XTPTrade 新增算法接口使用说明

一、新增的接口

1. API 新增接口

与普通用户使用的交易 API 相比，新增了如下接口：

```
///用户登录algo服务器请求
///@return 表明此资金账号登录是否成功，非"0"表示登录失败，可以调用GetApiLastError()来获取错误代码，"0"表示登录成功
///@param ip algo服务器地址，类似"127.0.0.1"
///@param port algo服务器端口号
///@param user 登录用户名
///@param password 登录密码
///@param sock_type "1"代表TCP，"2"代表UDP，目前暂时只支持TCP
///@param local_ip 本地网卡地址，类似"127.0.0.1"
///@remark 此函数为同步阻塞式，不需要异步等待登录成功，当函数返回即可进行后续操作，此api只需调用一次，所有用户共用即可
virtual int LoginALGO(const char* ip, int port, const char* user, const char* password, XTP_PROTOCOL_TYPE sock_type, const char* local_ip

///algo业务中查询用户策略请求
///@return 请求发送是否成功，"0"表示成功，非"0"表示出错，此时用户可以调用GetApiLastError()来获取错误代码
///@param strategy_type 需要查询的策略类型，可填0
///@param client_strategy_id 需要查询的策略用户自定义id，可填0
///@param xtp_strategy_id 需要查询的策略在xtp系统中的id，如果指定，就一定按指定查询，如果填0，则按其他筛选条件查询
///@param session_id 资金账户对应的session_id,登录时得到
///@param request_id 用于用户定位查询响应的ID，由用户自定义
///@remark xtp_strategy_id条件的优先级最高，只有当xtp_strategy_id为0时，其他条件才生效，此条请求可能对应多条回应消息
virtual int QueryStrategy(uint32_t strategy_type, uint64_t client_strategy_id, uint64_t xtp_strategy_id, uint64_t session_id, int32_t req

///用户请求使用algo服务器建立算法通道
///@return 表明此资金账号建立算法通道请求消息发送是否成功，非"0"表示发送失败，可以调用GetApiLastError()来获取错误代码，"0"表示发送成功
///@param oms_ip oms服务器地址，类似"127.0.0.1"，非algo服务器地址
///@param oms_port oms服务器端口号，非algo服务器端口号
///@param user 登录用户名
///@param password 登录密码
///@param session_id 资金账户对应的session_id,登录时得到
///@remark 此函数为异步方式，一个用户只能拥有一个算法通道，如果之前已经建立，则无需重复建立，在使用算法前，请先建立算法通道
virtual int ALGOUserEstablishChannel(const char* oms_ip, int oms_port, const char* user, const char* password, uint64_t session_id) = 0;

///algo业务中用户报算法单请求
///@return 算法报单请求发送是否成功，"0"表示成功，非"0"表示出错，此时用户可以调用GetApiLastError()来获取错误代码
///@param strategy_type 需要创建的策略类型
///@param client_strategy_id 用户自定义id，帮助用户定位
///@param strategy_param 策略参数
///@param session_id 资金账户对应的session_id,登录时得到
///@remark 仅能在用户建立算法通道后使用，算法单的异步通知得
virtual int InsertAlgoOrder(uint32_t strategy_type, uint64_t client_strategy_id, char* strategy_param, uint64_t session_id) = 0;

///algo业务中用户撤销算法单请求
///@return 请求发送是否成功，"0"表示成功，非"0"表示出错，此时用户可以调用GetApiLastError()来获取错误代码
///@param cancel_flag 是否需要撤销的算法单已下的订单，true-撤单，false-不撤单
///@param xtp_strategy_id 需要撤销的算法单在xtp algobus系统中的id
///@param session_id 资金账户对应的session_id,登录时得到
///@remark 仅能在用户建立算法通道后调用
virtual int CancelAlgoOrder(bool cancel_flag, uint64_t xtp_strategy_id, uint64_t session_id) = 0;

///获取算法单的母单ID
///@return 返回算法单的母单ID，如果返回为0表示不是算法单
///@param order_xtp_id 算法单对应的xtp id
///@param order_client_id 算法单对应的自定义ID，不可随意填写
///@remark 返回为0表示，不是算法单，如果传入的参数不对的话，可能会得不到正确结果，此函数调用不依赖于是否登录
virtual uint64_t GetAlgorithmIDByOrder(uint64_t order_xtp_id, uint32_t order_client_id) = 0;
```

