ETF赎回
OPT_AFTER_FIX_BUY	1043	盘后定价买入
OPT_AFTER_FIX_SELL	1044	盘后定价卖出
OPT_OPTION_COMB_EXERCISE	1089	组合行权
OPT_OPTION_BUILD_COMB_STRATEGY	1090	构建组合策略

变量名	值	含义
OPT_OPTION_RELEASE_COMB_STRATEGY	1091	解除组合策略
OPT_SLO_SELL_SPECIAL	1010	专项融券卖出
OPT_BUY_SECU_REPAY_SPECIAL	1011	专项买券还券
OPT_DIRECT_SECU_REPAY_SPECIAL	1012	专项直接还券
OPT_FIN_BUY_SPECIAL	1022	专项融资买入
OPT_SELL_CASH_REPAY_SPECIAL	1023	专项卖券还款
OPT_DIRECT_CASH_REPAY_SPECIAL	1024	专项直接还款

2.2.4. 报价类型EPriceType

变量名	值	含义
PRTP_INVALID	-1	无效值
PRTP_SALE5	0	卖5
PRTP_SALE4	1	卖4
PRTP_SALE3	2	卖3
PRTP_SALE2	3	卖2
PRTP_SALE1	4	卖1
PRTP_LATEST	5	最新价
PRTP_BUY1	6	买1
PRTP_BUY2	7	买2

变量名	值	含义
PRTP_BUY3	8	买3
PRTP_BUY4	9	买4
PRTP_BUY5	10	买5
PRTP_FIX	11	指定价
PRTP_MARKET	12	市价
PRTP_HANG	13	挂单价 跟盘价
PRTP_COMPETE	14	对手价
PRTP_MARKET_BEST	18	期货市价_最优价 郑商所
PRTP_MARKET_CANCEL	19	期货市价_即成剩撤 大商所
PRTP_MARKET_CANCEL_ALL	20	期货市价_全额成交或撤 大商所
PRTP_MARKET_CANCEL_1	21	期货市价_最优1档即成剩撤 中金所
PRTP_MARKET_CANCEL_5	22	期货市价_最优5档即成剩撤 中金所 上期所
PRTP_MARKET_CONVERT_1	23	期货市价_最优1档即成剩转 中金所
PRTP_MARKET_CONVERT_5	24	期货市价_最优5档即成剩转 中金所 上期所
PRTP_STK_OPTION_FIX_CANCEL_ALL	26	限价即时全部成交否则撤单 - 上海/深圳股票期权
PRTP_STK_OPTION_MARKET_CACEL_LEFT	27	市价_最优1档即时成交剩余撤单 上海股票期权市价
PRTP_STK_OPTION_MARKET_CANCEL_ALL	28	市价_即时全部成交否则撤单 上海股票期权市价
PRTP_STK_OPTION_MARKET_CONVERT_FIX	29	市价_剩余转限价 上海股票期权市价

变量名	值	含义
PRTP_MARKET_SH_CONVERT_5_CANCEL	42	市价_最优5档即时成交剩余撤销 上海股票市价
PRTP_MARKET_SH_CONVERT_5_LIMIT	43	市价_最优5档即时成交剩转限价 上海股票市价
PRTP_MARKET_PEER_PRICE_FIRST	44	市价_对手方最优价格委托 深圳股票期权和深圳股票市价，可用于上海科创板市价
PRTP_MARKET_MINE_PRICE_FIRST	45	市价_本方最优价格委托 深圳股票期权和深圳股票市价，可用于上海科创板市价
PRTP_MARKET_SZ_INSTBUSI_RESTCANCEL	46	市价_最优1档即时成交剩余撤销委托 深圳股票期权和深圳股票市价
PRTP_MARKET_SZ_CONVERT_5_CANCEL	47	市价_最优5档即时成交剩余撤销委托 深圳股票期权和深圳股票市价
PRTP_MARKET_SZ_FULL_REAL_CANCEL	48	市价_全额成交或撤销委托 深圳股票期权和深圳股票市价
PRTP_AFTER_FIX_PRICE	49	盘后定价申报
PRTP_OPTION_COMB_STRATEGY_CNSJC	50	股票期权组合保证金策略-认购牛市价差策略
PRTP_OPTION_COMB_STRATEGY_PXSJC	51	股票期权组合保证金策略-认沽熊市价差策略
PRTP_OPTION_COMB_STRATEGY_PNSJC	52	股票期权组合保证金策略-认沽牛市价差策略
PRTP_OPTION_COMB_STRATEGY_CXSJC	53	股票期权组合保证金策略-认购熊市价差策略
PRTP_OPTION_COMB_STRATEGY_KS	54	股票期权组合保证金策略-跨式空头

变量名	值	含义
PRTP_OPTION_COMB_STRATEGY_KKS	55	股票期权组合保证金策略-宽跨式空头
PRTP_OPTION_COMB_STRATEGY_ZBD	56	股票期权组合保证金策略-保证金开仓转备兑开仓
PRTP_OPTION_COMB_STRATEGY_ZXJ	57	股票期权组合保证金策略-备兑开仓转保证金开仓

2.2.5. 委托状态EEntrustStatus

变量名	值	含义
ENTRUST_STATUS_WAIT_END	0	委托状态已经在ENTRUST_STATUS_CANCELED或以上，但是成交数额还不够，等成交回报来
ENTRUST_STATUS_UNREPORTED	48	未报
ENTRUST_STATUS_WAIT_REPORTING	49	待报
ENTRUST_STATUS_REPORTED	50	已报
ENTRUST_STATUS_REPORTED_CANCEL	51	已报待撤
ENTRUST_STATUS_PARTSUCC_CANCEL	52	部成待撤
ENTRUST_STATUS_PART_CANCEL	53	部撤
ENTRUST_STATUS_CANCELED	54	已撤
ENTRUST_STATUS_PART_SUCC	55	部成
ENTRUST_STATUS_SUCCEEDED	56	已成
ENTRUST_STATUS_JUNK	57	废单
ENTRUST_STATUS_ACCEPT	58	已受理
ENTRUST_STATUS_CONFIRMED	59	已确认 担保品划转已确认状态
ENTRUST_STATUS_DETERMINED	86	已确认 协议回购已确认状态

变量名	值	含义
ENTRUST_STATUS_PREPARE_ORDER	87	预埋
ENTRUST_STATUS_PREPARE_CANCELED	88	预埋已撤
ENTRUST_STATUS_UNKNOWN	255	未知

2.2.6. 普通算法单笔基准量类型EVolumeType

枚举变量名	值	含义
EVolumeType.VOLUME_SALE12345	0	卖1到卖5量总和
EVolumeType.VOLUME_SALE1234	1	卖1到卖4量总和
EVolumeType.VOLUME_SALE123	2	卖1到卖3量总和
EVolumeType.VOLUME_SALE12	3	卖1到卖2量总和
EVolumeType.VOLUME_SALE1	4	卖1量
EVolumeType.VOLUME_BUY1	5	买1量
EVolumeType.VOLUME_BUY12	6	买1到买2量总和
EVolumeType.VOLUME_BUY123	7	买1到买3量总和
EVolumeType.VOLUME_BUY1234	8	买1到买4量总和
EVolumeType.VOLUME_BUY12345	9	买1到买5量总和
EVolumeType.VOLUME_FIX	10	目标量
EVolumeType.VOLUME_LEFT	11	目标剩余量
EVolumeType.VOLUME_POSITION	12	持仓数量

2.2.7. 智能算法类型m_strOrderType

算法名称	数据类型	含义
"VWAP"	str	VWAP
"TWAP"	str	TWAP

算法名称	数据类型	含义
"VP"	str	跟量
"PINLINE"	str	跟价
"FLOAT"	str	盘口
"ICEBERG"	str	冰山
"DMA"	str	快捷
"MOC"	str	尾盘
"MOO"	str	开盘
"SWITCH"	str	调仓
"STWAP"	str	分时加权平均价格
"VP+"	str	量加权平均价格（带限价）
"XTFAST"	str	市价单（融资融券）
"SLOS"	str	止损限价单
"SLOH"	str	止损市价单
"SNIPER"	str	狙击
"IS"	str	隔夜单
"VWAP+"	str	量加权平均价格（带限价）
"MOOPLUS"	str	零点单

2.2.8. 主动算法类型m_strOrderType

- 算法具体逻辑可以咨询客户经理

算法名称	类型	含义
VWAPA1	str	主动VWAP
VWAPPLUS	str	主动VWAPPLUS
VWAP_ALPHA	str	阿尔法VWAP

2.2.9. 登录状态EBrokerLoginStatus

枚举变量名	值	含义
EBrokerLoginStatus.BROKER_LOGIN_STATUS_OK	0	登录成功，初始化完成
EBrokerLoginStatus.BROKER_LOGIN_STATUS_WAITING_LOGIN	1	连接中
EBrokerLoginStatus.BROKER_LOGIN_STATUS_LOGINING	2	登录中
EBrokerLoginStatus.BROKER_LOGIN_STATUS_FAIL	3	失败
EBrokerLoginStatus.BROKER_LOGIN_STATUS_INITING	4	在初始化中
EBrokerLoginStatus.BROKER_LOGIN_STATUS_CORRECTING	5	数据刷新校正中
EBrokerLoginStatus.BROKER_LOGIN_STATUS_CLOSED	6	收盘后

2.2.10. 买卖方向EEntrustBS

变量名	值	含义
ENTRUST_BUY	48	买入
ENTRUST_SELL	49	卖出
ENTRUST_COVERED	50	备兑
ENTRUST_PLEDGE_IN	81	质押入库
ENTRUST_PLEDGE_OUT	66	质押出库

2.2.11. 指令状态EOrderCommandStatus

变量名	值	含义
OCS_CHECKING	-1	风控检查中
OCS_APPROVING	0	审批中

变量名	值	含义
OCS_REJECTED	1	已驳回
OCS_RUNNING	2	运行中
OCS_CANCELING	3	撤销中
OCS_FINISHED	4	已完成
OCS_STOPPED	5	已停止
OCS_FROCE_COMPLETED	6	强制撤销
OCS_CHECKFAILED	7	风控驳回

2.2.12. 委托类型EEntrustTypes

变量名	值	含义
ENTRUST_BUY_SELL	48	买卖
ENTRUST_QUERY	49	查询
ENTRUST_CANCEL	50	撤单
ENTRUST_APPEND	51	补单
ENTRUST_COMFIRM	52	确认
ENTRUST_BIG	53	大宗
ENTRUST_FIN	54	融资委托
ENTRUST_SLO	55	融券委托
ENTRUST_CLOSE	56	信用平仓
ENTRUST_CREDIT_NORMAL	57	信用普通委托
ENTRUST_CANCEL_OPEN	58	撤单补单
ENTRUST_TYPE_OPTION_EXERCISE	59	行权
ENTRUST_TYPE_OPTION_SECU_LOCK	60	锁定
ENTRUST_TYPE_OPTION_SECU_UNLOCK	61	解锁

变量名	值	含义
ENTRUST_TYPE_OPTION_COMB_EXERCISE	65	组合行权
ENTRUST_TYPE_OPTION_BUILD_COMB_STRATEGY	66	构建组合策略持仓
ENTRUST_TYPE_OPTION_RELEASE_COMB_STRATEGY	67	解除组合策略持仓

2.2.13. 期货交易类型EFutureTradeType

变量名	值	含义
FUTRUE_TRADE_TYPE_COMMON	48	普通成交
FUTURE_TRADE_TYPE_OPTIONSEXECUTION	49	期权成交 ptionsExecution
FUTURE_TRADE_TYPE_OTC	50	OTC成交
FUTURE_TRADE_TYPE_EFPDIRVED	51	期转现衍生成交
FUTURE_TRADE_TYPE_COMBINATION_DERIVED	52	组合衍生成交

2.2.14. 委托状态EEntrustStatus

变量名	值	含义
ENTRUST_STATUS_WAIT_END	0	委托状态已经在 ENTRUST_STATUS_CANCELED 或以上，但是成交数额还不够，等成交回报来
ENTRUST_STATUS_UNREPORTED	48	未报
ENTRUST_STATUS_WAIT_REPORTING	49	待报
ENTRUST_STATUS_REPORTED	50	已报
ENTRUST_STATUS_REPORTED_CANCEL	51	已报待撤
ENTRUST_STATUS_PARTSUCC_CANCEL	52	部成待撤
ENTRUST_STATUS_PART_CANCEL	53	部撤
ENTRUST_STATUS_CANCELED	54	已撤

变量名	值	含义
ENTRUST_STATUS_PART_SUCC	55	部成
ENTRUST_STATUS_SUCCEEDED	56	已成
ENTRUST_STATUS_JUNK	57	废单
ENTRUST_STATUS_ACCEPT	58	已受理
ENTRUST_STATUS_CONFIRMED	59	已确认 担保品划转已确认状态
ENTRUST_STATUS_DETERMINED	86	已确认 协议回购已确认状态
ENTRUST_STATUS_PREPARE_ORDER	87	预埋
ENTRUST_STATUS_PREPARE_CANCELED	88	预埋已撤
ENTRUST_STATUS_UNKNOWN	255	未知

2.2.15. 期货委托发送状态EEntrustSubmitStatus

变量名	值	含义
ENTRUST_SUBMIT_STATUS_InsertSubmitted	48	已经提交
ENTRUST_SUBMIT_STATUS_CancelSubmitted	49	撤单已经提交
ENTRUST_SUBMIT_STATUS_ModifySubmitted	50	修改已经提交
ENTRUST_SUBMIT_STATUS_OSS_Accepted	51	已经接受
ENTRUST_SUBMIT_STATUS_InsertRejected	52	报单已经被拒绝
ENTRUST_SUBMIT_STATUS_CancelRejected	53	撤单已经被拒绝
ENTRUST_SUBMIT_STATUS_ModifyRejected	54	改单已经被拒绝

2.2.16. 两融标的状态EXTSubjectsStatus

变量名	值	含义
SUBJECTS_STATUS_NORMAL	48	正常
SUBJECTS_STATUS_PAUSE	49	暂停

变量名	值	含义
SUBJECTS_STATUS_NOT	50	非标的

2.2.17. 两融负债状态EXTCompactStatus

变量名	值	含义
COMPACT_STATUS_ALL	32	不限制
COMPACT_STATUS_UNDONE	48	未归还
COMPACT_STATUS_PART_DONE	49	部分归还
COMPACT_STATUS_DONE	50	已归还
COMPACT_STATUS_DONE_BY_SELF	51	自行了结
COMPACT_STATUS_DONE_BY_HAND	52	手工了结
COMPACT_STATUS_NOT_DEBT	53	未形成负债
COMPACT_STATUS_EXPIRY_NOT_CLOSE	54	到期未平仓

2.2.18. 下单超过流控时处理方式EXtOverFreqOrderMode

变量名	值	含义
OFQ_FORBID	0	禁止
OFQ_QUEUE	1	队列

2.2.19. 停牌标志EXtSuspendedType

变量名	值	含义
XT_NO_SUSPENDED	0	不停牌
XT_SUSPENDED	1	停牌

2.2.20. 备兑标志ECoveredFlag

变量名	值	含义
COVERED_FLAG_FALSE	'0'	非备兑
COVERED_FLAG_TRUE	'1'	备兑

2.2.21. 期权持仓类型ESideFlag

变量名	值	含义
SIDE_FLAG_RIGHT	'0'	权利
SIDE_FLAG_DUTY	'1'	义务
SIDE_FLAG_COVERED	'2'	备兑

2.2.22. 订阅行情平台类型EXTOfferStatus

变量名	值	含义
XT_OFFER_STATUS_SP	48	实盘
XT_OFFER_STATUS_MN	49	模拟盘

2.2.23. 期货时间条件单类型ETimeCondition

变量名	值	含义
TIME_CONDITION_IOC	'1'	立即完成，否则撤销
TIME_CONDITION_GFS	'2'	本节有效
TIME_CONDITION_GFD	'3'	当日有效
TIME_CONDITION_GTD	'4'	指定日期前有效
TIME_CONDITION_GTC	'5'	撤销前有效
TIME_CONDITION_GFA	'6'	集合竞价有效

2.2.24. 期货数量条件单类型EVolumeCondition

变量名	值	含义
VOLUME_CONDITION_ANY	'1'	任何数量
VOLUME_CONDITION_MIN	'2'	最小数量
VOLUME_CONDITION_TOTAL	'3'	全部数量

2.2.25. 股票期权合约类型EOptionType

变量名	值	含义
OPTION_TYPE_CALL	48	认购合约
OPTION_TYPE_PUT	49	认沽合约

2.2.26. 币种EMoneyType

变量名	值	含义
MONEY_TYPE_RMB	48	人民币
MONEY_TYPE_USD	49	美元
MONEY_TYPE_HK	50	港币

2.2.27. 任务操作ETaskFlowOperation

变量名	值	含义
TFO_PAUSE	5	暂停
TFO_RESUME	6	恢复

2.2.28. 涨跌停控制EStopTradeForOwnHiLow

变量名	值	含义
STOPTRADE_NONE	0	无

变量名	值	含义
STOPTRADE_NO_BUY_SELL	1	涨停不卖跌停不买
STOPTRADE_SCHEDULED	2	原定时长执行
STOPTRADE_NOT_CHASE	3	涨停不买跌停不卖

2.2.29. 两融负债头寸来源EXTCompactBrushSource

变量名	值	含义
COMPACT_BRUSH_SOURCE_ALL	32	不限制
COMPACT_BRUSH_SOURCE_NORMAL	48	普通头寸
COMPACT_BRUSH_SOURCE_SPECIAL	49	专项头寸

2.2.30. 触价类型EOpTriggerType

变量名	值	含义
OTT_NONE	0	不使用触价
OTT_UP	1	向上触价
OTT_DOWN	2	向下触价

2.2.31. 市场状态EXTExchangeStatus

变量名	值	含义
EXCHANGE_STATUS_INVALID	0	无效状态
EXCHANGE_STATUS_IN_BEFORE_TRADING	48	开盘前
EXCHANGE_STATUS_NOTRADING	49	非交易
EXCHANGE_STATUS_IN_CONTINOUS	50	连续交易
EXCHANGE_STATUS_AUCTION_ORDERING	51	集合竞价报单
EXCHANGE_STATUS_AUCTION_BALANCE	52	集合竞价价格平衡

变量名	值	含义
EXCHANGE_STATUS_AUCTION_MATCH	53	集合竞价撮合
EXCHANGE_STATUS_IN_CLOSED	54	收盘
EXCHANGE_STATUS_ONLY_CANCEL	55	仅允许撤单
EXCHANGE_STATUS_DAZONG_ORDERING	56	大宗交易申报
EXCHANGE_STATUS_DAZONG_INTENTION_ORDERING	57	大宗交易意向申报
EXCHANGE_STATUS_DAZONG_CONFIRM_ORDERING	58	大宗交易成交申报
EXCHANGE_STATUS_DAZONG_CLOSE_PRICE_ORDERING	59	大宗交易盘后定价申报
EXCHANGE_STATUS_ONLY_GOLD_DELIVERY	60	交割申报
EXCHANGE_STATUS_ONLY_GOLD_MIDDLE	61	中立仓申报
EXCHANGE_STATUS_AFTER_HOURS_SALE	62	盘后协议买卖
EXCHANGE_STATUS_CLOSING_AUCTION_MATCH	63	收盘集合竞价
EXCHANGE_STATUS_CLOSE_PRICE_ORDERING	64	盘后定价申报

