XTP各种业务报单

1.普通股票交易下单示例

```
/*XTP::API::TraderApi* pUserApi ;//交易api的创建示例在此忽略
uint64_t session_id_ ;//登录成功后得到的会话
*/
void ordinaryInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "000001";//需要交易的股票代码，用户自行更改
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type = XTP_BUSINESS_TYPE_CASH;//XTP_BUSINESS_TYPE_CASH为
普通股票下单业务类型
    query_param.side = XTP_SIDE_BUY;//买，根据买卖方向进行修改
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型，可修改为其他类型，具体参考头
文件注释
    query_param.price = 12.51;//价格，可参考行情修改
    query_param.quantity = 200;//买卖股票的委托数量,需参考交易规则
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//=0说明下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }

}
```

2.新股申购交易下单代码示例

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
*/

void iposInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
```

```
    std::string ticker = "303001";//新股申购的新股代码，可参考QueryIPOInfoList()查询结果
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:存放证券代码的字符串长度
    query_param.business_type = XTP_BUSINESS_TYPE_IPOS;//XTP_BUSINESS_TYPE_IPOS为新股申购的业务类型
    query_param.side = XTP_SIDE_BUY;//买
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型，仅能使用限价
    query_param.quantity = 500;//申购数量，用户可根据实际可申购数量来填写，可参考QueryIPOQuotaInfo()查询结果，对于申购单位：深证500股，上证1000股，科创板500，委托数量必须是这些单位的整数倍
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单接口
        if(xtp_id == 0)//=0说明下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }

}
```

3.配股或者配债下单代码示例

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
*/

void allotmentInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "380033";//配股或者配债代码，可参考QueryPosition()查询结果
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:存放证券代码的字符串长度
    query_param.business_type = XTP_BUSINESS_TYPE_ALLOTMENT;//XTP_BUSINESS_TYPE_ALLOTMENT为配股业务类型
    query_param.side = XTP_SIDE_BUY;//买卖类型必须是买
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型，仅能使用限价
    query_param.quantity = 80;//配售股票或者债券的委托数量，单位是股或者张，用户可根据QueryPosition()接口查询的结果来填写
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
```

```
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

4.国债逆回购下单代码示例

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
*/

void repoInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "131810";//需要交易的股票代码，用户自行更改
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type = XTP_BUSINESS_TYPE_REPO;//XTP_BUSINESS_TYPE_REPO为
国债逆回购业务类型
    query_param.side = XTP_SIDE_SELL;//卖
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型，仅能使用限价
    query_param.price = 1.995;//价格，可参考行情修改
    query_param.quantity = 50;//买股票的委托数量，单位为张，需参考交易所规则，上交所
1000张的整数倍，深交所10张的整数倍，最大均不超过100w张
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

5.ETF申赎下单代码示例

(1) 申购

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
```

```
*/

void ETFInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SH_A;//市场类型：上证(跟证券代码所属的交易所相一致)
    std::string ticker = "515880";//ETF买卖代码，XTP内统一使用买卖代码，不使用申赎代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type = XTP_BUSINESS_TYPE_ETF;//XTP_BUSINESS_TYPE_ETF为ETF
申赎业务类型
    query_param.side = XTP_SIDE_PURCHASE;//申购
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价
    query_param.quantity = 1000000;//委托数量为最小申赎单位的倍数，可以参考QueryETF()
的查询结果填写
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(2) 赎回

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
*/

void ETFInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SH_A;//市场类型：上证(跟证券代码所属的交易所相一致)
    std::string ticker = "515880";//ETF买卖代码，XTP内统一使用买卖代码，不使用申赎代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type = XTP_BUSINESS_TYPE_ETF;//XTP_BUSINESS_TYPE_ETF为ETF
申赎业务类型
    query_param.side = XTP_SIDE_REDEMPTION;//赎回
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价
```

```
    query_param.quantity = 1000000;//委托数量为最小申赎单位的倍数，可以参考QueryETF()
的查询结果填写
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

6.信用业务交易示例代码

(1) 融资买入

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
*/
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "000001";//需要交易的股票代码，用户自行更改
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_MARGIN_TRADE;//融资买入
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价，可修改为其他类型，具体
参考头文件注释
    query_param.price = 19.05;//价格，可参考行情修改
    query_param.quantity = 200;//买卖股票的委托数量，具体参考交易所规则，最大数量可参考
QueryPosition()查询结果
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(2) 融券卖出

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "000001";//需要交易的股票代码，用户自行更改，可参考
QueryCreditTickerAssignInfo()查询结果
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_SHORT_SELL;//融券卖出
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价
    query_param.price = 19.05;//价格不得低于最新价
    query_param.quantity = 200;//买卖股票的委托数量，请参照交易规则，不可为零散股，可参
考QueryCreditTickerAssignInfo()查询结果
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(3) 卖券还款

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "000001";//需要交易的股票代码，用户自行更改
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_REPAY_MARGIN;//卖券还款
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型：限价，可修改为其他类型，具体
参考头文件注释
```

```
    query_param.price = 19.05;//价格，可参考行情修改
    query_param.quantity = 200;//委托数量，具体参考交易所规则，最大数量可参考
QueryPosition()查询结果
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(4) 买券还券

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "000001";
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_REPAY_STOCK;//买券还券
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价，可修改为其他类型，具体
参考头文件注释
    query_param.price = 19.05;//价格，可参考行情修改
    query_param.quantity = 200;//委托数量，具体参考交易所规则
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(5) 现券还券

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "000001";//需要还券的股票代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_STOCK_REPAY_STOCK;//现券还券
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价
    query_param.quantity = 100;//未还券数量(数量小于可还券数量)，可参考
QueryCreditTickerDebtInfo()查询结果
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(6) 余券划转

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "000001";//需要划转的余券代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_SURSTK_TRANS;//余券划转
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价
    query_param.quantity = 99;//余券数量，可参考QueryMulCreditExcessStock()查询结果
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
```

```
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(7) 担保品转入

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "300860";//需要转入的股票代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_GRTSTK_TRANSIN;//担保品转入
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价
    query_param.quantity = 20;//数量，需用户自行查询现货账户内的持仓数量
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(8) 担保品转出

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "300860";//持仓里的证券代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
```

```
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_GRTSTK_TRANSOUT;//担保品转出
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价
    query_param.quantity = 20;//数量(不可大于持仓可卖数量)，可参考QueryPosition()查询
结果填写
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(9) 担保品买

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "159919";//股票代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_BUY;//买
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价，可修改为其他类型，具体
参考头文件注释
    query_param.price = 5.721;//价格，可参考行情修改
    query_param.quantity = 200;//委托数量，具体参考交易所规则
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(10) 担保品卖

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "159919";//持仓里的证券代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_SELL;//卖
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价，可修改为其他类型，具体
参考头文件注释
    query_param.price = 5.721;//价格，可参考行情修改
    query_param.quantity = 200;//委托数量，具体参考交易所规则，不可超过持仓的可卖数量，
可参考QueryPosition()查询结果填写
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(11) 指定卖券还息

```
void marginInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "159919";//持仓里的证券代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_MARGIN;//XTP_BUSINESS_TYPE_MARGIN为融资融券业务类型
    query_param.side = XTP_SIDE_REPAY_MARGIN;//卖券还款
    query_param.position_effect = XTP_POSITION_EFFECT_INIT;//position_effect开平标
志，除期权以外，都使用该值
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价
    query_param.price = 5.721;//价格，可参考行情修改
    query_param.quantity = 200;//数量，不可超过持仓的可卖数量，可参考QueryPosition()
查询结果填写
```