2. SPI 新增回调接口

与普通用户使用的交易 SPI 相比，新增了如下回调接口：

```
///algo业务中查询策略列表的响应
///@param strategy_info 策略具体信息
///@param strategy_param 此策略中包含的参数，如果error_info.error_id为0时，有意义
///@param error_info 查询查询策略列表发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
///@param request_id 此消息响应函数对应的请求ID
///@param is_last 此消息响应函数是否为request_id这条请求所对应的最后一个响应，当为最后一个的时候为true，如果为false，表示还有其他后续消息响
///@param session_id 资金账户对应的session_id，登录时得到
///@remark 需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线
virtual void OnQueryStrategy(XTPStrategyInfoStruct* strategy_info, char* strategy_param, XTPRI *error_info, int32_t request_id, bool is_la

///algo业务中策略运行时策略状态通知
///@param strategy_state 用户策略运行情况的状态通知
///@param session_id 资金账户对应的session_id，登录时得到
///@remark 需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线
virtual void OnStrategyStateReport(XTPStrategyStateReportStruct* strategy_state, uint64_t session_id) {};

///algo业务中用户建立算法通道的消息响应
///@param user 用户名
///@param error_info 建立算法通道发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误，即算法通道成功
///@param session_id 资金账户对应的session_id，登录时得到
///@remark 算法通道建立成功后，才能对用户创建策略等操作，一个用户只能拥有一个算法通道，如果之前已经建立，则无需重复建立
virtual void OnALGOUserEstablishChannel(char* user, XTPRI* error_info, uint64_t session_id) {};

///algo业务中报送策略单的响应
///@param strategy_info 用户报送的策略单的具体信息
///@param error_info 报送策略单发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
///@param session_id 资金账户对应的session_id，登录时得到
///@remark 需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线
virtual void OnInsertAlgoOrder(XTPStrategyInfoStruct* strategy_info, XTPRI *error_info, uint64_t session_id) {};

///algo业务中撤销策略单的响应
///@param strategy_info 用户撤销的策略单的具体信息
///@param error_info 撤销策略单发生错误时返回的错误信息，当error_info为空，或者error_info.error_id为0时，表明没有错误
///@param session_id 资金账户对应的session_id，登录时得到
///@remark 需要快速返回，否则会堵塞后续消息，当堵塞严重时，会触发断线
virtual void OnCancelAlgoOrder(XTPStrategyInfoStruct* strategy_info, XTPRI *error_info, uint64_t session_id) {};

///当客户端与AlgoBus通信连接断开时，该方法被调用。
///@param reason 错误原因，请与错误代码表对应
///@remark 请不要堵塞此线程，否则会影响algo的登录，与Algo之间的连接，断线后会自动重连，用户无需做其他操作
virtual void OnAlgoDisconnected(int reason) {};

///当客户端与AlgoBus断线后重新连接时，该方法被调用，仅在断线重连成功后会被调用。
virtual void OnAlgoConnected() {};
```

二、新增接口用法

1. 采用算法总成策略下单逻辑

使用新增的算法总成接口，需要在用户成功登录了 XTP 交易服务器后执行如下逻辑：

（1）调用 LoginALGO () 登录 algo 算法服务器

（2）调用 ALGOUserEstablishChannel () 为已登录用户建立算法通道

（3）在 OnALGOUserEstablishChannel () 回调接口返回成功时，调用 InsertAlgoOrder() 为用户建立算法策略，注意其中的算法参数请参考具体算法的要求

（4）在 OnInsertAlgoOrder() 回调接口会返回用户算法策略建立情况

（5）OnStrategyStateReport 回调接口会通知用户策略运行状态

（6）在不想用算法下单时，调用 CancelAlgoOrder() 撤销用户策略

（7）OnCancelAlgoOrder() 回调接口会通知用户撤销策略情况

2. 与算法总成服务器断连、重连逻辑

当客户端与算法总成断线后，会有如下逻辑发生：

（1）OnAlgoDisconnected () 回调接口将会被调用，此时需要尽快返回，不可挂起回调线程

（2）Api 会进行自动重连，无需用户操作

（3）当 api 与算法总成服务器重连成功后，OnAlgoConnected () 回调接口将会被调用

（4）在重连成功后，无需重新建立算法通道

3. 确认算法子母单对应信息

根据订单信息得到算法母单编号：
调用此接口 GetAlgorithmIDByOrder () 即可通过算法子单得到对应的母单号