2.2.32. 是否使用大额单边保证金算法 EXtMaxMarginSideAlgorithmType

变量名	值	含义
XT_FTDC_MMSA_NO	48	不使用大额单边保证金算法
XT_FTDC_MMSA_YES	49	使用大额单边保证金算法

2.2.33. 普通算法订单类型EAlgoPriceType

变量名	值	含义
EALGO_PRT_MARKET	0	市价

变量名	值	含义
EALGO_PRT_FIX	1	限价

2.2.34. 银证转账方向ETransDirection

变量名	值	含义
TRANS_DIRECTION_BANK_TO_SECURITIES	49	银行转证券
TRANS_DIRECTION_SECURITIES_TO_BANK	50	证券转银行
TRANS_DIRECTION_QUICK_TO_CENTER	51	快速转集中
TRANS_DIRECTION_CENTER_TO_QUICK	52	集中转快速
TRANS_DIRECTION_QUERY	53	查询

2.2.35. 算法下单方式EOrderStrategyType

变量名	值	含义
E_ORDER_STRATEGY_TYPE_NORMAL	-1	普通
E_ORDER_STRATEGY_TYPE_APPROACH	0	近场交易，需要服务支持

2.2.36. 双中心状态 EDualStatus

变量名	值	含义
E_DUAL_STATUS_NONE	-1	无双中心
E_DUAL_STATUS_SH	0	仅有上海
E_DUAL_STATUS_SZ	1	仅有深圳
E_DUAL_STATUS_ALL	2	上海深圳双中心均有

2.2.37. 股份划拨信用划拨类别 ETransTypeCreditFlag

变量名	值	含义
TRANS_TRANSFER_SHARE	0	操作客户股份划拨，默认送0

变量名	值	含义
TRANS_TRANSFER_SPECIAL_POSITIONS	1	操作客户专向头寸调拨
TRANS_TRANSFER_CREDIT_SHARE	2	融资融券客户股份调拨

2.2.38. 资金股份划拨划拨方向 ESecuFundTransDirection

变量名	值	含义
SECUFUNDTRANS_TRANSFER_NORMAL_TO_FAST	0	从普通柜台拨入到极速柜台
SECUFUNDTRANS_TRANSFER_FAST_TO_NORMAL	1	从极速柜台拨出到普通柜台
SECUFUNDTRANS_TRANSFER_SH_TO_SZ	2	上海划到深圳(节点划拨)
SECUFUNDTRANS_TRANSFER_SZ_TO_SH	3	深圳划到上海(节点划拨)

2.2.39. 委托明细价格类型 EBrokerPriceType

枚举变量名	值	含义
BROKER_PRICE_ANY	49	市价
BROKER_PRICE_LIMIT	50	限价
BROKER_PRICE_BEST	51	最优价
BROKER_PRICE_PROP_ALLOTMENT	52	配股
BROKER_PRICE_PROP_REFER	53	转托
BROKER_PRICE_PROP_SUBSCRIBE	54	申购
BROKER_PRICE_PROP_BUYBACK	55	回购
BROKER_PRICE_PROP_PLACING	56	配售
BROKER_PRICE_PROP_DECIDE	57	指定
BROKER_PRICE_PROP_EQUITY	58	转股

枚举变量名	值	含义
BROKER_PRICE_PROP_SELLBACK	59	回售
BROKER_PRICE_PROP_DIVIDEND	60	股息
BROKER_PRICE_PROP_SHENZHEN_PLACING	68	深圳配售确认
BROKER_PRICE_PROP_CANCEL_PLACING	69	配售放弃
BROKER_PRICE_PROP_WDZY	70	无冻质押
BROKER_PRICE_PROP_DJZY	71	冻结质押
BROKER_PRICE_PROP_WDJY	72	无冻解押
BROKER_PRICE_PROP_JDJY	73	解冻解押
BROKER_PRICE_PROP_VOTE	75	投票
BROKER_PRICE_PROP_YYSGYS	76	要约收购预售
BROKER_PRICE_PROP_YSYYJC	77	预售要约解除
BROKER_PRICE_PROP_FUND_DEVIDEND	78	基金设红
BROKER_PRICE_PROP_FUND_ENTRUST	79	基金申赎
BROKER_PRICE_PROP_CROSS_MARKET	80	跨市转托
BROKER_PRICE_PROP_ETF	81	ETF

枚举变量名	值	含义
		申购
BROKER_PRICE_PROP_EXERCIS	83	权证行权
BROKER_PRICE_PROP_PEER_PRICE_FIRST	91	对手方最优价格
BROKER_PRICE_PROP_L5_FIRST_LIMITPX	92	最优五档即时成交剩余转限价
BROKER_PRICE_PROP_MIME_PRICE_FIRST	93	本方最优价格
BROKER_PRICE_PROP_INSTBUSI_RESTCANCEL	94	即时成交剩余撤销
BROKER_PRICE_PROP_L5_FIRST_CANCEL	95	最优五档即时成交剩余撤销
BROKER_PRICE_PROP_FULL_REAL_CANCEL	96	全额成交并撤单
BROKER_PRICE_PROP_FUND_CHAIHE	90	基金

枚举变量名	值	含义
		拆合
BROKER_PRICE_PROP_MARKET_BEST	131	市价_最优价
BROKER_PRICE_PROP_MARKET_CANCEL	132	市价_即成剩撤
BROKER_PRICE_PROP_MARKET_CANCEL_ALL	133	市价_全额成交或撤
BROKER_PRICE_PROP_MARKET_CANCEL_1	134	市价_最优1档即成剩撤
BROKER_PRICE_PROP_MARKET_CANCEL_5	135	市价_最优5档即成剩撤
BROKER_PRICE_PROP_MARKET_CONVERT_1	136	市价_最优1档即成剩转
BROKER_PRICE_PROP_MARKET_CONVERT_5	137	市价

枚举变量名	值	含义
		_最优5档即成剩转
BROKER_PRICE_PROP_STK_OPTION_ASK	138	股票期权-询价
BROKER_PRICE_PROP_STK_OPTION_FIX_CANCEL_ALL	139	股票期权-限价即时全部成交否则撤单
BROKER_PRICE_PROP_STK_OPTION_MARKET_CACEL_LEFT	140	股票期权-市价即时成交剩余撤单
BROKER_PRICE_PROP_STK_OPTION_MARKET_CANCEL_ALL	141	股票期权-市价即时全部成交否则撤

枚举变量名	值	含义
		单
BROKER_PRICE_PROP_STK_OPTION_MARKET_CONVERT_FIX	142	股票期权-市价剩余转限价
BROKER_PRICE_PROP_OPTION_COMB_EXERCISE	164	股票期权-组合行权
BROKER_PRICE_PROP_OPTION_BUILD_COMB_STRATEGY_CNSJC	165	股票期权-构建认购牛市价差策略
BROKER_PRICE_PROP_OPTION_BUILD_COMB_STRATEGY_PXSJC	166	股票期权-构建认沽熊市价差策略
BROKER_PRICE_PROP_OPTION_BUILD_COMB_STRATEGY_PNSJC	167	股票期权-构建认沽牛

枚举变量名	值	含义
		市价差策略
BROKER_PRICE_PROP_OPTION_BUILD_COMB_STRATEGY_CXSJC	168	股票期权-构建认购熊市价差策略
BROKER_PRICE_PROP_OPTION_BUILD_COMB_STRATEGY_KS	169	股票期权-构建跨式空头策略
BROKER_PRICE_PROP_OPTION_BUILD_COMB_STRATEGY_KKS	170	股票期权-构建宽跨式空头策略
BROKER_PRICE_PROP_OPTION_BUILD_COMB_STRATEGY_ZBD	171	股票期权-普通转备兑
BROKER_PRICE_PROP_OPTION_BUILD_COMB_STRATEGY_ZXJ	172	股票期权-备

枚举变量名	值	含义
		兑转普通
BROKER_PRICE_PROP_OPTION_RELEASE_COMB_STRATEGY_CNSJC	173	股票期权-解除认购牛市价差策略
BROKER_PRICE_PROP_OPTION_RELEASE_COMB_STRATEGY_PXSJC	174	股票期权-解除认沽熊市价差策略
BROKER_PRICE_PROP_OPTION_RELEASE_COMB_STRATEGY_PNSJC	175	股票期权-解除认沽牛市价差策略
BROKER_PRICE_PROP_OPTION_RELEASE_COMB_STRATEGY_CXSJC	176	股票期权-解除认购熊市价差策略

枚举变量名	值	含义
BROKER_PRICE_PROP_OPTION_RELEASE_COMB_STRATEGY_KS	177	股票期权-解除跨式空头策略
BROKER_PRICE_PROP_OPTION_RELEASE_COMB_STRATEGY_KKS	178	股票期权-解除宽跨式空头策略

2.2.40. 开平标志 EOffsetFlagType

枚举变量名	值	含义
EOFF_THOST_FTDC_OF_INVALID	-1	未知
EOFF_THOST_FTDC_OF_Open	48	开仓
EOFF_THOST_FTDC_OF_Close	49	平仓
EOFF_THOST_FTDC_OF_ForceClose	50	强平
EOFF_THOST_FTDC_OF_CloseToday	51	平今
EOFF_THOST_FTDC_OF_CloseYesterday	52	平昨
EOFF_THOST_FTDC_OF_ForceOff	53	强减
EOFF_THOST_FTDC_OF_LocalForceClose	54	本地强平
EOFF_THOST_FTDC_OF_PLEDGE_IN	81	质押入库
EOFF_THOST_FTDC_OF_PLEDGE_OUT	66	质押出库
EOFF_THOST_FTDC_OF_ALLOTMENT	67	股票配股

枚举变量名	值	含义
EOFF_THOST_FTDC_OF_OPTION_EXERCISE	68	行权
EOFF_THOST_FTDC_OF_OPTION_LOCK	69	锁定
EOFF_THOST_FTDC_OF_OPTION_UNLOCK	70	解锁

2.2.41. 投保标志 EHedgeFlagType

枚举变量名	值	含义
HEDGE_FLAG_SPECULATION	49	投机
HEDGE_FLAG_ARBITRAGE	50	套利
HEDGE_FLAG_HEDGE	51	套保

2.3. XtTraderApi数据结构说明

2.3.1. 股票、期货、股票期权的资产结构 CAccountDetail

属性	类型	注释
m_strAccountID	str	资金账号
m_nAccountType	int	账号类型
m_strStatus	str	状态
m_strTradingDate	str	交易日
m_dFrozenMargin	float	外源性 股票的保证金就是冻结资金 股票不需要
m_dFrozenCash	float	内外源冻结保证金和手续费四个的和
m_dFrozenCommission	float	外源性冻结资金源
m_dRisk	float	风险度
m_dNav	float	单位净值
m_dPreBalance	float	期初权益

属性	类型	注释
m_dBalance	float	动态权益, 总资产
m_dAvailable	float	可用资金
m_dCommission	float	已用手续费
m_dPositionProfit	float	持仓盈亏
m_dCloseProfit	float	平仓盈亏
m_dCashIn	float	出入金净值
m_dCurrMargin	float	当前占用保证金
m_dInstrumentValue	float	合约价值
m_dDeposit	float	入金
m_dWithdraw	float	出金
m_dCredit	float	信用额度
m_dMortgage	float	质押
m_dStockValue	float	股票总市值，期货没有
m_dLoanValue	float	债券总市值，期货没有
m_dFundValue	float	基金总市值，包括ETF和封闭式基金，期货没有
m_dRepurchaseValue	float	回购总市值，所有回购，期货没有
m_dLongValue	float	多单总市值，现货没有
m_dShortValue	float	空单总市值，现货没有
m_dNetValue	float	m_dNetValue
m_dAssureAsset	float	净资产
m_dTotalDebit	float	总负债
m_dPremiumNetExpense	float	权利金净支出 用于股票期权
m_dEnableMargin	float	可用保证金 用于股票期权
m_dFetchBalance	float	可取金额

属性	类型	注释
m_eDualStatus	float	双中心状态
m_dAvailableSH	float	双中心上海可用
m_dAvailableSZ	float	双中心深圳可用
m_strAccountKey	str	账号key
m_nProductId	int	账号
m_dUsedMargin	float	占用保证金 用于股票期权
m_dRoyalty	float	权利金支出 用于股票期权
m_strProductName	str	迅投产品名称
m_dDaysProfit	float	当日盈亏

2.3.2. 信用综合资产结构 CCreditDetail

属性	类型	注释
m_dPerAssurescaleValue	float	个人维持担保比例
m_dEnableBailBalance	float	可用保证金
m_dUsedBailBalance	float	已用保证金
m_dAssureEnbuyBalance	float	可买担保品资金
m_dFinEnbuyBalance	float	可买标的券资金
m_dSloEnrepaidBalance	float	可还券资金
m_dFinEnrepaidBalance	float	可还款资金
m_dFinMaxQuota	float	融资授信额度
m_dFinEnableQuota	float	融资可用额度
m_dFinUsedQuota	float	融资已用额度
m_dFinUsedBail	float	融资已用保证金额
m_dFinCompactBalance	float	融资合约金额

属性	类型	注释
m_dFinCompactFare	float	融资合约费用
m_dFinCompactInterest	float	融资合约利息
m_dFinMarketValue	float	融资市值
m_dFinIncome	float	融资合约盈亏
m_dSloMaxQuota	float	融券授信额度
m_dSloEnableQuota	float	融券可用额度
m_dSloUsedQuota	float	融券已用额度
m_dSloUsedBail	float	融券已用保证金额
m_dSloCompactBalance	float	融券合约金额
m_dSloCompactFare	float	融券合约费用
m_dSloCompactInterest	float	融券合约利息
m_dSloMarketValue	float	融券市值
m_dSloIncome	float	融券合约盈亏
m_dOtherFare	float	其它费用
m_dUnderlyMarketValue	float	标的证券市值
m_dFinEnableBalance	float	可融资金额
m_dDiffEnableBailBalance	float	可用保证金调整值
m_dBuySecuRepayFrozenMargin	float	买券还券冻结资金
m_dBuySecuRepayFrozenCommission	float	买券还券冻结手续费

2.3.3. 指令状态 COrderInfo

属性	类型	注释
m_strAccountID	str	资金账号
m_eBrokerType	int	账号类型

属性	类型	注释
m_nOrderID	int	指令ID
m_startTime	int	下达时间
m_endTime	int	结束时间
m_eStatus	int	指令状态
m_dTradedVolume	float	成交量
m_strMsg	str	指令执行信息
m_canceler	str	撤销者
m_eBrokerType	int	账号类型
m_strRemark	str	投资备注
m_strMarket	str	市场 仅三方算法指令有值
m_strInstrument	str	合约 仅三方算法指令有值
m_strOrderType	str	算法名称 仅三方算法指令有值
m_dPrice	float	基准价 仅三方算法指令有值
m_nVolume	int	下单总量 仅三方算法指令有值
m_nValidTimeStart	int	有效开始时间 仅三方算法指令有值
m_nValidTimeEnd	int	有效结束时间 仅三方算法指令有值
m_dMaxPartRate	float	量比比例 仅三方算法指令有值
m_dMinAmountPerOrder	float	委托最小金额 仅三方算法指令有值

属性	类型	注释
m_eOperationType	EOperationType	下单操作：买入卖出 仅三方算法指令有值
m_nStopTradeForOwnHiLow	EStopTradeForOwnHiLow	涨跌停控制 仅三方算法指令有值
m_strAccountKey	str	账号key

2.3.4. 委托明细 COrderDetail

属性	类型	注释
m_strAccountID	str	资金账号
m_nAccountType	int	账号类型，参见数据字典
m_strExchangeID	str	交易所代码，参见数据字典
m_strProductID	str	合约品种
m_strInstrumentID	str	合约代码
m_dLimitPrice	float	限价单的委托价格
m_strOrderSysID	str	委托号
m_nTradedVolume	int	已成交量
m_nTotalVolume	int	当前总委托量
m_dFrozenMargin	float	冻结保证金
m_dFrozenCommission	float	冻结手续费
m_dAveragePrice	float	成交均价
m_dTradeAmount	float	成交额 期货=均价量合约乘数
m_nErrorID	int	交易所废单情况下的错误编号
m_strErrorMsg	str	废单原因
m_strInsertDate	str	日期
m_strInsertTime	str	时间

属性	类型	注释
m_nOrderID	int	指令ID
m_nOrderPriceType	int	类型，例如市价单 限价单
m_nDirection	EEntrustBS	期货多空,该字段与m_eOffsetFlag一起判断期货的报单类型，股票无用。
m_eOffsetFlag	int	期货开平—股票单，用该字段判断方向
m_eHedgeFlag	EHedgeFlagType	投机 套利 套保
m_eOrderSubmitStatus	int	提交状态，股票里面不需要报单状态
m_eOrderStatus	int	委托状态
m_eEntrustType	int	委托类别
m_eCoveredFlag	int	备兑标记 '0' - 非备兑，'1' - 备兑
m_strRemark	str	投资备注
m_strCancelInfo	str	废单信息
m_strInstrumentName	str	合约名称
m_strAccountKey	str	账号key
m_strStrategyID	str	收益互换策略ID
m_strSecuAccount	str	股东号

2.3.5. 成交明细 CDealDetail

属性	类型	注释
m_strAccountID	str	资金账号
m_nAccountType	int	账号类型, 参见数据字典
m_strExchangeID	str	交易所代码
m_strProductID	str	合约品种
m_strInstrumentID	str	合约代码

属性	类型	注释
m_strTradeID	str	成交编号
m_strOrderSysID	str	委托号
m_dAveragePrice	float	成交均价
m_nVolume	int	成交量 (期货单位手 股票做到股)
m_strTradeDate	str	成交日期
m_strTradeTime	str	成交时间
m_dComssion	float	手续费
m_dAmount	float	成交额 (期货=均价量合约乘数)
m_nOrderID	int	指令ID
m_nDirection	EEntrustBS	期货多空 该字段与m_eOffsetFlag一起判断期货的报单类型。股票无用
m_nOffsetFlag	int	期货开平 股票买卖
m_nHedgeFlag	int	投机 套利 套保
m_nOrderPriceType	int	类型，例如市价单 限价单
m_eEntrustType	int	委托类别
m_eCoveredFlag	int	备兑标记 '0' - 非备兑，'1' - 备兑
m_strRemark	str	投资备注
m_strInstrumentName	str	合约名称
m_strAccountKey	str	账号key
m_strStrategyID	str	收益互换策略ID
m_strSecuAccount	str	股东号
m_nProductId	int	迅投产品ID
m_strProductName	str	迅投产品名称

2.3.6. 持仓明细 CPositionDetail

属性	类型	注释
m_strAccountID	str	资金账号
m_strExchangeID	str	市场代码
m_strProductID	str	合约品种
m_strInstrumentID	str	合约代码
m_strInstrumentName	str	合约名称
m_strOpenDate	str	开仓日期
m_strTradeID	str	最初开仓位的成交编号
m_nVolume	int	持仓量 当前拥股
m_dOpenPrice	float	开仓价
m_strTradingDay	str	交易日
m_dMargin	float	使用的保证金 (历史的直接用ctp的，新的自己用成本价*存量*系数算 股票不需要)
m_dOpenCost	float	开仓成本 (等于股票的成本价*第一次建仓的量，后续减持不影响，不算手续费 股票不需要)
m_dSettlementPrice	str	结算价 对于股票的当前价
m_nCloseVolume	int	平仓量 等于股票已经卖掉的 股票不需要
m_dCloseAmount	float	平仓额 等于股票每次卖出的量*卖出价*合约乘数（股票为1）的累加 股票不需要
m_dFloatProfit	float	浮动盈亏 浮动盈亏 当前量*（当前价-开仓价）*合约乘数（股票为1）
m_dCloseProfit	float	平仓盈亏 平仓额 - 开仓价*平仓量*合约乘数（股票为1） 股票不需要