```
    std::string str_debt_id = "202111020012000000015";//负债信息的合约编码，可自行修
改
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi-
>CreditSellStockRepayDebtInterestFee(&query_param, stdstr_debt_id.c_str(),
session_id_);//下单接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(12) 现金还款

```
void readyCreditCashRepay(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    double amount = 60000.0;//申请还款金额，待还资金可参考QueryCreditAssetDebtInfo()
查询结果
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->CreditCashRepay(amount, session_id_);//现金直
接还款请求
        if(xtp_id == 0)
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }

}
```

(13) 指定现金还息

```
void readyCreditCashRepayDebtInterestFee(XTP::API::TraderApi* pUserApi, uint64_t
session_id_)
{
    std::string debt_id = "20190311000510000017 5";//需要还息的负债合约编号
    double amount = 60000.0;//现金还息的金额，可参考QueryCreditDebtInfo()查询结果
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi-
>CreditCashRepayDebtInterestFee(debt_id.c_str(), amount, session_id_);//现金还指定
负债合约息费请求
        if(xtp_id == 0)
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
```

```
    }
}
```

7. 期权业务交易相关示例代码

(1) 期权业务

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
*/
void optInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "91003521";//合约代码
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_OPTION;//XTP_BUSINESS_TYPE_OPTION为期权业务类型
    query_param.side = XTP_SIDE_BUY;//买
    query_param.position_effect = XTP_POSITION_EFFECT_OPEN;//开
    query_param.price_type = XTP_PRICE_LIMIT;//价格类型为限价，可修改为其他类型，具体
参考头文件注释
    query_param.price = 2.5093;//价格，可参考行情修改
    query_param.quantity = 3;//数量，具体参考交易所规则
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(2) 行权

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
*/
void optInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOrderInsertInfo query_param;//XTPOrderInsertInfo 为下单参数结构体
    memset(&query_param, 0, sizeof(query_param));//结构体初始化
    //参数赋值
```

```
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟证券代码所属的交易所相一致)
    std::string ticker = "91003521";//当日日期与该合约行权日一样，才能做行权
    strcpy_s(query_param.ticker, XTP_TICKER_LEN, ticker.c_str());//XTP_TICKER_LEN:
存放证券代码的字符串长度
    query_param.business_type =
XTP_BUSINESS_TYPE_EXECUTE;//XTP_BUSINESS_TYPE_EXECUTE为行权业务类型
    query_param.quantity = 3;//数量,用户可自行修改
    if(pUserApi && session_id_ !=0 )
    {
        uint64_t xtp_id = pUserApi->InsertOrder(&query_param, session_id_);//下单
接口
        if(xtp_id == 0)//下单发送失败
        {
            XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
        }
    }
}
```

(3) 组合策略的合并

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
*/
void optInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOptCombOrderInsertInfo query_param;//XTPOptCombOrderInsertInfo 为组合策略下
单的结构体
    memset(&query_param, 0, sizeof(query_param));
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟腿的合约代码所属的交易所相一
致)
    query_param.quantity = 3;//合并的合约数目，用户自行修改
    query_param.business_type = XTP_BUSINESS_TYPE_OPTION_COMBINE;//组合策略业务类型
    query_param.side = XTP_SIDE_OPT_COMBINE;//合并
    std::string std_str_strategy_id = "CNSJC";//策略代码类型
    strcpy_s(query_param.opt_comb_info.strategy_id, XTP_STRATEGY_ID_LEN,
std_str_strategy_id.c_str());//XTP_STRATEGY_ID_LEN策略代码的存放长度
    // 组合策略腿合约信息结构体，目前组合策略的合并只支持两条腿的合约代码合并
    query_param.opt_comb_info.num_legs = 2;//两个腿
    std::string std_str_ticker1 = "91003521";//腿1的合约代码，用户自行修改
    std::string std_str_ticker2 = "91003536";//腿2的合约代码，用户自行修改
    strcpy_s(query_param.opt_comb_info.leg_detail[0].leg_security_id,
XTP_TICKER_LEN, std_str_ticker1.c_str());//XTP_TICKER_LEN:存放合约代码的字符串长度
    strcpy_s(query_param.opt_comb_info.leg_detail[1].leg_security_id,
XTP_TICKER_LEN, std_str_ticker2.c_str());
    uint64_t xtp_id = pUserApi->InsertOptionCombinedOrder(&query_param,
session_id_);//组合策略下单接口
    if(xtp_id == 0)//下单发送失败
    {
        XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
    }
}
```