属性	类型	注释
m_dMarketValue	float	市值
m_dPositionCost	float	持仓成本 股票不需要
m_dPositionProfit	float	持仓盈亏 股票不需要
m_dLastSettlementPrice	float	最新结算价 股票不需要
m_dInstrumentValue	float	合约价值 股票不需要
m_bIsToday	int	是否今仓
m_nOrderID	int	指令ID
m_nFrozenVolume	int	冻结数量 期货不用这个字段
m_nCanUseVolume	int	可用数量 期货不用这个字段
m_nOnRoadVolume	int	在途数量 期货不用这个字段
m_nYesterdayVolume	int	昨日拥股 期货不用这个字段
m_dLastPrice	float	结算价 对于股票的当前价
m_nHedgeFlag	EHedgeFlagType	投机 套利 套保
m_nDirection	EEntrustBS	期货多空，该字段与m_eOffsetFlag一起判断期货的报单类型
m_nCoveredAmount	int	备兑数量
m_nAccountType	int	账号类型
m_dProfitRate	float	持仓盈亏比例
m_strAccountKey	str	账号key
m_strStrategyID	str	收益互换策略ID
m_strSecuAccount	str	股东号

2.3.7. 持仓统计 CPositionStatics

属性	类型	注释
m_strAccountID	str	资金账号
m_strExchangeID	str	市场代码
m_strProductID	str	合约品种
m_strInstrumentID	str	合约代码
m_strInstrumentName	str	合约名称
m_bIsToday	int	是否今仓
m_nPosition	int	持仓量
m_dOpenCost	float	开仓成本
m_dPositionCost	float	持仓成本
m_dAveragePrice	float	算法待找
m_dPositionProfit	float	持仓盈亏
m_dFloatProfit	float	浮动盈亏
m_dOpenPrice	float	开仓均价
m_nCanCloseVolume	int	可平量
m_dUsedMargin	float	已使用保证金
m_dUsedCommission	float	已使用的手续费
m_dFrozenMargin	float	冻结保证金
m_dFrozenCommission	float	冻结手续费
m_dInstrumentValue	float	合约价值
m_nOpenTimes	int	开仓次数
m_nOpenVolume	int	总开仓量
m_nCancelTimes	int	撤单次数
m_nFrozenVolume	int	冻结数量

属性	类型	注释
m_nCanUseVolume	int	可用数量
m_nOnRoadVolume	int	在途数量
m_nYesterdayVolume	int	昨日拥股
m_dSettlementPrice	float	前收盘价
m_dProfitRate	float	持仓盈亏比例
m_nDirection	EEntrustBS	期货多空，该字段与m_eOffsetFlag一起判断期货的报单类型
m_nHedgeFlag	int	投机 套利 套保
m_nCoveredAmount	int	备兑数量
m_nAccountType	int	账号类型
m_dLastPrice	float	最新价
m_strStrategyID	str	收益互换策略ID
m_strAccountKey	str	账号key
m_strSecuAccount	str	股东号

2.3.8. 普通委托 COrdinaryOrder

属性	类型	注释
m_strAccountID	str	资金账号
m_dPrice	float	指定价
m_dSuperPriceRate	float	单笔超价百分比，小数
m_nVolume	int	委托量, 直接还券的数量
m_strMarket	str	市场代码
m_strInstrument	str	合约代码
m_ePriceType	EPriceType	报价方式

属性	类型	注释
m_eOperationType	EOperationType	下单类型
m_eHedgeFlag	EHedgeFlagType	投机、套利、套保
m_dOccurBalance	float	直接还款的金额 //仅直接还款用
m_eTimeCondition	int	期货条件单时间条件
m_eVolumeCondition	int	期货条件单数量条件
m_strSecondInstrument	str	期权组合委托合约
m_dSecondPrice	float	期权组合委托价
m_eFirstSideFlag	int	第一腿合约持仓类型
m_eSecondSideFlag	int	第二腿合约持仓类型
m_strCombID	str	组合持仓编号，用于解除组合策略
m_strRemark	str	投资备注
m_eTriggerType	EOpTriggerType	触价类型
m_dTriggerPrice	float	触价价格
m_dSuperPrice	float	单笔超价,和m_dSuperPriceRate只用设置一个，优先使用m_dSuperPriceRate
m_nPortfolioID	int	投组类型编号
m_strPortfolioStrategyName	int	投组策略名称，优先使用m_nPortfolioID
m_eAbroadDurationType	EAbroadDurationType	外盘期货报价类型
m_strStrategyID	str	收益互换策略ID
m_strSecuAccount	str	股东号
m_eAdjustOrderNum	EAdjustOrderNum	下单根据可用调整可下单

属性	类型	注释
		量，仅api直连pb时有效

2.3.9. 算法委托 CAlgorithmOrder

属性	类型	注释
m_strAccountID	str	资金账号
m_strMarket	str	市场代码
m_strInstrument	str	合约代码
m_dPrice	float	指定价格
m_dSuperPriceRate	float	单笔超价百分比，小数
m_nSuperPriceStart	int	超价起始笔数
m_nVolume	int	委托量
m_dSingleVolumeRate	float	单比下单比率
m_nSingleNumMin	int	单比下单量最小值
m_nSingleNumMax	int	单比下单量最大值
m_nLastVolumeMin	int	尾单最小量
m_dPlaceOrderInterval	float	下单间隔
m_dWithdrawOrderInterval	float	撤单间隔
m_nMaxOrderCount	int	最大下单次数
m_nValidTimeStart	int	有效开始时间
m_nValidTimeEnd	int	有效结束时间
m_eOperationType	EOperationType	下单类型
m_eSingleVolumeType	int	单笔基准量
m_ePriceType	EPriceType	报价方式
m_eHedgeFlag	EHedgeFlagType	投机、套利、套保

属性	类型	注释
m_eOverFreqOrderMode	int	委托频率过快时的处理方式
m_eTimeCondition	int	期货条件单时间条件
m_eVolumeCondition	int	期货条件单数量条件
m_strRemark	str	投资备注
m_eTriggerType	EOpTriggerType	触价类型
m_dTriggerPrice	float	触价价格
m_dSuperPrice	float	单笔超价,和m_dSuperPriceRate只用设置一个，优先使用m_dSuperPriceRate
m_eAlgoPriceType	EAlgoPriceType	订单类型
m_nExtraLimitType	int	扩展限价类型
m_dExtraLimitValue	float	扩展限价
m_strStrategyID	str	收益互换策略ID

2.3.10. 智能算法委托 CIntelligentAlgorithmOrder

属性	类型	注释
m_strAccountID	str	资金账号
m_strMarket	str	市场代码
m_strInstrument	str	合约代码
m_strOrderType	str	智能算法名称，可以参考数据字典
m_dPrice	float	基准价
m_nVolume	int	委托量
m_nValidTimeStart	int	有效开始时间

属性	类型	注释
m_nValidTimeEnd	int	有效结束时间
m_dMaxPartRate	float	量比比例
m_dMinAmountPerOrder	float	委托最小金额
m_eOperationType	EOperationType	下单类型
m_ePriceType	EPriceType	报价方式
m_strRemark	str	投资备注
m_dOrderRateInOpenAcution	float	开盘集合竞价参与比例(取值0-1) 仅开盘+算法有用
m_dPriceOffsetBpsForAuction	int	开盘集合竞价价格偏移量(取值0-10000) 仅开盘+算法有用
m_nStopTradeForOwnHiLow	EStopTradeForOwnHiLow	涨跌停控制
m_bOnlySellAmountUsed	bool	仅用卖出金额 0,1 换仓特有
m_dBuySellAmountDeltaPct	float	买卖偏差上限 0.03-1 换仓特有
m_nMaxTradeDurationAfterET	int	收盘后是否继续执行， 0不继续，非0继续
m_eOrderStrategyType	EOrderStrategyType	算法下单方式
m_strStrategyID	str	收益互换策略ID

2.3.11. 主动算法委托 CExternAlgorithmOrder

属性	类型	注释
m_strAccountID	str	资金账号

属性	类型	注释
m_strMarket	str	市场代码
m_strInstrument	str	合约代码
m_strOrderType	str	主动算法名称，FTAIWAP，ALGOINTERFACE，ZEUS....
m_dPrice	float	基准价
m_nVolume	int	委托量
m_nValidTimeStart	int	有效开始时间
m_nValidTimeEnd	int	有效结束时间
m_dMaxPartRate	float	量比比例
m_dMinAmountPerOrder	float	委托最小金额
m_eOperationType	EOperationType	下单类型
m_strRemark	str	投资备注
m_nStopTradeForOwnHiLow	EStopTradeForOwnHiLow	涨跌停控制
m_eOrderStrategyType	EOrderStrategyType	算法下单方式
m_nMaxTradeDurationAfterET	float	收盘后是否继续执行， 0不继续，非0继续
m_strStrategyID	int	收益互换策略ID

2.3.12. 错误消息 XtError

属性	类型	注释
m_nErrorID	int	错误编号
m_strErrorMsg	str	错误原因

2.3.13. 委托错误信息 COrderError(委托请求被迅投风控或者柜台打回，会推送该消息)

属性	类型	注释
m_strAccountID	str	资金账号
m_nErrorID	int	错误编号
m_strErrorMsg	str	错误原因
m_nOrderID	str	指令编号
m_nRequestID	int	请求编号
m_strRemark	str	投资备注
m_strAccountKey	str	账号key

2.3.14. 撤销委托错误信息 CCancelError (撤销委托请求被迅投风控或者柜台打回，会推送该消息)

属性	类型	注释
m_strAccountID	str	资金账号
m_nErrorID	int	错误编号
m_strErrorMsg	str	错误原因
m_nOrderID	str	指令编号
m_nRequestID	int	请求编号
m_strRemark	str	投资备注
m_strAccountKey	str	账号key

2.3.15. 两融标的信息 CStkSubjects

属性	类型	注释
m_strAccountID	str	账号
m_strExchangeID	str	市场

属性	类型	注释
m_strInstrumentID	str	合约
m_dSloRatio	float	融资融券保证金比例
m_eSloStatus	EXTSubjectsStatus	融券状态
m_nEnableAmount	int	融券可融数量
m_dFinRatio	float	融资保证金比例
m_eFinStatus	EXTSubjectsStatus	融资状态
m_dAssureRatio	float	担保品折算比例
m_eAssureStatus	EXTSubjectsStatus	是否可做担保
m_dFinRate	float	融资日利率
m_dSloRate	float	融券日利率
m_dFinPenaltyRate	float	融资日罚息利率
m_dSloPenaltyRate	float	融券日罚息利率

2.3.16. 两融标的状态枚举类型 EXTSubjectsStatus

变量名	值	含义
SUBJECTS_STATUS_NORMAL	48	正常
SUBJECTS_STATUS_PAUSE	49	暂停
SUBJECTS_STATUS_NOT	50	非标的

2.3.17. 个股期权备兑持仓信息 CCoveredStockPosition

属性	类型	注释
m_strAccountID	str	账号
m_strExchangeID	str	交易类别
m_strExchangeName	str	市场名字

属性	类型	注释
m_strInstrumentID	str	合约代码
m_strInstrumentName	str	合约名称
m_nTotalAmount	int	总持仓量
m_nLockAmount	int	锁定量
m_nUnlockAmount	int	未锁定量
m_nCoveredAmount	int	备兑数量

2.3.18. 股票期权组合策略持仓信息 CStockOptionCombPositionDetail

属性	类型	注释
m_strAccountID	str	账号
m_strExchangeID	str	交易类别
m_strExchangeName	str	市场名字
m_strContractAccount	str	合约账号
m_strCombID	str	组合编号
m_strCombCode	str	组合策略编码
m_strCombCodeName	str	组合策略名称
m_nVolume	int	总持仓量
m_nFrozenVolume	int	冻结数量
m_nCanUseVolume	int	可用数量
m_strFirstCode	str	合约一
m_eFirstCodeType	EOptionType	合约一类型
m_strFirstCodeName	str	合约一名称
m_eFirstCodePosType	ESideFlag	合约一持仓类型

属性	类型	注释
m_nFirstCodeAmt	int	合约一数量
m_strSecondCode	str	合约二
m_eSecondCodeType	EOptionType	合约二类型
m_strSecondCodeName	str	合约二名称
m_eSecondCodePosType	ESideFlag	合约二持仓类型
m_nSecondCodeAmt	int	合约二数量
m_dCombBailBalance	float	占用保证金

2.3.19. 股票期权合约类型 EOptionType

枚举变量名	值	含义
OPTION_TYPE_CALL	48	认购合约
OPTION_TYPE_PUT	49	认沽合约

2.3.20. 期权持仓类型 ESideFlag

变量名	值	含义
SIDE_FLAG_RIGHT	'0'	权利
SIDE_FLAG_DUTY	'1'	义务
SIDE_FLAG_COVERED	'2'	备兑

2.3.21. 订阅行情请求 CSubscribData

属性	类型	注释
m_nPlatformID	int	平台ID
m_strExchangeID	str	市场代码
m_strInstrumentID	str	合约代码,当其值为allCode时，订阅整个市场

属性	类型	注释
m_eOfferStatus	EXTOfferStatus	报盘状态

2.3.22. 汇率信息 CReferenceRate

属性	类型	注释
m_strAccountID	str	账号
m_strExchangeID	str	市场代码
m_strExchangeName	str	市场名字
m_eMoneyType	EMoneyType	币种
m_bBidReferenceRate	float	买入参考汇率
m_bAskReferenceRate	float	卖出参考汇率
m_bBidSettlementRate	float	买入结算汇率
m_bAskSettlementRate	float	卖出结算汇率
m_dDayBuyRiseRate	float	日间买入参考汇率浮动比例
m_dNightBuyRiseRate	float	夜市买入参考汇率浮动比例
m_dDaySaleRiseRate	float	日间卖出参考汇率浮动比例
m_dNightSaleRiseRate	float	日间卖出参考汇率浮动比例
m_bMidReferenceRate	float	参考汇率中间价

2.3.23. 行情数据 CPriceData

属性	类型	注释
m_strTradingDay	str	交易日
m_strExchangeID	str	交易所代码
m_strInstrumentID	str	合约代码
m_strInstrumentName	str	合约名称

属性	类型	注释
m_strExchangeInstID	str	合约在交易所的代码
m_dLastPrice	float	最新价
m_dUpDown	float	涨跌
m_dUpDownRate	float	涨跌幅
m_dAveragePrice	float	当日均价
m_nVolume	int	数量
m_dTurnover	float	成交金额
m_dPreClosePrice	float	昨收盘
m_dPreSettlementPrice	float	上次结算价
m_dPreOpenInterest	float	昨持仓量
m_dOpenInterest	float	持仓量
m_dSettlementPrice	float	本次结算价
m_dOpenPrice	float	今开盘
m_dHighestPrice	float	最高价
m_dLowestPrice	float	最低价
m_dClosePrice	float	今收盘
m_dUpperLimitPrice	float	涨停板价
m_dLowerLimitPrice	float	跌停板价
m_dPreDelta	float	昨虚实度
m_dCurrDelta	float	今虚实度
m_strUpdateTime	str	最后修改时间
m_nUpdateMillisec	int	最后修改毫秒
m_dBidPrice1	float	申买价一
m_nBidVolume1	int	申买量一

属性	类型	注释
m_dAskPrice1	float	申卖价一
m_nAskVolume1	int	申卖量一
m_dBidPrice2	float	申买价二
m_nBidVolume2	int	申买量二
m_dAskPrice2	float	申卖价二
m_nAskVolume2	int	申卖量二
m_dBidPrice3	float	申买价三
m_nBidVolume3	int	申买量三
m_dAskPrice3	float	申卖价三
m_nAskVolume3	int	申卖量三
m_dBidPrice4	float	申买价四
m_nBidVolume4	int	申买量四
m_dAskPrice4	float	申卖价四
m_nAskVolume4	int	申卖量四
m_dBidPrice5	float	申买价五
m_nBidVolume5	int	申买量五
m_dAskPrice5	float	申卖价五
m_nAskVolume5	int	申卖量五
m_dPrePrice	float	前一次的价格
m_nStockStatus	int	股票状态

2.3.24. 股票（合约）信息 CInstrumentDetail

属性	类型	注释
m_strInstrumentID	str	合约代码

属性	类型	注释
m_strInstrumentName	str	合约名称
m_strProductID	str	品种代码
m_strExchangeID	str	交易所代码
m_strOptUndlCode	str	期权标的代码
m_strOptUndlName	str	期权标的证券名称
m_strEndDelivDate	str	最后日期
m_dPriceTick	float	价格波动
m_nOptUnit	int	期权合约单位
m_dMarginUnit	float	保证金单位
m_dUpStopPrice	float	涨停价
m_dDownStopPrice	float	跌停价
m_dSettlementPrice	float	前结算
m_dOptExercisePrice	float	期权行权价格
m_nVolumeMultiple	int	合约乘数
m_nSuspendedType	EXtSuspendedType	停牌状态
m_nCallOrPut	int	合约种类(个股期权：0认购，1认沽)

2.3.24. 组合单下单参数 CGroupOrder

- 注意 list 容量限制500

属性	类型	注释
m_orderParam	CAlgorithmOrder	下单配置
m_strMarket	List[str]	市场列表
m_strInstrument	List[str]	证券代码

属性	类型	注释
m_nVolume	List[int]	每只股票的下单量
m_eOperationType	List[EOperationType]	每只股票的下单类型
m_nOrderNum	int	股票只数
m_strRemark	str	投资备注

2.3.24. 算法组合单下单参数 CAlgGroupOrder

- 注意 list 容量限制500

属性	类型	注释
m_orderParam	CIntelligentAlgorithmOrder	下单配置
m_strMarket	List[str]	市场列表
m_strInstrument	List[str]	证券代码
m_nVolume	List[int]	每只股票的下单量
m_eOperationType	List[EOperationType]	每只股票的下单类型
m_nOrderNum	int	股票只数
m_strRemark	str	投资备注

2.3.25. 外部算法组合单下单参数 CExternAlgGroupOrder

- 注意 list 容量限制500

属性	类型	注释
m_orderParam	CIntelligentAlgorithmOrder	下单配置
m_strMarket	List[str]	市场列表
m_strInstrument	List[str]	证券代码
m_nVolume	List[int]	每只股票的下单量
m_eOperationType	List[EOperationType]	每只股票的下单类型

属性	类型	注释
m_nOrderNum	int	股票只数
m_strRemark	str	投资备注

2.3.26. 普通组合单下单参数 COrdinaryGroupOrder

属性	类型	注释
m_strAccountID	str	资金账户ID，如果为子账户，则为子账户ID
m_dSuperPriceRate	float	单笔超价百分比
m_ePriceType	EPriceType	报价方式： 指定价，最新价 对手价……
m_eHedgeFlag	EHedgeFlagType	套利标志
m_eOverFreqOrderMode	EXtOverFreqOrderMode	委托频率过快时的处理方式
m_eTimeCondition	ETimeCondition	期货条件单时间条件
m_eVolumeCondition	EVolumeCondition	期货条件单数量条件
m_strMarket	list[str]	市场列表
m_strInstrument	list[str]	证券代码
m_nVolume	list[int]	每只股票的下单量
m_dPrice	list[float]	每只股票的下单价格
m_eOperationType	list[EOperationType]	每只股票的下单类型
m_nOrderNum	int	股票只数
m_strRemark	str	投资备注
m_dSuperPrice	float	单笔超价,和m_dSuperPriceRate只用设置一个，优先使用m_dSuperPriceRate

属性	类型	注释
m_strStrategyID	int	收益互换策略ID

2.3.26. 账号主键 CAccountKey

属性	类型	注释
m_nPlatformID	int	经纪公司编号
m_eBrokerType	EXTBrokerType	经纪公司类别
m_nAccountType	int	账号类型编号
m_strAccountID	str	资金账号
m_strSubAccount	str	账号类型
m_strAccountKey	str	资金账号对应唯一主键

2.3.27. 普通柜台资金信息 CStockComFund

属性	类型	注释
m_strAccountID	str	资金账户ID，如果为子账户，则为子账户ID
m_strAccountKey	str	账号key
m_dAvailable	float	可用资金

2.3.28. 普通柜台持仓信息 CStockComPosition

属性	类型	注释
m_strAccountID	str	资金账户ID，如果为子账户，则为子账户ID
m_strAccountKey	str	账号key
m_strExchangeID	str	市场代码
m_strInstrumentID	str	合约代码
m_strInstrumentName	str	合约名称

属性	类型	注释
m_nVolume	int	持仓量
m_nCanUseVolume	int	可用数量
m_dLastPrice	float	最新价
m_dCostPrice	float	成本价
m_dIncome	float	盈亏
m_dIncomeRate	float	盈亏比例
m_strHandFlag	str	股手标记
m_strStockAccount	str	证券账号
m_dInstrumentValue	float	市值
m_dCostBalance	float	成本总额
m_nOnRoadVolume	int	在途量
m_nPREnableVolume	int	申赎可用量
m_bFast	bool	是否是从极速接口查到的

2.4. XtTraderApi API说明

2.4.1. 普通单委托接口

```
order(ordinary_order, requestID, accountKey)
```

- 释义对股票、信用、期货等品种进行单笔普通委托下单操作
- 参数

参数	参数释义	参数类型
ordinary_order	普通单的下单结构，参见数据字典	COrdinaryOrder
requestID	请求ID，每个请求的ID自增	int
accountKey	唯一标识一个账号的key	str

- 示例股票资金账号6000000083对浦发银行买入100股, 使用限价价格12.3元, 委托备注为'alpha'

```
orderInfo = COrdinaryOrder()  # 初始化数据，普通单结构体
# 资金账号，必填参数。不填会被api打回，并且通过onOrder反馈失败
orderInfo.m_strAccountID = '6000000083'
orderInfo.m_dPrice = 12.3  # 报单价格，默认为double最大值
orderInfo.m_dSuperPriceRate = 0  # 单笔超价百分比，选填字段。默认为0
# 报单委托量，必填字段。默认int最大值，填0或不填会被api打回
orderInfo.m_nVolume = 100
# 报单市场。必填字段。股票市场有"SH"/"SZ"，如果填空或填错都会被api直接打回
orderInfo.m_strMarket = "SH"
orderInfo.m_strInstrument = "600000"  # 报单合约代码，必填字段
# 枚举类型，报单价格类型，必填字段
orderInfo.m_ePriceType = EPriceType.PRTP_FIX
# 报单委托类型。必填字段
orderInfo.m_eOperationType = EOperationType.OPT_BUY
orderInfo.m_strRemark = "alpha"
self.m_nRequestId += 1
self.m_client.order(orderInfo, self.m_nRequestId, accountKey)
```

2.4.2. 下达指令后的回调

```
onOrder(self, requestID, order_id, error)
```

- 释义调用order下达指令后的回调接口
- 参数

参数	参数释义	参数类型
requestID	请求ID，每个请求的ID自增	int
order_id	指令编号	int
error	错误消息	XtError

- 示例

```
#下达指令后的回调
def onOrder(self, requestID, order_id, error):
    if error.isSuccess():
        print u'[onOrder] isSuccess :True',u', orderId:',order_id,\
        u', nRequestId: ',requestID, u', errorMsg: ',error.errorMsg()
        #self.cancel_cmd(orderId)
    else:
        print u'[onOrder] isSuccess :false',u'orderId:',order_id,\
        u'nRequestId: ',requestID, u'errorMsg: ',error.errorMsg()
```

2.4.3. 指令状态主推接口

```
onRtnOrder(self, data)
```

- 释义调用order下达指令后，指令状态变化的主推接口
- 参数

参数	参数释义	参数类型
data	指令状态	COrderInfo

- 示例

```
# 获取主推的指令状态，data对应COrderInfo结构
def onRtnOrder(self, data):
    if data.m_eStatus == EOrderCommandStatus.OCS_CHECKING:
        orderStatus = u'风控检查中'
    elif data.m_eStatus == EOrderCommandStatus.OCS_RUNNING:
        orderStatus = u'运行中'
    elif data.m_eStatus == EOrderCommandStatus.OCS_APPROVING:
        orderStatus = u'审批中'
    elif data.m_eStatus == EOrderCommandStatus.OCS_REJECTED:
        orderStatus = u'已驳回'
    elif data.m_eStatus == EOrderCommandStatus.OCS_RUNNING:
        orderStatus = u'运行中'
    elif data.m_eStatus == EOrderCommandStatus.OCS_CANCELING:
        orderStatus = u'撤销中'
    elif data.m_eStatus == EOrderCommandStatus.OCS_FINISHED:
        orderStatus = u'已完成'
    elif data.m_eStatus == EOrderCommandStatus.OCS_STOPPED:
        orderStatus = u'已撤销'
    else:
        orderStatus = u'默认状态:', data.m_eStatus
    print u'[onRtnOrder] 下单ID: ', data.m_nOrderID, u'm_strRemark:',\
    data.m_strRemark, u'指令状态:', orderStatus, u'成交量:', data.\
    m_dTradedVolume, u'撤销者:', data.m_canceler, u'指令执行信息:',\
    data.m_strMsg
```

2.4.4. 委托主推接口

```
onRtnOrderDetail(self, data)
```

- 释义调用order下达指令后，委托数据的实时推送接口
- 参数

参数	参数释义	参数类型
data	委托明细	COrderInfo

- 示例

```
def onRtnOrderDetail(self, data):
    if data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_UNREPORTED:
        entrust_status = u'未报'
    elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_WAIT_REPORTING:
        entrust_status = u'待报'
    elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_REPORTED:
        entrust_status = u'已报'
    elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_REPORTED_CANCEL:
        entrust_status = u'已报待撤'
    elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_PARTSUCC_CANCEL:
        entrust_status = u'部成待撤'
    elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_PART_CANCEL:
        entrust_status = u'部撤'
    elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_CANCELED:
        entrust_status = u'已撤'
    elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_PART_SUCC:
        entrust_status = u'部成'
    elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_SUCCEEDED:
        entrust_status = u'已成'
    elif data.m_eOrderStatus == EEntrustStatus.ENTRUST_STATUS_JUNK:
        entrust_status = u'废单'
    else:
        entrust_status = u'默认状态->', data.m_eOrderStatus
    print u'[onRtnOrderDetail] 指令编号:', data.m_nOrderID,
    u', 投资备注:', data.m_strRemark, u', 合同编号:', data.m_strOrderSysID, \
        u', 委托状态:', entrust_status, u', 已成量:',data.m_nTradedVolume,\
        u', 成交均价:', data.m_dAveragePrice, u', 代码:', \
        data.m_strInstrumentID, u', ErrorID:', data.m_nErrorID,\
        u', ErrorMsg:', data.m_strErrorMsg
```

2.4.5. 成交主推接口

```
onRtnDealDetail(self, data)
```

- 释义调用order下达指令后，成交数据的实时推送接口
- 参数

参数	参数释义	参数类型
data	成交明细	CDealDetail

- 示例

```
# 获得主推的成交明细，data对应CDealDetail
def onRtnDealDetail(self, data):
    print u'[onRtnDealDetail] 指令编号:', data.m_nOrderID,  u', 投资备注:',\
    data.m_strRemark, data.m_nOrderID, u', 成交量:', data.m_nVolume, \
    u', 成交均价:', data.m_dAveragePrice
```

2.4.6. 撤指令

```
cancel(order_id, requestID)
```

- 释义根据指令编号order_id对指令进行撤单操作，order_id对应onOrder里的orderID以及onRtnOrder里的data->m_nOrderID
- 参数

参数名称	参数释义	参数类型
order_id	指令编号	int
requestID	请求ID，每个请求的ID自增	int

- 示例对指令编号为order_id的指令进行撤销

```
#撤销指令
def cancel_cmd(self, order_id):
    print u'[cancel_cmd] 撤指令, 指令编号:', order_id
    self.m_nRequestId += 1
    self.m_client.cancel(order_id,self.m_nRequestId)
```

2.4.7. 撤销指令回调

```
onCancel(self, requestID, error)
```

- 释义调用cancel对指令进行撤销后的回调接口
- 参数

参数名称	参数释义	参数类型
requestID	请求ID，每个请求的ID自增	int
error	错误消息	XtError

- 示例

```
#撤销指令回调
def onCancel(self, requestID, error):
    if error.isSuccess():
        print u'[onCancel] success'
    else:
        print u'[onCancel] failed, error_msg: ', error.errorMsg()
```

2.4.8. 撤委托

```
cancelOrder(accountID, order_sysid, market, code, requestID,  accountKey)
```

- 释义根据券商柜台返回的合同编号对委托进行撤单操作
- 参数

参数名	参数释义	参数类型
accountID	证券账号	str
order_sysid	合同编号	str
market	市场代码	str
code	合约代码	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str

- 示例对股票资金账号account_id合同编号为order_sysid的委托进行撤单

```
#撤销委托
def cancel_order(self, accountID, order_sysid, market, code, accountKey):
    print u'[cancel_order] 撤委托, 合同编号:', accountID, ', ',
    order_sysid,  ', ', market,  ', ', code,  ', ', accountKey,
    ', ', order_sysid
    self.m_nRequestId += 1
    self.m_client.cancelOrder(accountID, order_sysid, market, code,
    self.m_nRequestId, accountKey)
```

2.4.9. 撤销委托回调

```
onCancelOrder(self, requestID, error)
```

- 释义调用cancelOrder对委托进行撤销后的回调接口
- 参数

参数	参数释义	参数类型
requestID	请求ID，每个请求的ID自增	int
error	错误消息	XtError

- 示例

```
#撤销委托回调
def onCancelOrder(self, requestID, error):
    if error.isSuccess():
        print u'[onCancelOrder] success'
    else:
        print u'[onCancelOrder] failed, error_msg: ', error.errorMsg()
```

2.4.10. 主推委托撤销错误接口

```
onRtnCancelError(self, data)
```

- 释义主推的撤单错误信息，data对应CCancelError结构
- 参数

参数	参数释义	参数类型
data	撤单错误对象	CCancelError

- 示例

```
# 获得主推的撤单错误信息，data对应CCancelError结构
# 撤委托请求被迅投系统风控打回，或者撤销请求被柜台打回，均通过这个接口推送
def onRtnCancelError(self, data):
    print u'[onRtnCancelError]orderId:', data.m_nOrderID,
    u'm_nErrorID:', data.m_nErrorID, u'm_strErrorMsg:', data.m_strErrorMsg
```

2.4.11. 资产查询

```
reqAccountDetail(accountID, requestID, accountKey)
```

- 释义查询资金账号对应的资产，需要注意的是信用账号的资产数据需要调用reqCreditDetail接口
- 参数

参数名	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str

- 示例查询账号account_id对应的资产数据

```
# 请求账号资产数据
def reqAccountDetail(self, accountID, accountKey):
    print u'[reqAccountDetail]查询账号:', accountID, u"的资产数据"
    self.m_nRequestId += 1
    self.m_client.reqAccountDetail(accountID, self.m_nRequestId, accountKey)
```

2.4.12. 资产查询的回调

```
onReqAccountDetail(self,accountID, requestID, data, isLast, error)
```

- 释义调用reqAccountDetail查询资产后的回调接口
- 参数

参数	参数释义	类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	资产对象	CAccountDetail
isLast	是否为最后一条，这个值永远是True	bool
error	错误消息对象	XtError

- 示例

```
#请求资金数据的回调函数
def onReqAccountDetail(self,accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onReqAccountDetail] success, accountID:',\
        accountID, ', isLast:', isLast
    else:
        print u'[onReqAccountDetail] failed, accountID:', \
        accountID, ', errorMsg:',error.errorMsg()
```

2.4.13. 委托查询

```
reqOrderDetail(accountID, requestID, accountKey)
```

- 释义查询资金账号对应的当日所有委托
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str

- 示例查询账号account_id对应的当日所有委托

```
# 请求委托明细
def reqOrderDetail(self, accountID, accountKey):
    print u'[reqOrderDetail]查询账号:', accountID, u"的当日委托明细"
    self.m_nRequestId += 1
    self.m_client.reqOrderDetail(accountID, self.m_nRequestId, accountKey)
```

2.4.14. 委托查询的回调

```
onReqOrderDetail(self,accountID, requestID, data, isLast, error)
```

- 释义调用reqOrderDetail查询账号的当日委托后的回调接口
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	委托明细对象	COrderDetail
isLast	是否为最后一条，返回最后一条委托时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#委托明细查询的回调函数
def onReqOrderDetail(self,accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onReqOrderDetail] sucess, accountID:',
        accountID, ', isLast:', isLast
    else:
        print u'[onReqOrderDetail] failed, errorMsg:',error.errorMsg()
```

2.4.15. 成交查询

```
reqDealDetail(accountID, requestID, accountKey)
```

- 释义查询账号account_id对应的当日所有成交
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str

- 示例

```
# 请求成交明细
def reqDealDetail(self, accountID, accountKey):
    print u'[reqDealDetail]查询账号:', accountID, u"的当日成交明细"
    self.m_nRequestId += 1
    self.m_client.reqDealDetail(accountID, self.m_nRequestId, accountKey)
```

2.4.16. 成交查询的回调

```
onReqDealDetail(self,accountID, requestID, data, isLast, error)
```

- 释义调用reqDealDetail查询账号的当日成交后的回调接口
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	成交明细对象	CDealDetail
isLast	是否为最后一条，返回最后一条成交明细时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#成交明细查询的回调函数
def onReqDealDetail(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onReqDealDetail] sucess, accountID:', accountID,\
        ', isLast:', isLast
    else:
        print u'[onReqDealDetail] failed, accountID:', accountID, \
        ', errorMsg:',error.errorMsg()
```

2.4.17. 持仓明细查询

```
reqPositionDetail(accountID, requestID, accountKey)
```

- 释义查询资金账号对应的持仓
- 参数

参数名	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str

- 示例查询账号account_id对应的持仓明细

```
# 请求持仓明细
def reqPositionDetail(self, accountID, accountKey):
    print u'[reqPositionDetail]查询账号:', accountID, u"的持仓明细"
    self.m_nRequestId += 1
    self.m_client.reqPositionDetail(accountID, self.m_nRequestId, accountKey)
```

2.4.18. 持仓明细查询的回调

```
onReqPositionDetail(self, accountID, requestID, data, isLast, error)
```

- 释义调用reqPositionDetail查询账号的当日持仓明细后的回调接口
- 参数

参数名	参数释义	类型
accountID	参数释义资金账号	str
requestID	参数释义请求ID，每个请求的ID自增	int
data	参数释义持仓明细对象	CPositionDetail
isLast	参数释义是否为最后一条，返回最后一条持仓明细时这个值是True，其它情况是False	bool
error	参数释义错误消息对象	XtError

- 示例

```
#请求持仓明细的回调函数
def onReqPositionDetail(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onReqPositionDetail] sucess, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onReqPositionDetail] failed, accountID:', \
        accountID, ', errorMsg:',error.errorMsg()
```

2.4.19. 持仓统计查询

```
reqPositionStatics(accountID, requestID, accountKey)
```

- 释义查询资金账号对应的持仓统计
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str

- 示例查询账号account_id对应的持仓统计

```
# 请求持仓统计
def reqPositionStatics(self, accountID, accountKey):
    print u'[reqPositionStatics]查询账号:', accountID, u"的持仓明细"
    self.m_nRequestId += 1
    self.m_client.reqPositionStatics(accountID, self.m_nRequestId, accountKey)
```

2.4.20. 持仓统计查询的回调

```
onReqPositionStatics(self, accountID, requestID, data, isLast, error)
```

- 释义调用reqPositionStatics查询账号持仓统计后的回调接口
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	持仓统计对象	CPositionStatics
isLast	是否为最后一条，返回最后一条持仓统计时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#请求持仓统计的回调函数
def onReqPositionStatics(self,accountID, requestID, data, isLast,error):
    if error.isSuccess():
        print u'[onReqPositionStatics] sucess, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onReqPositionStatics] failed, accountID:', \
        accountID, ', errorMsg:',error.errorMsg()
```

2.4.21. 请求信用账号标的信息

```
reqStksubjects(self, *accountID*, r*equest_id,*accountKey*)*
```

- 释义请求信用账号标的信息
- 参数

参数名	参数释义	类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例查询账号account_id对应的信用账号标的信息

```
# 请求信用账号标的信息
def reqStksubjects(self, accountID, accountKey):
    print(u'[reqStksubjects]查询账号:', accountID, u"的信用账号标的信息")
    self.m_nRequestId += 1
    self.m_client.reqStksubjects(accountID, self.m_nRequestId, accountKey)
```

2.4.22. 请求信用账号标的信息的回调

该方法是请求信用账号标的信息的回调，方法如下：

```
onReqStksubjects(self, *accountID*, r*equest_id*, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	证券信息	str
isLast	是否最后一次响应	bool
error	异常信息	XtError

- 示例：

```
# 请求两融账号标的的回调函数
def onReqStksubjects(self, accountID, requestID, data, isLast, error):
    """
    请求两融账号标的的回调函数。
    """
    if error.isSuccess():
        print(u'[onReqStksubjects] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onReqStksubjects] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

2.4.23. 请求期权账号备兑持仓信息

```
reqCoveredStockPosition(self, accountID, requestID, accountKey)
```

- 释义：请求期权账号备兑持仓信息。
- 参数：

参数名	参数释义	类型
accountID	账号ID。	str
requestID	客户自己维护的请求顺序ID。	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号。	str

- 示例：查询账号account_id对应的期权账号备兑持仓信息。

```
# 请求期权账号备兑持仓信息
def reqCoveredStockPosition(self, accountID, accountKey):
    """
    查询账号account_id对应的期权账号备兑持仓信息。
    """
    print(u'[reqCoveredStockPosition]查询账号:', accountID, \
    u"的期权账号备兑持仓信息")
    self.m_client.reqCoveredStockPosition(accountID, requestID, accountKey)
```

2.4.24. 请求期权账号备兑持仓的回调函数

```
onReqCoveredStockPosition(self, accountID, requestID, data, isLast, error)
```

- 释义：请求期权账号备兑持仓的回调函数。
- 参数：

参数名称	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
data	期权账号备兑持仓数据	CCoveredStockPosition
isLast	请求数据可能有多条，需要多次回调该函数，标记是否是一次请求的最后一次回调。最后一条期权账号备兑持仓数据时，这个值为True；否则为False。	bool
error	错误信息	XtError

- 示例：

```
# 请求期权账号备兑持仓的回调函数
def onReqCoveredStockPosition(self, accountID, requestID, data, isLast, error):
    """
    请求期权账号备兑持仓的回调函数。
    """
    if error.isSuccess():
        print(u'[onReqCoveredStockPosition] sucess, accountID:',\
         accountID, ', isLast:', isLast)
    else:
        print(u'[onReqCoveredStockPosition] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

2.4.25. 请求期权账号组合持仓信息

```
reqStkOptCombPositionDetail(self, accountID, requestID, accountKey):
```

- 释义：请求期权账号组合持仓信息。

- 参数：

参数名称	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例：

```
# 请求期权账号组合持仓信息
def reqStkOptCombPositionDetail(self, accountID, accountKey):
    """
    请求期权账号组合持仓信息。
    """
    print(u'[reqStkOptCombPositionDetail]查询账号:', accountID, \
    u'的期权账号组合持仓信息')
    self.m_nRequestId += 1
    self.m_client.reqStkOptCombPositionDetail(accountID, self.m_nRequestId, \
    accountKey)
```

2.4.26. 请求期权账号组合持仓信息的回调

```
onReqStkOptCombPositionDetail(self, accountID, requestID, data, isLast, error)
```

- 释义：调用reqStkOptCombPositionDetail函数查询期权账号组合持仓信息后的回调函数。
- 参数：

参数名称	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
data	期权账号组合持仓数据	CStockOptionCombPositionDetail
isLast	是否为最后一条，返回最后一条持仓数据时这个值是True，其它情况是False	bool