(4) 组合策略的拆分

```
/*XTP::API::TraderApi* pUserApi //交易api的创建示例在此忽略
uint64_t session_id_ //登录成功后得到的会话
*/
void optInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOptCombOrderInsertInfo query_param;//XTPOptCombOrderInsertInfo 为组合策略下单的结构体
    memset(&query_param, 0, sizeof(query_param));
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟腿的合约代码所属的交易所相一致)
    query_param.quantity = 3;//要拆分的合约数量，可自行修改
    query_param.business_type = XTP_BUSINESS_TYPE_OPTION_COMBINE;//组合策略业务类型
    query_param.side = XTP_SIDE_OPT_SPLIT;//拆分
    std::string std_str_comb_num = "00B0R4JJOA000026";//要拆分的组合编码
    strcpy_s(query_param.opt_comb_info.comb_num, XTP_SECONDARY_ORDER_ID_LEN, std_str_comb_num.c_str());//XTP_SECONDARY_ORDER_ID_LEN组合编码的字符串长度
    std::string std_str_strategy_id = "CNSJC";//该组合的策略代码类型
    strcpy_s(query_param.opt_comb_info.strategy_id, XTP_STRATEGY_ID_LEN, std_str_strategy_id.c_str());//XTP_STRATEGY_ID_LEN策略代码的存放长度
    uint64_t xtp_id = pUserApi->InsertOptionCombinedOrder(&query_param, session_id_);//组合策略下单接口
    if(xtp_id == 0)//下单发送失败
    {
        XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
    }
}
```

(5) 合并行权

```
void optInsertOrder(XTP::API::TraderApi* pUserApi, uint64_t session_id_)
{
    XTPOptCombOrderInsertInfo query_param;
    memset(&query_param, 0, sizeof(query_param));
    query_param.market = XTP_MKT_SZ_A;//市场类型：深证(跟腿的合约代码所属的交易所相一致)
    query_param.quantity = 3;//要合并行权的合约数量，可自行修改
    query_param.business_type = XTP_BUSINESS_TYPE_EXECUTE_COMBINE;//行权合并的业务类型
    std::string std_str_strategy_id = "exec";
    strcpy_s(query_param.opt_comb_info.strategy_id, XTP_STRATEGY_ID_LEN, std_str_strategy_id.c_str());//XTP_STRATEGY_ID_LEN策略代码的存放长度
    query_param.opt_comb_info.num_legs = 2;//腿数目为2(目前最多只支持两条腿)
    std::string ticker1 = "91003536";//腿1的合约代码，用户自行修改
    std::string ticker2 = "91003521";//腿2的合约代码，用户自行修改
    strcpy_s(query_param.opt_comb_info.leg_detail[0].leg_security_id, XTP_TICKER_LEN, ticker1.c_str());//XTP_TICKER_LEN:存放合约代码的字符串长度
```

```
    strcpy_s(query_param.opt_comb_info.leg_detail[1].leg_security_id,
XTP_TICKER_LEN, ticker2.c_str());
    uint64_t xtp_id = pUserApi->InsertOptionCombinedOrder(&query_param,
session_id_);//组合策略下单接口
    if(xtp_id == 0)//下单发送失败
    {
        XTPRI * error = pUserApi->GetApiLastError();//获取下单发送失败的错误码
    }
}
```