参数名称	参数释义	参数类型
error	错误消息对象	XtError

- 示例：

```
# 请求期权账号组合持仓数据的回调函数
def onReqStkOptCombPositionDetail(self, accountID, requestID, data, isLast, error):
    """
    调用reqStkOptCombPositionDetail函数查询期权账号组合持仓信息后的回调函数。
    """
    if error.isSuccess():
        print(u'[onReqStkOptCombPositionDetail] success, accountID:',\
         accountID, ', 'isLast:', 'isLast)
    else:
        print(u'[onReqStkOptCombPositionDetail] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

2.4.27. 请求账号历史委托明细

```
reqHistoryOrderDetail(self, accountID, start_date, end_date, requestID, accountKey)
```

- 释义：请求账号历史委托明细。
- 参数：

参数名称	参数释义	参数类型
accountID	资金账号	str
start_date	查询开始时间（yyyy-MM-dd）	str
end_date	查询结束时间（yyyy-MM-dd）	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例：

```
# 请求账号历史委托明细
def reqHistoryOrderDetail(self, accountID, start_date, end_date, accountKey):
    """
    请求账号历史委托明细。
    """
    print(u'[reqHistoryOrderDetail] 查询账号', accountID, u'历史委托明细')
    self.m_nRequestId += 1
    self.m_client.reqHistoryOrderDetail(accountID, start_date, end_date,
    self.m_nRequestId, accountKey)
```

2.4.28. 请求账号历史委托明细的回调函数

```
onReqHistoryOrderDetail(self, accountID, requestID, data, isLast, error)
```

- 释义调用reqHistoryOrderDetail查询账号历史委托明细后的回调接口
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	历史委托明细对象	COrderDetail
isLast	是否为最后一条，返回最后一条历史委托明细时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
# 请求账号历史委托明细的回调函数
def onReqHistoryOrderDetail(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print(u'[onReqHistoryOrderDetail] success, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onReqHistoryOrderDetail] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

2.4.29. 请求账号历史成交明细

```
reqHistoryDealDetail(self, accountID, start_date, end_date, requestID, accountKey)
```

- 释义：请求账号历史成交明细
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
start_date	开始日期（yyyy-MM-dd）	str
end_date	结束日期（yyyy-MM-dd）	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
# 请求账号历史成交明细
def reqHistoryDealDetail(self, accountID, start_date, end_date, accountKey = ''):
    print(u'[reqHistoryDealDetail]查询账号:', accountID, u"历史成交明细")
    self.m_nRequestId += 1
    self.m_client.reqHistoryDealDetail(accountID, start_date, end_date,
    self.m_nRequestId, accountKey)
```

2.4.30. 历史成交查询回调

```
onReqHistoryDealDetail(self, accountID, requestID, data, isLast, error)
```

- 释义调用reqHistoryDealDetail查询账号历史成交明细后的回调接口
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int

参数名称	参数释义	参数类型
data	成交明细对象	CDealDetail
isLast	是否为最后一条，返回最后一条成交明细时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#请求历史成交查询的回调函数
def onReqHistoryDealDetail(self,accountID, requestID, data, isLast,error):
    if error.isSuccess():
        print u'[onReqHistoryDealDetail] sucess, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onReqHistoryDealDetail] failed, accountID:', \
        accountID, ', errorMsg:',error.errorMsg()
```

2.4.31. 请求账号历史持仓统计

```
reqHistoryPositionStatics(self, accountID, start_date, end_date, requestID, accountKey)
```

- 释义请求账号历史持仓统计
- 参数

参数名称	参数释义	参数类型
accountID	账号ID	str
start_date	查询开始时间	str
end_date	查询结束时间	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
# 请求账号历史持仓统计
def reqHistoryPositionStatics(self, accountID, start_date, end_date, accountKey=""):
    print(u'[reqHistoryPositionStatics]查询账号:', accountID, \
    u"的历史持仓统计")
    self.m_nRequestId += 1
    self.m_client.reqHistoryPositionStatics(accountID, start_date, \
    end_date, self.m_nRequestId, accountKey)
```

2.4.32. 请求账号历史持仓统计的回调

```
onReqHistoryPositionStatics(self, accountID, requestID, data, isLast, error)
```

- 释义调用reqHistoryPositionStatics查询账号历史持仓统计后的回调接口
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	持仓统计数据	CPositionStatics
isLast	是否为最后一条，返回最后一条持仓统计时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#请求历史持仓统计的回调函数
def onReqHistoryPositionStatics(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print(u'[onReqHistoryPositionStatics] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onReqHistoryPositionStatics] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

2.4.33. 订阅行情数据

```
subscribQuote(self, data, requestID)
```

- 释义订阅行情数据，当code为"allCode"时，订阅整个市场，调用此函数后，回调onSubscribQuote。
- 参数

参数名	参数释义	参数类型
data	订阅行情参数	CSubscribData
requestID	客户自己维护的请求顺序ID	int

- 示例

```
# 订阅行情数据
def subscribQuote(self, data):
    self.m_nRequestId += 1
    self.m_client.subscribQuote(data, self.m_nRequestId)
```

2.3.34. 行情订阅的回调函数

```
onSubscribQuote(self, requestID, data, error)
```

- 释义：调用subscribQuote订阅行情数据后的回调接口
- 参数：

参数名称	参数释义	参数类型
requestID	请求ID，每个请求的ID自增	int
data	行情订阅对象	CSubscribData
error	错误消息对象	XtError

- 示例：

```
# 行情订阅回调函数
def onSubscribQuote(self, requestID, data, error):
    if error.isSuccess():
        print("行情订阅成功！")
    else:
        print("行情订阅失败，错误信息：", error.errorMsg())
```

2.4.35. 退订行情数据

```
unSubscribQuote(self, data, requestID)
```

- 释义：退订行情数据，当code为"allCode"时，退订整个市场，调用此函数后，回调onUnSubscribQuote。
- 参数

参数名	参数释义	参数类型
data	订阅行情参数	CSubscribData
requestID	客户自己维护的请求顺序ID	int

- 示例

```
# 退订行情数据
def unSubscribQuote(self, data):
    self.m_nRequestId += 1
    self.m_client.unSubscribQuote(data, self.m_nRequestId)
```

2.4.36. 退订行情数据的回调函数

```
onUnSubscribQuote(self, requestID, data, error)
```

- 释义行情退订的回调函数。
- 参数

参数名	参数释义	参数类型
requestID	客户自己维护的请求顺序ID	int

参数名	参数释义	参数类型
data	行情订阅数据	CSubscribData
error	错误信息	XtError

- 示例

```
# 行情退订回调函数
def onUnSubscribQuote(self, requestID, data, error):
    if not error.isSuccess():
        print("退订行情失败: ", error.errorMsg())
    else:
        print("退订行情成功")
```

参数名	参数释义	参数类型
data	行情数据	CPriceData

2.4.37. 获取XtTraderApi实例

```
createXtTraderApi(address)
```

- 释义获取XtTraderApi实例。
- 参数

参数名	参数释义	参数类型
address	XtApiService监听端口	str

- 返回值 XTraderApi实例对象

```
api = XtTraderApi.createXtTraderApi("127.0.0.1:65300")
```

2.4.38. 启动XtTraderApi多实例线程

```
joinAll()
```

- 释义启动XtTraderApi多实例线程，多实例情况下必须使用该函数，同时单实例也可调用该函数。
- 参数 无

```
XtTraderApi.joinAll()
```

2.4.39. 销毁XtTraderApi多实例线程

```
destroyAll()
```

- 释义销毁XtTraderApi多实例线程，所有实例停止工作，joinAll()函数解除等待状态。
- 参数 无

```
XtTraderApi.destroyAll()
```

2.4.40. 设置数据回调对象

```
setCallback(pCallback)
```

- 释义设置数据回调对象。
- 参数

参数名	参数释义	参数类型
pCallback	XtTraderApiCallback类实例	XtTraderApiCallback

- 返回值 无

```
api.setCallback(callback)
```

2.4.41. 初始化XtTraderApi实例

```
init(configFilePath="../config")
```

- 释义创建api实例，并进行初始化。

- 参数

参数名	参数释义	参数类型
configFilePath	配置文件夹目录，默认是"../config"，运行目录上一层的config下	str

- 返回值 初始化是否成功

```
api.init(configFilePath="../config")
```

2.4.42. 析构XtTraderApi实例

```
destroy()
```

- 释义析构XtTraderApi实例。
- 参数 无
- 返回值 无

```
api.destroy()
```

2.4.43. 启动XtTraderApi单实例线程

```
join()
```

- 释义启动XtTraderApi单实例线程，多实例情况必须调用joinAll()函数。
- 参数 无

```
api.join()
```

2.4.44. 用户登陆

```
userLogin(self, user_name, password, requestID, machine_info=None, \
appid=None, authcode=None)
```

- 释义 用户登陆，调用此函数后，回调onUserLogin。
- 参数

参数名	参数释义	参数类型
user_name	用户名	str
password	用户密码	str
requestID	客户自己维护的请求顺序ID	int
machine_info	如果需要下单设置站点信息，则传入，否则可以传空	str
appid	如果是期货中继模式，需传入监管处申请的appid，则传入，否则可以传空	str
authcode	如果是期货中继模式，需传入监管处申请的authcode，则传入，否则可以传空	str

- 示例

```
# 用户登陆
def userLogin(self, user_name, password):
    self.m_nRequestId += 1
    self.m_client.userLogin(user_name, password, self.m_nRequestId)
```

2.4.45. 用户登录的回调函数

```
onUserLogin(self, user_name, password, requestID, error)
```

- 释义 用户登录的回调函数，用户成功或者失败的登陆信息会在此函数回调中返回
- 参数

参数名	参数释义	参数类型
user_name	用户名	str
password	用户密码	str
requestID	客户自己维护的请求顺序ID	int
error	错误信息	XtError

- 示例

```python
# 用户登录回调
def onUserLogin(self, user_name, password, requestID, error):
    if error.isSuccess():
        print("登录成功")
    else:
        print("登录失败:", error.errorMsg())
```

2.4.46. 用户登出

```
userLogout(self, userName, password, requestID)
```

- 释义：用户登出。调用此函数后，回调onUserLogout。
- 参数：

参数名	参数释义	参数类型
userName	用户名	str
password	用户密码	str
requestID	客户自己维护的请求顺序ID	int

- 示例：

```python
# 用户登出
def userLogout(self):
    self.m_nRequestId += 1
    self.m_client.userLogout(self.m_userName, self.m_password, self.m_nRequestId)
```

2.4.47. 保留位

2.4.48. 请求行情数据信息

```
reqPriceData(self, exchange_id, instrument_id, requestID)
```

- 释义请求行情数据信息，调用此函数后，回调onReqPriceData。
- 参数

参数名	参数释义	参数类型
exchange_id	市场，取值参考MarketType.h	str
instrument_id	合约代码	str
requestID	客户自己维护的请求顺序ID	int

- 示例

```
# 请求行情数据信息
def reqPriceData(self, exchange_id, instrument_id):
    self.m_nRequestId += 1
    self.m_client.reqPriceData(exchange_id, instrument_id, self.m_nRequestId)
```

2.4.49. 请求行情数据的回调函数

```
onReqPriceData(self, requestID, data, error)
```

- 释义：请求行情数据的回调函数。
- 参数：

参数名	参数释义	参数类型
requestID	客户自己维护的请求顺序ID	int
data	行情数据	CPriceData
error	错误信息	XtError

- 示例：

```
# 请求行情数据的回调函数
def onReqPriceData(self, requestID, data, error):
    if error.isSuccess():
        # 成功
        print("请求行情数据成功")
        # TODO: 处理行情数据
    else:
        # 失败
        print(f"请求行情数据失败，错误信息：{error.getErrorMsg()}")
```

注意事项：

- 当**`error`返回报错时，`data`**内容不可用。

2.4.50. 按市场请求行情数据信息

```
reqPriceDataByMarket(self, exchange_id, requestID)
```

- 释义按市场请求行情数据信息，调用此函数后，回调onReqCInstrumentDetail。
- 参数

参数名	参数释义	参数类型
exchange_id	市场，取值参考MarketType.h	str
requestID	客户自己维护的请求顺序ID	int

- 示例

```
# 按市场请求行情数据信息
def reqPriceDataByMarket(self, exchange_id):
    self.m_nRequestId += 1
    self.m_client.reqPriceDataByMarket(exchange_id, self.m_nRequestId)
```

2.4.51. 请求账号期权行情信息

```
reqInstrumentDetail(self, accountID, requestID, accountKey="")
```

- 释义请求账号期权行情信息，调用此函数后，回调onReqCInstrumentDetail。
- 参数

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
# 请求账号期权行情信息
def reqInstrumentDetail(self, accountID):
    self.m_nRequestId += 1
    self.m_client.reqInstrumentDetail(accountID, self.m_nRequestId)
```

2.4.52. 请求合约数据的回调函数

```
onReqCInstrumentDetail(self, accountID, requestID, data, isLast, error)
```

- 释义请求合约数据的回调函数。
- 参数

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
data	合约数据数据	CInstrumentDetail
isLast	请求数据可能有多条，需要多次回调该函数，标记是否是一次请求的最后一次回调	bool
error	错误信息	XtError

- 示例

```
# 请求合约数据的回调函数
def onReqCInstrumentDetail(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print("请求合约数据成功！")
        # 处理合约数据
        do something
    else:
        print("请求合约数据失败，错误原因：", error.errorMsg())
```

2.4.53. 请求账号汇率信息

```
reqGGTReferenceRate(self, accountID, requestID, accountKey="")
```

- 释义：请求账号汇率信息，调用此函数后，回调onReqReferenceRate。
- 参数：

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例：

```
# 请求账号汇率信息
def reqGGTReferenceRate(self, accountID):
    self.m_nRequestId += 1
    self.m_client.reqGGTReferenceRate(accountID, self.m_nRequestId)
```

2.4.54. 请求账号汇率信息的回调函数

```
onReqReferenceRate(self, accountID, requestID, data, isLast, error)
```

- 释义请求账号汇率信息的回调函数。
- 参数

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
data	港股账号汇率数据	CReferenceRate
isLast	请求数据可能有多条，需要多次回调该函数，标记是否是一次请求的最后一次回调	bool
error	错误信息	XtError

- 示例

```
# 请求账号汇率信息的回调函数
def onReqReferenceRate(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print("请求账号汇率信息成功！")
        # 处理账号汇率数据
        do something
    else:
        print("请求账号汇率信息失败，错误原因：", error.errorMsg())
```

2.4.55. 请求两融账号综合资金信息

```
reqCreditDetail(self, accountID, requestID, accountKey="")
```

- 释义：请求两融账号综合资金信息，调用此函数后，回调onReqCreditDetail。
- 参数：

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例：

```
# 请求两融账号综合资金信息
def reqCreditDetail(self, accountID):
    self.m_nRequestId += 1
    self.m_client.reqCreditDetail(accountID, self.m_nRequestId)
```

2.4.56. 请求两融账号两融综合资金数据的回调函数

```
onReqCreditDetail(self, accountID, requestID, data, isLast, error)
```

- 释义请求两融账号两融综合资金数据的回调函数。
- 参数

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
data	两融账号两融综合资金数据数据	CCreditDetail
isLast	请求数据可能有多条，需要多次回调该函数，标记是否是一次请求的最后一次回调	bool
error	错误信息	XtError

- 示例

```
# 请求两融账号两融综合资金数据的回调函数
def onReqCreditDetail(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print("请求两融账号两融综合资金数据成功！")
        # 处理两融综合资金数据
        do something
    else:
        print("请求两融账号两融综合资金数据失败，错误原因：", error.errorMsg())
```

2.4.57. 组合算法单下单，只支持股票

```
order(self, order_info, requestID, accountKey="")
```

- 释义组合算法单下单，只支持股票，回调onOrder。
- 参数

参数名	参数释义	参数类型
order_info	组合算法单请求参数	CGroupOrder
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
# 组合算法单下单，只支持股票
def order(self, accountID, volume, direction, order_type):
    group_order = CGroupOrder()
                group_order.m_strMarket = ['SH','SZ']
    self.m_client.order(group_order, self.m_nRequestId)
```

2.4.58. 组合智能算法单下单

```
order(self, order_info, requestID, accountKey="")
```

- 释义组合智能算法单下单，支持股票，期货，个股期权，期货期权，沪港通，深港通，回调onOrder。
- 参数

参数名	参数释义	参数类型
order_info	组合智能算法单请求参数	CAlgGroupOrder
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
# 组合智能算法单下单
def order(self, accountID, volume, direction, order_type, algo_name):
    alg_order = CAlgGroupOrder()
    self.m_nRequestId += 1
    self.m_client.order(alg_order, self.m_nRequestId)
```

2.4.59. 组合外部算法单下单

```
order(self, order_info, requestID, accountKey="")
```

- 释义组合外部算法单下单，支持股票，期货，个股期权，期货期权，沪港通，深港通。
- 参数

参数名	参数释义	参数类型
order_info	组合外部算法单请求参数	CExternAlgGroupOrder
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
# 组合外部算法单下单
def order_external_alg_group(self, accountID, instrument, exchange_id, \
hedge_flag, direction, price, volume, order_type):
    # 构造下单参数
    order_info = CExternAlgGroupOrder()
    # 见快速入门
    # 发送下单请求
    self.m_nRequestId += 1
    self.m_client.order(order_info, self.m_nRequestId)
```

2.4.60. 算法单下单

```
order(self, order_info, requestID, accountKey="")
```

- 释义算法单下单，支持股票，期货，个股期权，期货期权，沪港通，深港通。
- 参数

参数名	参数释义	参数类型
order_info	算法单请求参数	CAlgorithmOrder
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
# 算法单下单
def order_algorithm(self, accountID, instrument, exchange_id, direction, \
price, volume, order_type):
    # 构造下单参数
    order_info = CAlgorithmOrder()
    # 见快速入门
    # 发送下单请求
    self.m_nRequestId += 1
    self.m_client.order(order_info, self.m_nRequestId)
```

2.4.61. 智能算法下单

```
order(self, order_info, requestID, accountKey)
```

- 释义智能算法下单，支持股票，期货，个股期权，期货期权，沪港通，深港通。
- 参数

参数名	参数释义	参数类型
order_info	智能算法交易请求参数	CIntelligentAlgorithmOrder
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

2.4.62. 主动算法下单

```
order(self, order_info, requestID, accountKey)
```

- 释义主动算法下单，支持股票，期货，个股期权，期货期权，沪港通，深港通。
- 参数

参数名	参数释义	参数类型
order_info	主动算法交易请求参数	CExternAlgorithmOrder
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相	str

参数名	参数释义	参数类型
	同的账号	

2.4.63. 普通组合下单

```
order(self, order_info, requestID, accountKey)
```

- 释义：普通组合下单，支持股票，期货，个股期权，期货期权，沪港通，深港通。
- 参数：

参数名	参数释义	参数类型
order_info	普通组合交易请求参数	COrdinaryGroupOrder
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

2.4.64. 上传终端ctp采集信息

```
registerUserSystemInfo(self, accountID, ip_port_addr, info_len, \
ctp_system_info, requestID, accountKey)
```

- 释义：上传终端ctp采集信息，用于传入ctp柜台需要留痕的终端信息。
- 参数：

参数名	参数释义	参数类型
accountID	账号ID	str
ip_port_addr	上传终端信息机器的ip和端口信息(格式为ip:port，例如127.0.0.1:58000)	str
info_len	ctp采集信息内容的长度len，即CTP_GetSystemInfo入参nLen	int
ctp_system_info	ctp采集到的内容	str

参数名	参数释义	参数类型
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

2.4.65. 设置用户下单指令冻结选项

```
setCmdFrzCheckOption(self, cmd_freeze_check_option)
```

- 释义：设置用户下单指令冻结选项。
- 参数：

参数名	参数释义	参数类型
cmd_freeze_check_option	用户下单指令冻结选项，1 禁止 2 警告	int

2.4.66. 获取用户下所有账号key

```
reqAccountKeys(self, requestID)
```

- 释义：获取用户下所有账号key，调用此函数后，回调 onReqAccountKey。
- 参数：

参数名	参数释义	参数类型
requestID	客户自己维护的请求顺序ID	int

2.4.67. 获取用户下所有账号key的回调函数

```
onReqAccountKey(self, requestID, data, isLast, error)
```

- 释义：获取用户下所有账号key的回调函数。
- 参数：

参数名	参数释义	参数类型
requestID	客户自己维护的请求顺序ID	int

参数名	参数释义	参数类型
data	账号key数据	CAccountKey
isLast	请求数据可能有多条，需要多次回调该函数，标记是否是一次请求的最后一次回调	bool
error	错误信息	XtError

- 示例：

```
# 获取用户下所有账号key的回调函数
def onReqAccountKey(self, requestID, data, isLast, error):
    if error.isSuccess():
        print("获取账号key成功！")
        # 处理账号key数据
        do something
    else:
        print("获取账号key失败，错误原因：", error.errorMsg())
```

2.4.68. 主推的委托错误信息回调函数

```
onRtnOrderError(self, data)
```

- 释义：主推的委托错误信息。
- 参数：

参数名	参数释义	参数类型
data	下单失败的错误信息	COrderError

2.4.69. 请求账号普通柜台资金

```
reqComFund(self, accountID, requestID, accountKey)
```

- 释义：请求账号普通柜台资金。调用此函数后，回调 onReqComFund。
- 参数：

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

2.4.70. 请求账号普通柜台资金的回调函数

```
onReqComFund(self, accountID, requestID, data, isLast, error)
```

- 释义：请求账号普通柜台资金的回调函数。
- 参数：

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
data	账号资金划拨普通柜台资金数据	CStockComFund
isLast	请求数据可能有多条，需要多次回调该函数，标记是否是一次请求的最后一次回调	bool
error	错误信息	XtError

- 示例

```
# 请求账号普通柜台资金的回调函数
def onReqComFund(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print("请求账号普通柜台资金成功！")
        # 处理账号资金数据
        do something
    else:
        print("请求账号普通柜台资金失败，错误原因：", error.errorMsg())
```

2.4.71. 请求账号普通柜台持仓

```
reqComPosition(self, accountID, requestID, accountKey)
```

- 释义：请求账号普通柜台持仓。调用此函数后，回调 onReqComPosition。
- 参数：

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

2.4.72. 请求账号普通柜台持仓的回调函数

```
onReqComPosition(self, accountID, requestID, data, isLast, error)
```

- 释义：请求账号普通柜台持仓的回调函数。
- 参数：

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
data	账号股份划拨普通柜台持仓数据	CStockComPosition
isLast	请求数据可能有多条，需要多次回调该函数，标记是否是一次请求的最后一次回调	bool
error	错误信息	XtError

- 示例：

```
# 请求账号普通柜台持仓的回调函数
def onReqComPosition(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print(f"请求账号{accountID}普通柜台持仓成功！")
        # 处理股份划拨普通柜台持仓数据
        do something
    else:
        print(f"请求账号{accountID}普通柜台持仓失败，错误原因：", error.errorMsg())
```

2.4.73. 请求账号可下单量

```
reqCanOrderVolume(self, opVolumeReq, requestID, accountKey)
```

- 释义：请求账号可下单量。调用此函数后，回调 onReqCanOrderOpVolume。
- 参数：

参数名	参数释义	参数类型
opVolumeReq	可下单量请求参数	COpVolumeReq
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例请求账号可下单量

```
#请求账号可下单量
def reqCanOrderVolumeSync(self, accountID, accountKey):
    reqParam = COpVolumeReq()
    reqParam.m_strAccountID = accountID
    reqParam.m_strMarket = "SH"
    reqParam.m_strInstrument = "601919"
    reqParam.m_dPrice = 9.88
    # reqParam.m_eOperationType = EOperationType.OPT_BUY
    reqParam.m_eOperationType = EOperationType.OPT_SELL
    reqParam.m_eHedgeFlag = EHedgeFlagType.HEDGE_FLAG_SPECULATION
    print("reqCanOrderVolume makret {} code {}"
    .format(reqParam.m_strMarket, reqParam.m_strInstrument))
    self.m_api.reqCanOrderVolume(reqParam, self.m_nRequestId, accountKey)
    self.m_nRequestId +=1
```

2.4.74. 请求账号可下单量的回调函数

```
onReqCanOrderVolume(self, accountID, requestID, accountKey,
market, instrument, nVolume, error)
```

- 释义：请求账号普通柜台持仓的回调函数。
- 参数：

参数名	参数释义	参数类型
accountID	账号ID	str
requestID	客户自己维护的请求顺序ID	int
accountKey	账号Key	str
market	市场	str
instrument	代码	str
nVolume	可交易数量	int
error	错误信息	XtError

- 示例 请求账号可下单量的回调：

```
def onReqCanOrderVolume(self, accountID, nRequestId, accountKey, market,\
 instrument, nVolume, error):
    if error.isSuccess():
        print("查询市场{} 代码 {}可下单量为: {}".format(market,instrument, nVolume))
    else:
        print("onReqCanOrderVolume failed, errmsg:", error.errorMsg())
```

3. 同步接口篇

3.1. 同步接口说明

本篇只描述同步接口，其余内容请参考上面。

3.1.1. 客户端用户登录 同步接口

```
userLoginSync(self, user_name, password, machine_info=None, appid=None, authcode=None)
```

- 释义 用户登陆，调用此函数后，返回 XtError结构, 判断 isSuccess
- 参数

参数名	参数释义	参数类型
user_name	用户名	str
password	用户密码	str
machine_info	如果需要下单设置站点信息，则传入，否则可以传空	str
appid	如果是期货中继模式，需传入监管处申请的appid，则传入，否则可以传空	str
authcode	如果是期货中继模式，需传入监管处申请的authcode，则传入，否则可以传空	str

- 示例

```
# 用户登陆
def userLoginSync(self, user_name, password):
    error = self.m_client.userLoginSync(user_name, password, )
        if error.isSuccess():
                pass
        else:
                pase
```

3.1.2. 资产查询 同步接口

```
reqAccountDetailSync(accountID, error, accountKey)
```

- 释义查询资金账号对应的资产，返回 accountlist 对应 CAccountDetail 结构，处理返回数据之前需要先判断error.isSuccess() 注意:信用账号的资产数据需要调用reqCreditDetail接口
- 参数

参数名	参数释义	参数类型
accountID	资金账号	str
error	错误信息	XtError
accountKey	账号key	str

- 示例查询账号account_id对应的资产数据

```
# 请求账号资产数据
def reqAccountDetailSync(self, accountID, accountKey):
    print u'[reqAccountDetailSync]查询账号:', accountID, u"的资产数据"
    error = XtError(0, "")
    accountlist = self.m_client.reqAccountDetailSync(accountID, error, \
        accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.3. 委托查询 同步接口

```
reqOrderDetailSync(accountID, error, accountKey)
```

- 释义查询资金账号对应的当日所有委托，返回 orderlist 对应 COrderDetail 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
error	错误信息	XtError
accountKey	账号key	str

- 示例查询账号account_id对应的当日所有委托

```
# 请求委托明细
def reqOrderDetailSync(self, accountID, accountKey):
    print u'[reqOrderDetailSync]查询账号:', accountID, u"的当日委托明细"
    error = XtError(0, "")
    orderlist = self.m_client.reqOrderDetailSync(accountID, error,\
    accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.4. 成交查询 同步接口

```
reqDealDetailSync(accountID, error, accountKey)
```

- 释义查询账号account_id对应的当日所有成交，返回 deallist 对应 CDealDetail 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数	参数释义	参数类型
accountID	资金账号	str

参数	参数释义	参数类型
error	错误信息	XtError
accountKey	账号key	str

- 示例

```
# 请求成交明细
def reqDealDetailSync(self, accountID, accountKey):
    print u'[reqDealDetailSync]查询账号:', accountID, u"的当日成交明细"
    error = XtError(0, "")
    deallist = self.m_client.reqDealDetailSync(accountID, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.5. 持仓明细查询 同步接口

```
reqPositionDetailSync(accountID, error, accountKey)
```

- 释义查询资金账号对应的持仓，返回 posdetailslist 对应 CPositionDetail 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名	参数释义	参数类型
accountID	资金账号	str
error	错误信息	XtError
accountKey	账号key	str

- 示例查询账号account_id对应的持仓明细

```
# 请求持仓明细
def reqPositionDetailSync(self, accountID, accountKey):
    print u'[reqPositionDetailSync]查询账号:', accountID, u"的持仓明细"
    error = XtError(0, "")
    posdetailslist = self.m_client.reqPositionDetailSync(
    accountID, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.6. 持仓统计查询 同步接口

```
reqPositionStaticsSync(accountID, error, accountKey)
```

- 释义查询资金账号对应的持仓统计，返回 posstaticslist 对应 CPositionStatics 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
error	错误信息	XtError
accountKey	账号key	str

- 示例查询账号account_id对应的持仓统计

```
# 请求持仓统计
def reqPositionStaticsSync(self, accountID, accountKey):
    print u'[reqPositionStaticsSync]查询账号:', accountID, u"的持仓明细"
    error = XtError(0, "")
    posstaticslist = self.m_client.reqPositionStaticsSync(accountID,
    error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.7. 普通单下单 同步接口

```
orderSync(self, order_info, error, accountKey="")
```

- 释义普通单下单，支持股票，期货，个股期权，期货期权，沪港通，深港通，返回指令号orderId，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名	参数释义	参数类型
order_info	普通单请求参数	COrdinaryOrder
error	错误信息	XtError
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
def testStockOrdinaryOrder(self, accountID, accountKey):
    print(u'[orderSync]股票普通单, accountId:', accountID)
    orderInfo = COrdinaryOrder()  # 初始化数据，普通单结构体
    # 资金账号，必填参数。不填会被api打回，并且通过onOrder反馈失败
    orderInfo.m_strAccountID = accountID
    orderInfo.m_dPrice = 14.5  # 报单价格
    # 单笔超价百分比，选填字段。默认为0
    orderInfo.m_dSuperPriceRate = 0
    # 报单委托量，必填字段。默认int最大值，填0或不填会被api打回
    orderInfo.m_nVolume = 400
    # 报单市场。必填字段。股票市场有"SH"/"SZ"，如果填空或填错都会被api直接打回
    orderInfo.m_strMarket = "SH"
    orderInfo.m_strInstrument = "600004"  # 报单合约代码，必填字段
    # 枚举类型，报单价格类型，必填字段
    orderInfo.m_ePriceType = EPriceType.PRTP_FIX
    # 报单委托类型。必填字段
    orderInfo.m_eOperationType = EOperationType.OPT_BUY
    orderInfo.m_strRemark = "alpha"
    orderInfo.m_strStrategyID = "81844999998980442228";
    error = XtError(0, "")
    order = self.m_client.orderSync(orderInfo, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.8. 撤指令 同步接口

```
cancelSync(order_id, error, accountKey="")
```

- 释义根据指令编号order_id对指令进行撤单操作，order_id对应orderSync里的m_nOrderID以及COrderDetail里的m_nOrderID，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名称	参数释义	参数类型
order_id	指令编号	int
error	错误信息	XtError

参数名称	参数释义	参数类型
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例对指令编号为order_id的指令进行撤销

```
#撤销指令
def cancel_cmd(self, order_id, accountKey):
    print u'[cancel_cmd] 撤指令, 指令编号:', order_id
    error = XtError(0, "")
    canceldata = self.m_client.cancelSync(order_id, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.9. 撤委托 同步接口

```
cancelOrderSync(accountID, order_sysid, market, code, error, accountKey)
```

- 释义根据券商柜台返回的合同编号对委托进行撤单操作，error，isSuccess()判断是否撤单成功
- 参数

参数名	参数释义	参数类型
accountID	证券账号	str
order_sysid	合同编号	str
market	市场代码	str
code	合约代码	str
error	错误信息	XtError
accountKey	账号key	str

- 示例对股票资金账号account_id合同编号为order_sysid的委托进行撤单

```
#撤销委托
def cancelOrderSync(self, accountID, order_sysid, market, code, accountKey):
    print (u'[cancelOrderSync] 撤委托，合同编号:', accountID, ', ',
    order_sysid,  ', ', market,  ', ', code,  ', ', accountKey,
    ', ', order_sysid)
    error = XtError(0, "")
    canceldata = self.m_client.cancelOrderSync(accountID, order_sysid,
    market, code, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.10. 请求行情数据信息 同步接口

```
reqPriceDataSync(exchangeId, instrumentId, error)
```

- 释义查询行情数据信息，返回 priceDataList 对应 CPriceData 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数	参数释义	参数类型
exchangeId	exchangeId 市场，取值参考MarketType.h	str
instrumentId	合约代码	str
error	错误信息	XtError

- 示例查询账号account_id对应的持仓统计

```
# 查询行情数据信息
def reqPriceDataSync(self, exchangeId, instrumentId):
    print u'[reqPriceDataSync]查询合约:', instrumentId, u"的实时行情信息"
    error = XtError(0, "")
    priceDataList = self.m_client.reqPriceDataSync(exchangeId, instrumentId, error)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.11. 请求用户所有账号Key信息 同步接口

```
reqAccountKeysSync(error)
```

- 释义：获取用户下所有账号key，返回 accList 对应 CAccountKey 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数	参数释义	参数类型

| error | 错误信息 | XtError |

- 示例查询用户下所有账号key

```
# 获取用户下所有账号key
def reqAccountKeysSync(self):
    print u'[reqAccountKeysSync]查询用户下所有账号key"
    error = XtError(0, "")
    accList = self.m_client.reqAccountKeysSync(error)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.12. 组合智能算法单下单 同步接口

```
orderSync(self, order_info, error, accountKey="")
```

- 释义组合智能算法单下单，支持股票，期货，个股期权，期货期权，沪港通，深港通，返回指令号orderId，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名	参数释义	参数类型
order_info	组合智能算法单请求参数	CAlgGroupOrder
error	错误信息	XtError
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
def testStockOrdinaryOrder(self, accountID, accountKey):
    orderInfo = CAlgGroupOrder()   # 算法单实例
    orderInfo.m_orderParam = CIntelligentAlgorithmOrder()   # 算法单实例
    orderInfo.m_orderParam.m_strAccountID = accountID    #资金账号
    # 委托类型
    orderInfo.m_orderParam.m_eOperationType = EOperationType.OPT_BUY
    orderInfo.m_orderParam.m_ePriceType = EPriceType.PRTP_MARKET  # 报价类型
    orderInfo.m_orderParam.m_dPrice = 12.3  # 委托价格
    orderInfo.m_orderParam.m_strOrderType = "TWAP"
    # 指令有效时间
    orderInfo.m_orderParam.m_nValidTimeStart = int(time.time())
    orderInfo.m_orderParam.m_nValidTimeEnd = \
    orderInfo.m_orderParam.m_nValidTimeStart + 1800
    orderInfo.m_orderParam.m_dMaxPartRate = 1  # 量比比例
    orderInfo.m_orderParam.m_dMinAmountPerOrder = 100  # 委托最小金额
    orderInfo.m_orderParam.m_dOrderRateInOpenAcution = 0.4
    orderInfo.m_orderParam.m_dPriceOffsetBpsForAuction = 400
    orderInfo.m_orderParam.m_nStopTradeForOwnHiLow = \
    EStopTradeForOwnHiLow.STOPTRADE_NONE
    orderInfo.m_orderParam.m_bOnlySellAmountUsed = True
    orderInfo.m_orderParam.m_dBuySellAmountDeltaPct = 3.0

    orderInfo.m_strMarket = ["SZ", "SH"]  # 市场
    orderInfo.m_strInstrument = ["300356", "600004"]  # 合约代码
    orderInfo.m_nVolume = [1000, 1000]  # 委托数量
    orderInfo.m_eOperationType = [EOperationType.OPT_SELL, \
    EOperationType.OPT_BUY]  # 委托数量
    orderInfo.m_nOrderNum = 2
    orderInfo.m_strRemark = 'CAlgGroupOrder' # 投资备注
    error = XtError(0, "")
    orderId = self.m_client.orderSync(orderInfo, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.13. 算法单下单 同步接口

```
orderSync(self, order_info, error, accountKey="")
```

- 释义算法单下单，支持股票，期货，个股期权，期货期权，沪港通，深港通，返回指令号orderId，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名	参数释义	参数类型
order_info	算法单请求参数	CAlgorithmOrder
error	错误信息	XtError
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
def testStockOrdinaryOrder(self, accountID, accountKey):
    orderInfo = CAlgorithmOrder()   # 算法单实例
    orderInfo.m_strAccountID = accountID    #资金账号
    orderInfo.m_strMarket = "SH"  # 市场
    orderInfo.m_strInstrument = "600000"  # 合约代码
    orderInfo.m_eOperationType = EOperationType.OPT_BUY  # 委托类型
    orderInfo.m_ePriceType = EPriceType.PRTP_FIX  # 报价类型
    orderInfo.m_dPrice = 12.3  # 委托价格
    orderInfo.m_nVolume = 1000  # 委托数量
    # 单笔基准量，默认为目标量
    orderInfo.m_eSingleVolumeType = EVolumeType.VOLUME_FIX
    orderInfo.m_dSingleVolumeRate = 0.1 # 基准量比例
    orderInfo.m_dPlaceOrderInterval = 10 # 下单间隔
    orderInfo.m_dWithdrawOrderInterval = 10 #撤单间隔
    # 价格波动区间，必填字段 0 <= orderInfo.m_dPriceRangeRate <= 1
    orderInfo.m_dPriceRangeRate = 0.1
    # 单笔最小量，股票最小为100，期货最小为1
    orderInfo.m_nSingleNumMin = 100
    orderInfo.m_nSingleNumMax = 1000 # 单笔最大量
    # 指令有效时间
    orderInfo.m_nValidTimeStart = 0
    orderInfo.m_nValidTimeEnd = orderInfo.m_nValidTimeStart + 1800
    orderInfo.m_nMaxOrderCount = 100  # 最大下单笔数
    orderInfo.m_strRemark = 'algorithm' # 投资备注
    error = XtError(0, "")
    orderId = self.m_client.orderSync(orderInfo, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.14. 智能算法下单 同步接口

```
orderSync(self, order_info, error, accountKey="")
```

- 释义智能算法单下单下单，支持股票，期货，个股期权，期货期权，沪港通，深港通，返回指令号orderId，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名	参数释义	参数类型
order_info	智能算法单请求参数	CIntelligentAlgorithmOrder
error	错误信息	XtError
accountKey	账号Key，用于区分不同类型的账号ID相同的账号	str

- 示例

```
def testStockOrdinaryOrder(self, accountID, accountKey):
    orderInfo = CIntelligentAlgorithmOrder()   # 算法单实例
    orderInfo.m_strAccountID = accountID    #资金账号
    orderInfo.m_strMarket = "SH"  # 市场
    orderInfo.m_strInstrument = "600000"  # 合约代码
    orderInfo.m_eOperationType = EOperationType.OPT_BUY  # 委托类型
    orderInfo.m_ePriceType = EPriceType.PRTP_FIX  # 报价类型
    orderInfo.m_dPrice = 12.3  # 委托价格
    orderInfo.m_nVolume = 1000  # 委托数量
    orderInfo.m_strOrderType = "VWAP"
    # 指令有效时间
    orderInfo.m_nValidTimeStart = int(time.time())
    orderInfo.m_nValidTimeEnd = orderInfo.m_nValidTimeStart + 1800
    orderInfo.m_dMaxPartRate = 1  # 量比比例
    orderInfo.m_dMinAmountPerOrder = 100  # 委托最小金额
    orderInfo.m_strRemark = 'intelligent' # 投资备注
    error = XtError(0, "")
    orderId = self.m_client.orderSync(orderInfo, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.15.查询日初持仓统计信息 同步接口

```
reqInitialPositionStaticsSync(accountID, error, accountKey)
```

- 释义请求账号日初持仓统计信息，返回 posstaticslist 对应 CPositionStatics 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
error	错误信息	XtError
accountKey	账号key	str

- 示例查询账号account_id对应的持仓统计

```
# 请求持仓统计
def reqInitialPositionStaticsSync(self, accountID, accountKey):
    print u'[reqInitialPositionStaticsSync]查询账号:', accountID,\
    u"的日初持 仓明细"
    error = XtError(0, "")
    posstaticslist = self.m_client.reqInitialPositionStaticsSync(accountID,\
    error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.16.查询用户产品信息 同步接口

```
reqProductDataSync(error)
```

- 释义请求用户产品信息，返回 productlist 对应 CProductData 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数	参数释义	参数类型
error	错误信息	XtError

- 示例查询用户产品信息

```
# 请求持仓统计
def reqProductDataSync(self, accountID, accountKey):
    print u'[reqProductDataSync]查询用户产品信息'
    error = XtError(0, "")
    productlist = self.m_client.reqProductDataSync(error)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.17. 历史委托查询 同步接口

```
reqHistoryOrderDetailSync(accountID, startDate, endDate, error, accountKey)
```

- 释义查询资金账号对应日期的历史所有委托，返回 orderlist 对应 COrderDetail 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
startDate	查询开始时间	str
endDate	查询结束时间	str
error	错误信息	XtError
accountKey	账号key	str

- 示例查询账号account_id对应的历史所有委托

```
# 请求委托明细
def reqHistoryOrderDetailSync(self, startDate, endDate, accountID, accountKey):
    print u'[reqHistoryOrderDetailSync]查询账号:', accountID, u"的历史委托明细"
    error = XtError(0, "")
    orderlist = self.m_client.reqHistoryOrderDetailSync(accountID,\
        startDate, endDate, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.18. 历史成交查询 同步接口

```
reqHistoryDealDetailSync(accountID, startDate, endDate, error, accountKey)
```

- 释义查询资金账号对应日期的历史所有成交，返回 deallist 对应 CDealDetail 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
startDate	查询开始时间	str
endDate	查询结束时间	str
error	错误信息	XtError
accountKey	账号key	str

- 示例查询账号account_id对应的历史所有成交

```
# 请求成交明细
def reqHistoryDealDetailSync(self, startDate, endDate, accountID, accountKey):
    print u'[reqHistoryDealDetailSync]查询账号:', accountID, u"的历史成交明细"
    error = XtError(0, "")
    deallist = self.m_client.reqHistoryDealDetailSync(accountID,\
        startDate, endDate, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.19. 历史持仓查询 同步接口

```
reqHistoryPositionStaticsSync(accountID, startDate, endDate, error, accountKey)
```

- 释义请求账号历史持仓统计，返回 posotionlist 对应 CPositionStatics 结构，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
startDate	查询开始时间	str
endDate	查询结束时间	str
error	错误信息	XtError
accountKey	账号key	str

- 示例查询账号account_id对应的历史所有持仓统计

```
# 请求持仓明细
def reqHistoryPositionStaticsSync(self, startDate, endDate, \
accountID, accountKey):
    print u'[reqHistoryPositionStaticsSync]查询账号:', accountID, u"的历史持仓统计"
    error = XtError(0, "")
    posotionlist = self.m_client.reqHistoryPositionStaticsSync(accountID, \
    startDate, endDate, error, accountKey)
    if error.isSuccess():
        pass
    else:
        pase
```

3.1.20. 账号状态查询 同步接口

```
reqAccountStatusSync(accountKey)
```

- 释义查询账号状态，返回值是bool
- 参数

参数名称	参数释义	参数类型
accountKey	账号key	str

- 示例查询账号状态

```
# 查询账号状态
def reqAccountStatusSync(self, accountKey):
    print u'[reqAccountStatusSync]查询账号:', accountID, u"的状态"
    accStatus = self.m_client.reqAccountStatusSync(accountKey)
    if accStatus:
```

3.1.21. 查询产品持仓统计 同步接口

```
reqPositionStaticsSyncWithProductId(accountKey)
```

- 释义查询产品持仓统计，返回值是 std::vectorxti::CPositionStatics
- 参数

参数名称	参数释义	参数类型
error	错误信息	XtError
productId	产品ID	int

- 示例查询产品持仓统计

```
# 查询产品持仓统计
def reqPositionStaticsSyncWithProductId(self, productId):
    print u'[reqPositionStaticsSyncWithProductId]查询产品:',\
        productId, u"的持仓统计"
    error = XtError(0, "")
    positionStatics = self.m_client.reqPositionStaticsSyncWithProductId(error, productId)
    if error.isSuccess():
        pass
    else:
        pass
```

3.1.22. 暂停指令 同步接口

```
pauseSync(order_id, error)
```

- 释义根据指令编号order_id对指令进行暂停操作，order_id对应orderSync里的m_nOrderID以及COrderDetail里的m_nOrderID，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名称	参数释义	参数类型
order_id	指令编号	int
error	错误信息	XtError

- 示例对指令编号为order_id的指令进行暂停

```
#暂停指令
def pauseSync(self, order_id):
    print u'[pauseSync] 暂停指令，指令编号:', order_id
    error = XtError(0, "")
    canceldata = self.m_client.pauseSync(order_id, error)
    if error.isSuccess():
        pase
    else:
        pase
```

3.1.23. 恢复指令并恢复任务 同步接口

```
resumeSync(order_id, error)
```

- 释义根据指令编号order_id对指令进行恢复操作，order_id对应orderSync里的m_nOrderID以及COrderDetail里的m_nOrderID，处理返回数据之前需要先判断error.isSuccess()
- 参数

参数名称	参数释义	参数类型
order_id	指令编号	int
error	错误信息	XtError

- 示例对指令编号为order_id的指令进行恢复

```
#恢复指令
def resumeSync(self, order_id):
    print u'[resumeSync] 恢复指令，指令编号:', order_id
    error = XtError(0, "")
    canceldata = self.m_client.resumeSync(order_id, error)
    if error.isSuccess():
        pase
    else:
        pase
```

3.1.24. 请求账户所有指令信息 同步接口

```
reqCommandsInfoSync(order_id, error)
```

- 释义请求账户所有指令信息
- 参数

参数名称	参数释义	参数类型
error	错误信息	XtError

- 示例请求账户所有指令信息

```
#请求账户所有指令信息
def reqCommandsInfoSync(self):
    error = XtError(0, "")
    canceldata = self.m_client.reqCommandsInfoSync(error)
    if error.isSuccess():
        pase
    else:
        pase
```

3.1.25. 根据指令号请求账号委托明细信息 同步接口

```
reqOrderDetailSyncByOrderID(accountID, error, orderID, accountKey)
```

- 释义根据指令号请求账号委托明细信息
- 参数

参数名称	参数释义	参数类型
accountID	账号ID	str
error	错误信息	XtError
orderID	指令号	int
accountKey	错误信息	str

- 示例根据指令号请求账号委托明细信息

```
#根据指令号请求账号委托明细信息
def reqOrderDetailSyncByOrderID(self, accountID, orderID, accountKey):
    error = XtError(0, "")
    canceldata = self.m_client.reqOrderDetailSyncByOrderID(accountID,\
     error, orderID, accountKey)
    if error.isSuccess():
        pase
    else:
        pase
```

3.1.26. 根据指令号请求账号成交明细信息 同步接口

```
reqDealDetailSyncByOrderID(accountID, error, orderID, accountKey)
```

- 释义根据指令号请求账号成交明细信息
- 参数

参数名称	参数释义	参数类型
accountID	账号ID	str
error	错误信息	XtError
orderID	指令号	int
accountKey	错误信息	str

- 示例根据指令号请求账号成交明细信息

```
#根据指令号请求账号成交明细信息
def reqDealDetailSyncByOrderID(self, accountID, orderID, accountKey):
    error = XtError(0, "")
    canceldata = self.m_client.reqDealDetailSyncByOrderID(accountID, \
    error, orderID, accountKey)
    if error.isSuccess():
        pase
    else:
        pase
```

3.1.27. 请求账号可下单量 同步接口

```
reqCanOrderVolumeSync(opVolumeReq, error, accountKey)
```

- 释义请求账号可下单量
- 参数

参数名称	参数释义	参数类型
opVolumeReq	可下单量请求参数	COpVolumeReq
error	错误信息	XtError
accountKey	账号Key	str

- 示例请求账号可下单量

```
#请求账号可下单量
def reqCanOrderVolumeSync(self, accountID, accountKey):
    error = XtError(0, "")
    reqParam = COpVolumeReq()
    reqParam.m_strAccountID = accountID
    reqParam.m_strMarket = "SH"
    reqParam.m_strInstrument = "601919"
    reqParam.m_dPrice = 9.88
    # reqParam.m_eOperationType = EOperationType.OPT_BUY
    reqParam.m_eOperationType = EOperationType.OPT_SELL
    reqParam.m_eHedgeFlag = EHedgeFlagType.HEDGE_FLAG_SPECULATION
    volume = self.m_api.reqCanOrderVolumeSync(reqParam, error, accountKey)
    if error.isSuccess():
        print("reqCanOrderVolumeSync market {}, code {}, volume {}"\
        .format(reqParam.m_strMarket, reqParam.m_strInstrument, volume))
    else:
        pase
```

3.1.28. 请求账号成交统计信息 同步接口

```
reqDealStaticsSync(accountID, accounKey)
```

- 释义查询成交统计，返回值是 std::vectorxti::CDealStatics
- 参数

参数名称	参数释义	参数类型
accountID	账号ID	str
error	错误信息	XtError
accountKey	账号Key	str

- 示例请求账号成交统计信息

```
# 查询成交统计
def reqDealStaticsSync(self, accountID, accounKey):
    print u'[reqDealStaticsSync]查询成交统计"
    error = XtError(0, "")
    dealStatics = self.m_client.reqDealStaticsSync(accountID, error, accounKey)
    if error.isSuccess():
        pass
    else:
        pass
```

3.1.29. 查询枚举名称 同步接口

```
reqEnumItemName(enumName, enumItemValue)
```

- 释义查询枚举名称，返回值是char*类型枚举名称
- 参数

参数名称	参数释义	参数类型
enumName	枚举类型	str
enumItemValue	枚举值	int

- 示例查询枚举名称

```
# 查询枚举名称
def reqEnumItemName():
    print u'[reqEnumItemName]查询成交统计"
    enumItemName = self.m_client.reqEnumItemName("EBrokerPriceType", \
    EBrokerPriceType.BROKER_PRICE_PROP_OPTION_RELEASE_COMB_STRATEGY_KKS)
    print(enumItemName)
```

4. 批量回调接口篇

4.1. 批量回调接口说明

本篇只描述批量回调接口，其余内容请参考上面。 对于有多记录数据返回的接口，本篇基于第二篇提供能一次返回多条记录的批量回调接口。 在traderApi.ini的配置项将isBatchCallback 配置为 1，requestNum配置您希望一次最大返回的数量，默认10000。如果总的数量超过配置的最大返回数量，我们将通过多次进行返回。在返回的函数中有isLast出参，判断为ture返回完成，false还会继续返回回调。

4.1.1. 资产查询的批量回调

```
onBatchReqAccountDetail(self,accountID, requestID, data, isLast, error)
```

- 释义调用reqAccountDetail查询资产后的回调接口
- 参数

参数	参数释义	类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	资产对象	存放CAccountDetail的一个list
isLast	是否为最后一次返回	bool
error	错误消息对象	XtError

- 示例

```
#请求资金数据的回调函数
def onBatchReqAccountDetail(self,accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onBatchReqAccountDetail] success, accountID:',accountID,\
        ', isLast:', isLast
    else:
        print u'[onBatchReqAccountDetail] failed, accountID:', accountID,\
        ', errorMsg:',error.errorMsg()
```

```
onBatchReqAccountDetailWithAccKey(self,accountID, requestID, accountKey, \
data, isLast, error)
```

- 释义调用reqAccountDetail查询资产后的回调接口
- 参数

参数	参数释义	类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str
data	资产对象	存放CAccountDetail的一个list
isLast	是否为最后一次返回	bool
error	错误消息对象	XtError

- 示例

```
#请求资金数据的回调函数
def onBatchReqAccountDetailWithAccKey(self,accountID, requestID,\
    accountKey, data, isLast, error):
    if error.isSuccess():
        print u'[onBatchReqAccountDetailWithAccKey] success, accountID:',\
        accountID, ', isLast:', isLast
    else:
        print u'[onBatchReqAccountDetailWithAccKey] failed, accountID:', \
        accountID, ', errorMsg:',error.errorMsg()
```

4.1.2. 信用资产查询的批量回调

```
onBatchReqCreditAccountDetail(self,accountID, requestID, data, isLast, error)
```

- 释义调用reqAccountDetail查询资产后的回调接口
- 参数

参数	参数释义	类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	资产对象	存放 CCreditAccountDetail 的一个list
isLast	是否为最后一次返回	bool
error	错误消息对象	XtError

- 示例

```
#请求资金数据的回调函数
def onBatchReqCreditAccountDetail(self,accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onBatchReqCreditAccountDetail] success, accountID:',\
        accountID, ', isLast:', isLast
    else:
        print u'[onBatchReqCreditAccountDetail] failed, accountID:', \
        accountID, ', errorMsg:',error.errorMsg()
```

```
onBatchReqCreditAccountDetailWithAccKey(self,accountID, requestID,\
 accountKey, data, isLast, error)
```

- 释义调用reqAccountDetail查询资产后的回调接口
- 参数

参数	参数释义	类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str
data	资产对象	存放 CCreditAccountDetail 的一个list
isLast	是否为最后一次返回	bool
error	错误消息对象	XtError

- 示例

```
#请求资金数据的回调函数
def onBatchReqCreditAccountDetailWithAccKey(self,accountID, requestID, accountKey,\
data, isLast, error):
    if error.isSuccess():
        print u'[onBatchReqCreditAccountDetailWithAccKey] success, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onBatchReqCreditAccountDetailWithAccKey] failed, accountID:',\
        accountID, ', errorMsg:',error.errorMsg()
```

4.1.3. 委托查询的批量回调

```
onBatchReqOrderDetail(self,accountID, requestID, data, isLast, error)
```

- 释义调用reqOrderDetail查询账号的当日委托后的回调接口
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	委托明细对象	存放COrderDetail的一个list
isLast	是否为最后一次返回，返回最后一条委托时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#委托明细查询的回调函数
def onBatchReqOrderDetail(self,accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onBatchReqOrderDetail] sucess, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onBatchReqOrderDetail] failed, errorMsg:', \
        error.errorMsg()
```

```
onBatchReqOrderDetailWithAccKey(self,accountID, requestID, accountKey,\
 data, isLast, error)
```

- 释义调用reqOrderDetail查询账号的当日委托后的回调接口
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str
data	委托明细对象	存放COrderDetail的一个list
isLast	是否为最后一次返回，返回最后一条委托时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#委托明细查询的回调函数
def onBatchReqOrderDetailWithAccKey(self,accountID, requestID, accountKey,\
    data, isLast, error):
    if error.isSuccess():
        print u'[onBatchReqOrderDetailWithAccKey] sucess, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onBatchReqOrderDetailWithAccKey] failed, errorMsg:', \
        error.errorMsg()
```

4.1.4. 成交查询的批量回调

```
onBatchReqDealDetail(self,accountID, requestID, data, isLast, error)
```

- 释义调用reqDealDetail查询账号的当日成交后的回调接口
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	成交明细对象	存放CDealDetail 的一个list
isLast	是否为最后一次返回，返回最后一条成交明细时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#成交明细查询的回调函数
def onBatchReqDealDetail(self, accountID, requestID, data, isLast, error):
    if error.isSuccess():
        print u'[onBatchReqDealDetail] sucess, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onBatchReqDealDetail] failed, accountID:', \
        accountID, ', errorMsg:',error.errorMsg()
```

```
onBatchReqDealDetailWithAccKey(self,accountID, requestID, accountKey, \
data, isLast, error)
```

- 释义调用reqDealDetail查询账号的当日成交后的回调接口
- 参数

参数	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
accountKey	账号key	str
data	成交明细对象	存放CDealDetail 的一个list
isLast	是否为最后一次返回，返回最后一条成交明细时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#成交明细查询的回调函数
def onBatchReqDealDetailWithAccKey(self, accountID, requestID, accountKey,\
data, isLast, error):
    if error.isSuccess():
        print u'[onBatchReqDealDetailWithAccKey] sucess, accountID:', \
        accountID, ', isLast:', isLast
    else:
        print u'[onBatchReqDealDetailWithAccKey] failed, accountID:', \
        accountID, ', errorMsg:',error.errorMsg()
```

4.1.5. 持仓明细查询的回调

```
onBatchReqPositionDetail(self, accountID, requestID, data, isLast, error)
```

- 释义调用reqPositionDetail查询账号的当日持仓明细后的回调接口
- 参数

参数名	参数释义	类型
accountID	参数释义资金账号	str
requestID	参数释义请求ID，每个请求的ID自增	int
data	参数释义持仓明细对象	存放CPositionDetail的一个list
isLast	是否为最后一次返回，返回最后一条持仓明细时这个值是True，其它情况是False	bool
error	参数释义错误消息对象	XtError

- 示例

```
#请求持仓明细的回调函数
def onBatchReqPositionDetail(self, accountID, requestID, data, isLast, error):
if error.isSuccess():
    print u'[onBatchReqPositionDetail] sucess, accountID:', \
    accountID, ', isLast:', isLast
else:
    print u'[onBatchReqPositionDetail] failed, accountID:', \
    accountID, ', errorMsg:',error.errorMsg()
```

4.1.6. 持仓统计查询的回调

```
onBatchReqPositionStatics(self, accountID, requestID, data, isLast, error)
```

- 释义调用reqPositionStatics查询账号持仓统计后的回调接口
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	持仓统计对象	存放CPositionStatics 的一个list
isLast	是否为最后一条，返回最后一条持仓统计时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#请求持仓统计的回调函数
def onBatchReqPositionStatics(self,accountID, requestID, data, isLast,error):
if error.isSuccess():
    print u'[onBatchReqPositionStatics] sucess, accountID:', \
    accountID, ', isLast:', isLast
else:
    print u'[onBatchReqPositionStatics] failed, accountID:', \
    accountID, ', errorMsg:',error.errorMsg()
```

4.1.7. 求账号投资组合可用持仓统计的回调

```
onBatchReqRevolvePositions(self, accountID, requestID, data, isLast, error)
```

- 释义调用reqRevolvePositions查询账号投资组合可用持仓统计的回调接口
- 参数

参数名称	参数释义	参数类型
accountID	资金账号	str
requestID	请求ID，每个请求的ID自增	int
data	持仓统计对象	存放CPFPositionStatics 的一个list
isLast	是否为最后一条，返回最后一条时这个值是True，其它情况是False	bool
error	错误消息对象	XtError

- 示例

```
#请求持仓统计的回调函数
def onBatchReqRevolvePositions(self,accountID, requestID, data, isLast,error):
if error.isSuccess():
    print u'[onBatchReqRevolvePositions] sucess, accountID:', \
    accountID, ', isLast:', isLast
else:
    print u'[onBatchReqRevolvePositions] failed, accountID:', \
    accountID, ', errorMsg:',error.errorMsg()
```

4.1.8. 请求信用账号标的信息的回调

该方法是请求信用账号标的信息的回调，方法如下：

```
onBatchReqStksubjects(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	两融账号标的数据	存放 CStkSubjects 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求两融账号标的的回调函数
def onBatchReqStksubjects(self, accountID, requestID, data, isLast, error):
"""
请求两融账号标的的回调函数。
"""
if error.isSuccess():
    print(u'[onBatchReqStksubjects] sucess, accountID:', \
    accountID, ', isLast:', isLast)
else:
    print(u'[onBatchReqStksubjects] failed, accountID:', \
    accountID, ', errorMsg:', error.errorMsg())
```

4.1.9. 请求两融账号负债的回调函数

该方法是请求两融账号负债的回调，方法如下：

```
onBatchReqStkcompacts(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int

参数名称	参数释义	类型
data	两融账号负债数据	存放 CStkCompacts 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求两融账号负债的回调函数
def onBatchReqStkcompacts(self, accountID, requestID, data, isLast, error):
    """
    请求两融账号负债的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqStkcompacts] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqStkcompacts] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.10. 请求期权账号备兑持仓的回调函数

该方法是请求期权账号备兑持仓的回调，方法如下：

```
onBatchReqCoveredStockPosition(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	期权账号备兑持仓数据	存放 CCoveredStockPosition 的一个list

参数名称	参数释义	类型
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求期权账号备兑持仓的回调函数
def onBatchReqCoveredStockPosition(self, accountID, requestID, data, isLast, error):
    """
    请求期权账号备兑持仓的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqCoveredStockPosition] sucess, \
        accountID:', accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqCoveredStockPosition] failed, \
        accountID:', accountID, ', errorMsg:', error.errorMsg())
```

4.1.11. 请求产品信息的回调函数

该方法是请求产品信息的回调，方法如下：

```
onBatchReqProductData(self, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
requestID	请求ID	int
data	产品信息数据	存放 CProductData 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求产品信息的回调函数
def onBatchReqProductData(self, requestID, data, isLast, error):
    """
    请求产品信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqProductData] sucess, isLast:', isLast)
    else:
        print(u'[onBatchReqProductData] failed, errorMsg:', error.errorMsg())
```

4.1.12. 请求合约数据的回调函数

该方法是请求合约数据的回调，方法如下：

```
onBatchReqCInstrumentDetail(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	合约数据	存放CInstrumentDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求合约数据的回调函数
def onBatchReqCInstrumentDetail(self, accountID, requestID, data, isLast, error):
    """
    请求合约数据的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqCInstrumentDetail] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqCInstrumentDetail] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.13. 请求期权账号组合持仓数据的回调函数

该方法是请求期权账号组合持仓数据的回调，方法如下：

```
onBatchReqStkOptCombPositionDetail(self, accountID, requestID, data, isLast, error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	期权账号组合持仓数据	存放CStockOptionCombPositionDetail的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求期权账号组合持仓数据的回调函数
def onBatchReqStkOptCombPositionDetail(self, accountID, requestID, data, isLast, error):
    """
    请求期权账号组合持仓数据的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqStkOptCombPositionDetail] sucess, accountID:',\
         accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqStkOptCombPositionDetail] failed, accountID:',\
         accountID, ', errorMsg:', error.errorMsg())
```

4.1.14. 请求港股账号汇率数据的回调函数

该方法是请求港股账号汇率数据的回调，方法如下：

```
onBatchReqReferenceRate(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	港股账号汇率数据	存放CReferenceRate的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求港股账号汇率数据的回调函数
def onBatchReqReferenceRate(self, accountID, requestID, data, isLast, error):
    """
    请求港股账号汇率数据的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqReferenceRate] sucess, accountID:', accountID, \
        ', isLast:', isLast)
    else:
        print(u'[onBatchReqReferenceRate] failed, accountID:', accountID, \
        ', errorMsg:', error.errorMsg())
```

4.1.15. 请求两融账号两融综合资金数据的回调函数

该方法是请求两融综合资金数据的回调，方法如下：

```
onBatchReqCreditDetail(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	两融综合资金数据	存放 CCreditDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求两融账号两融综合资金数据的回调函数
def onBatchReqCreditDetail(self, accountID, requestID, data, isLast, error):
    """
    请求两融账号两融综合资金数据的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqCreditDetail] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqCreditDetail] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.16. 请求新股额度数据的回调函数

该方法是请求新股额度数据的回调，方法如下：

```
onBatchReqSubscribeInfo(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	新股额度数据	存放 CSubscribeInfo 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求新股额度数据的回调函数
def onBatchReqSubscribeInfo(self, accountID, requestID, data, isLast, error):
    """
    请求新股额度数据的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqSubscribeInfo] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqSubscribeInfo] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.17. 请求未了结负债信息的回调函数

该方法是请求两融未了结负债信息数据的回调，方法如下：

```
onBatchReqStkUnCloseCompact(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	未了结负债信息数据	存放CStkUnClosedCompacts的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求未了结负债信息的回调函数
def onBatchReqStkUnCloseCompact(self, accountID, requestID, data, isLast, error):
    """
    请求未了结负债信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqStkUnCloseCompact] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqStkUnCloseCompact] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.18. 请求已了结负债信息的回调函数

该方法是请求两融已了结负债信息数据的回调，方法如下：

```
onBatchReqStkClosedCompact(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	已了结负债信息	存放CStkClosedCompacts的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求已了结负债信息的回调函数
def onBatchReqStkClosedCompact(self, accountID, requestID, data, isLast, error):
    """
    请求已了结负债信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqStkClosedCompact] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqStkClosedCompact] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.19. 获取用户下所有账号key的回调函数

该方法是获取用户下所有账号key数据的回调，方法如下：

```
onBatchReqAccountKey(self, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
requestID	请求ID	int
data	accountKey数据	存放 CAccountKey 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 获取用户下所有账号key的回调函数
def onBatchReqAccountKey(self, requestID, data, isLast, error):
    """
    获取用户下所有账号key的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqAccountKey] sucess, isLast:', isLast)
    else:
        print(u'[onBatchReqAccountKey] failed, errorMsg:', error.errorMsg())
```

4.1.20. 根据委托号请求账号成交明细信息的回调函数

该方法是根据委托号请求账号成交明细信息的回调，方法如下：

```
onBatchReqDealDetailBySysID(self, accountID, requestID, orderSyeId, \
exchangeId, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
orderSyeId	委托号	str
exchangeId	委托所属市场	str
data	账号成交明细数据	存放 CDealDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 根据委托号请求账号成交明细信息的回调函数
def onBatchReqDealDetailBySysID(self, accountID, requestID, orderSyeId,\
 exchangeId, data, isLast, error):
    """
    根据委托号请求账号成交明细信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqDealDetailBySysID] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqDealDetailBySysID] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.21. 请求账号结算单信息的回调函数

该方法是请求账号结算单信息数据的回调，方法如下：

```
onBatchReqDeliveryDetail(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	账号结算单数据	存放CDeliveryDetail的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求账号结算单信息的回调函数
def onBatchReqDeliveryDetail(self, accountID, requestID, data, isLast, error):
    """
    请求账号结算单信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqDeliveryDetail] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqDeliveryDetail] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.22. 请求两融账号融券可融数量的回调函数

该方法是请求融券可融数量数据的回调，方法如下：

```
onBatchReqCreditSloCode(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	融券可融数量数据	存放CCreditSloCode 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求两融账号融券可融数量的回调函数
def onBatchReqCreditSloCode(self, accountID, requestID, data, isLast, error):
    """
    请求两融账号融券可融数量的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqCreditSloCode] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqCreditSloCode] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.23. 请求两融账号融资融券标的的回调函数

该方法是请求融资融券标的数据的回调，方法如下：

```
onBatchReqCreditSubjects(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	融资融券标的数据,	存放CCreditSubjects 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求两融账号融资融券标的的回调函数
def onBatchReqCreditSubjects(self, accountID, requestID, data, isLast, error):
    """
    请求两融账号融资融券标的的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqCreditSubjects] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqCreditSubjects] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.24. 请求两融账号担保标的的回调函数

该方法是请求担保标的数据的回调，方法如下：

```
onBatchReqCreditAssure(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	担保标的数据,	存放 CCreditAssure 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求两融账号担保标的的回调函数
def onBatchReqCreditAssure(self, accountID, requestID, data, isLast, error):
    """
    请求两融账号担保标的的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqCreditAssure] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqCreditAssure] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.25. 请求账号银证转账银行信息的回调函数

该方法是请求账号银证转账银行信息数据的回调，方法如下：

```
onBatchReqTransferBank(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	账号银证转账银行信息数据,	存放CQueryBankInfo 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求账号银证转账银行信息的回调函数
def onBatchReqTransferBank(self, accountID, requestID, data, isLast, error):
    """
    请求账号银证转账银行信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqTransferBank] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqTransferBank] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.26. 请求账号银证转账银行流水的回调函数

该方法是请求账号银证转账银行流水数据的回调，方法如下：

```
onBatchReqTransferSerial(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	账号银证转账银行流水数据,	存放 CTransferSerial 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求账号银证转账银行流水的回调函数
def onBatchReqTransferSerial(self, accountID, requestID, data, isLast, error):
    """
    请求账号银证转账银行流水的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqTransferSerial] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqTransferSerial] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.27. 按市场请求合约信息的回调函数

该方法是获取合约信息数据的回调，方法如下：

```
onBatchReqInstrumentInfoByMarket(self, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
requestID	请求ID	int
data	合约信息数据	存放CInstrumentInfo 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 按市场请求合约信息的回调函数
def onBatchReqInstrumentInfoByMarket(self, requestID, data, isLast, error):
    """
    按市场请求合约信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqInstrumentInfoByMarket] sucess, isLast:', isLast)
    else:
        print(u'[onBatchReqInstrumentInfoByMarket] failed, errorMsg:', \
        error.errorMsg())
```

```
onBatchReqInstrumentInfoByMarketWithMkt(self, requestID, exchangeId, data, isLast,error):
```

- 参数:

参数名称	参数释义	类型
requestID	请求ID	int
exchangeId	市场ID	str
data	合约信息数据	存放CInstrumentInfo 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例:

```
# 按市场请求合约信息的回调函数
def onBatchReqInstrumentInfoByMarketWithMkt(self, requestID, exchangeId, data,\
 isLast, error):
    """
    按市场请求合约信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqInstrumentInfoByMarketWithMkt] sucess, isLast:', isLast)
    else:
        print(u'[onBatchReqInstrumentInfoByMarketWithMkt] failed, errorMsg:',\
         error.errorMsg())
```

4.1.28. 请求账号可撤单委托明细的回调函数

该方法是请求账号可撤单委托明细数据的回调，方法如下：

```
onBatchReqCanCancelOrderDetail(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	账号委托明细数据,	存放 COrderDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求账号可撤单委托明细的回调函数
def onBatchReqCanCancelOrderDetail(self, accountID, requestID, data, isLast, error):
    """
    请求账号可撤单委托明细的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqCanCancelOrderDetail] sucess, accountID:',\
         accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqCanCancelOrderDetail] failed, accountID:',\
         accountID, ', errorMsg:', error.errorMsg())
```

4.1.29. 请求所有下单信息的回调函数

该方法是获取所有下单信息数据的回调，方法如下：

```
onBatchReqCommandsInfo(self, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
requestID	请求ID	int
data	下单信息数据	存放 COrderInfo 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求所有下单信息的回调函数
def onBatchReqCommandsInfo(self, requestID, data, isLast, error):
    """
    请求所有下单信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqCommandsInfo] sucess, isLast:', isLast)
    else:
        print(u'[onBatchReqCommandsInfo] failed, errorMsg:', error.errorMsg())
```

4.1.30. 请求账号账号普通柜台资金的回调函数

该方法是请求账号资金划拨普通柜台资金数据的回调，方法如下：

```
onBatchReqComFund(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	账号资金划拨普通柜台资金数据,	存放CStockComFund 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求账号账号普通柜台资金的回调函数
def onBatchReqComFund(self, accountID, requestID, data, isLast, error):
    """
    请求账号账号普通柜台资金的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqComFund] sucess, accountID:', accountID,\
         ', isLast:', isLast)
    else:
        print(u'[onBatchReqComFund] failed, accountID:', accountID,\
         ', errorMsg:', error.errorMsg())
```

4.1.31. 请求账号普通柜台持仓的回调函数

该方法是请求账号股份划拨普通柜台持仓数据的回调，方法如下：

```
onBatchReqComPosition(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	账号股份划拨普通柜台持仓数据,	存放CStockComPosition的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求账号普通柜台持仓的回调函数
def onBatchReqComPosition(self, accountID, requestID, data, isLast, error):
    """
    请求账号普通柜台持仓的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqComPosition] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqComPosition] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.32. 请求账号历史委托明细的回调函数

该方法是请求账号历史委托明细数据的回调，方法如下：

```
onBatchReqHistoryOrderDetail(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	账号历史委托明细,	存放 COrderDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求账号历史委托明细的回调函数
def onBatchReqHistoryOrderDetail(self, accountID, requestID, data, isLast, error):
    """
    请求账号历史委托明细的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqHistoryOrderDetail] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqHistoryOrderDetail] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.33. 请求账号历史成交明细的回调函数

该方法是请求账号历史成交明细数据的回调，方法如下：

```
onBatchReqHistoryDealDetail(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	账号历史成交明细,	存放 CDealDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求账号历史成交明细的回调函数
def onBatchReqHistoryDealDetail(self, accountID, requestID, data, isLast, error):
    """
    请求账号历史成交明细的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqHistoryDealDetail] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqHistoryDealDetail] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.34. 请求账号历史持仓统计的回调函数

该方法是请求账号历史持仓统计数据的回调，方法如下：

```
onBatchReqHistoryPositionStatics(self, accountID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	资金账号	str
requestID	请求ID	int
data	账号历史持仓统计，	存放CPositionStatics 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求账号历史成交明细的回调函数
def onBatchReqHistoryPositionStatics(self, accountID, requestID, data, isLast, error):
    """
    请求账号历史成交明细的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqHistoryPositionStatics] sucess, accountID:', \
        accountID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqHistoryPositionStatics] failed, accountID:', \
        accountID, ', errorMsg:', error.errorMsg())
```

4.1.35. 查询产品Id下所有的投资组合的回调函数

该方法是查询产品Id下所有的投资组合数据的回调，方法如下：

```
onBatchReqProductPortfolio(self, nProductID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
nProductID	产品ID	str
requestID	请求ID	int
data	账号历史持仓统计，	存放 CPortfolioInfo 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 查询产品Id下所有的投资组合的回调函数
def onBatchReqProductPortfolio(self, nProductID, requestID, data, isLast, error):
    """
    查询产品Id下所有的投资组合的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqProductPortfolio] sucess, nProductID:', \
        nProductID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqProductPortfolio] failed, nProductID:', \
        nProductID, ', errorMsg:', error.errorMsg())
```

4.1.36. 请求投资组合委托信息的回调函数

该方法是查询投资组合委托信息的回调，方法如下：

```
onBatchReqPortfolioOrder(self, nPortfolioID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
nPortfolioID	产品ID	str
requestID	投资组合ID	int
data	账号委托明细数据,,	存放 COrderDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求投资组合委托信息的回调函数
def onBatchReqPortfolioOrder(self, nPortfolioID, requestID, data, isLast, error):
    """
    请求投资组合委托信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqPortfolioOrder] sucess, nPortfolioID:',\
         nPortfolioID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqPortfolioOrder] failed, nPortfolioID:', \
        nPortfolioID, ', errorMsg:', error.errorMsg())
```

4.1.37. 请求投资组合一段时间内的委托信息的回调函数

该方法是查询投资组合一段时间内的委托信息的回调，方法如下：

```
onBatchReqPortfolioMultiOrder(self, nPortfolioID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
nPortfolioID	产品ID	str
requestID	投资组合ID	int
data	账号委托明细数据,,	存放 COrderDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求投资组合一段时间内的委托信息的回调函数
def onBatchReqPortfolioMultiOrder(self, nPortfolioID, requestID, data, isLast, error):
    """
    请求投资组合一段时间内的委托信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqPortfolioMultiOrder] sucess, nPortfolioID:', \
        nPortfolioID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqPortfolioMultiOrder] failed, nPortfolioID:', \
        nPortfolioID, ', errorMsg:', error.errorMsg())
```

4.1.38. 请求投资组合成交明细的回调函数

该方法是查询投资组合成交信息的回调，方法如下：

```
onBatchReqPortfolioDeal(self, nPortfolioID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
nPortfolioID	产品ID	str
requestID	投资组合ID	int
data	投资组合成交明细数据,,	存放 CDealDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求投资组合成交明细的回调函数
def onBatchReqPortfolioDeal(self, nPortfolioID, requestID, data, isLast, error):
    """
    请求投资组合成交明细的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqPortfolioDeal] sucess, nPortfolioID:', \
        nPortfolioID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqPortfolioDeal] failed, nPortfolioID:', \
        nPortfolioID, ', errorMsg:', error.errorMsg())
```

4.1.39. 请求投资组合一段时间内的成交信息的回调函数

该方法是查询投资组合一段时间内的成交信息的回调，方法如下：

```
onBatchReqPortfolioMultiDeal(self, nPortfolioID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
nPortfolioID	产品ID	str
requestID	投资组合ID	int
data	投资组合成交明细数据,,	存放 CDealDetail 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求投资组合一段时间内的成交信息的回调函数
def onBatchReqPortfolioMultiDeal(self, nPortfolioID, requestID, data, isLast, error):
    """
    请求投资组合一段时间内的成交信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqPortfolioMultiDeal] sucess, nPortfolioID:', \
        nPortfolioID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqPortfolioMultiDeal] failed, nPortfolioID:', \
        nPortfolioID, ', errorMsg:', error.errorMsg())
```

4.1.40. 请求投资组合持仓信息的回调函数

该方法是查询投资组合持仓信息的回调，方法如下：

```
onBatchReqPortfolioPosition(self, nPortfolioID, requestID, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
nPortfolioID	产品ID	str
requestID	投资组合ID	int
data	账号持仓统计数据,,	存放CPositionStatics 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求投资组合持仓信息的回调函数
def onBatchReqPortfolioPosition(self, nPortfolioID, requestID, data, isLast, error):
    """
    请求投资组合持仓信息的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqPortfolioPosition] sucess, nPortfolioID:', \
        nPortfolioID, ', isLast:', isLast)
    else:
        print(u'[onBatchReqPortfolioPosition] failed, nPortfolioID:', \
        nPortfolioID, ', errorMsg:', error.errorMsg())
```

4.1.41. 请求收益互换账号框架号的回调函数

该方法是查询投资组合持仓信息的回调，方法如下：

```
onBatchReqStrategyInfo(self, accountID, requestID, accountKey, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	产品ID	str
requestID	投资组合ID	int
accountKey	账号key	str
data	收益互换账号框架号数据	存放 CStrategyInfo 的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求收益互换账号框架号的回调函数
def onBatchReqStrategyInfo(self, accountID, requestID, accountKey, data, isLast, error):
    """
    请求收益互换账号框架号的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqStrategyInfo] sucess, accountID:', accountID, \
        ', isLast:', isLast)
    else:
        print(u'[onBatchReqStrategyInfo] failed, accountID:', accountID, \
        ', errorMsg:', error.errorMsg())
```

4.1.42. 请求股东号的回调函数

该方法是请求股东号的回调，方法如下：

```
onBatchReqSecuAccount(self, accountID, requestID, accountKey, data, isLast,error):
```

- 参数：

参数名称	参数释义	类型
accountID	产品ID	str
requestID	投资组合ID	int
accountKey	账号key	str
data	股东号数据	存放CSecuAccount的一个list
isLast	是否最后一次响应，返回最后一条时这个值是True，其它情况是False	bool
error	异常信息	XtError

- 示例：

```
# 请求股东号的回调函数
def onBatchReqSecuAccount(self, accountID, requestID, accountKey, data, isLast, error):
    """
    请求股东号的回调函数。
    """
    if error.isSuccess():
        print(u'[onBatchReqSecuAccount] sucess, accountID:', accountID,\
         ', isLast:', isLast)
    else:
        print(u'[onBatchReqSecuAccount] failed, accountID:', accountID, \
        ', errorMsg:', error.errorMsg())
```