**委托控制**

一、粗/细粒度委托控制逻辑比较

**（一）粗粒度委托控制。**

**（二）细粒度委托控制。**

收到on\_bar调用

cancel\_all全撤

执行新一轮委托

全成或撤单清除缓存

已结束才允许下单l检查委托号缓存

已结束才允许下单l检查委托号缓存

必要时执行撤单b下单后缓存委托号÷

必要时执行撤单b下单后缓存委托号÷

以上是委托控制的基本原理，但委托控制的具体实现有无限种可能，每个人使用的委托算法都可能不同，主要依赖个人经验，只有通过大量实践才能找到最适合自己策略的委托控制算法。

二、关于委托状态（OrderData.Status）

（一）on\_order是order的回调函数，用于接收委托的状态变化。只要调用发单函数挂出去1个委托，不管这个委托是否会被交易系统拒单。vnpy会立即推送委托状态，第一个委托状态就是**SUBMITTING**提交中，提示交易者vnpy已经将单子发出去了。此时委托在交易系统端的状态，vnpy还不知道，要等vnpy收到交易系统端的反馈之后才会更新委托状态。

（二）非交易时段，ctp会直接拒单，会收到委托状态为**REJECTED**。有的股票柜台，例如鼎点、恒生等，如果在非交易时段报单，交易系统不会有反馈，委托会被缓存在系统里，交易者的交易界面上会显示该委托一直处于**SUBMITTING**状态。这种委托状态，有的可以撤，有的不可以撤，要等到开盘之后才能反馈委托在交易系统中的状态。

（三）★交易时段，委托进入交易系统后，首先会进入orderbook，如果未能立即成交，该委托会一直挂在orderbook，系统会向vnpy反馈状态**NOTTRADED**，该委托可以撤销。此时，如果恰好有对手委托过来，可以立即成交，但可能只是部分成交，部分成交则交易系统会向vnpy反馈**PARTTRADED**状态，也可能全部成交，那么交易系统则会向vnpy反馈**ALLTRADED**状态。

（四）**SUBMITTING、NOTTRADED、PARTTRADED**都是委托还处于**ACTIVE\_STATUSES**状态，他们的共同特征是都还可以撤单。★★★

（五）**ALLTRADED、CANCELED、REJECTED**都是委托已经处于**结束状态**，该委托已经被踢出orderbook，此笔委托最终结果已定，无法撤销。例如：1.在非交易时段发单，得到**REJECTED**状态反馈；2.发送的价格跳动值不对，也会得到**REJECTED**状态反馈；3.仓位不足、被交易系统风控等都会得到**REJECTED**状态反馈；4.交易者主动撤单，则会得到**CANCELLED**状态反馈；5.全部成交，则会得到**ALLTRADED**状态反馈。

（六）鉴于上述，在发出委托前要确认此前的委托是否已经结束，一定要在收到状态回报后（必须等到撤销回报收到之后）再发出新的单子，这样才能达到比较精确的控制。委托结束就会被清空缓存，所有已经结束的委托的缓存被清空之后才能下单。

判断委托是否结束的代码路径：

vnpy/app/trader/converter/class position\_holding

self.active\_orders = {}

其中submiting提交中、not\_traded未成交、part\_traded部分成交都属于active状态。

（七）停止单（stoporder）。

1. 停止单又叫条件单，停止单的委托价格是触发价格，当行情价格突破触发价格时，交易委托被触发，立即发出市价委托或者能立即成交的委托，如果未能成交，会持续往更高/更低的价格下单，直到成交。在大多数情况下，停止单委托成交价都是比触发价格更差的，但也不排除更好的可能。

2. 停止单分为本地停止单和服务器停止单。本地停止单在交易系统本地维护，但实际上交易引擎会将其自动转化为限价单发出执行；服务器停止单，即交易引擎提前将委托发到远端服务器上等待下单委托被触发。国内期货市场是不支持停止单的，vnpy中的停止单其实是本地停止单，最终发出去也是转化为超价的限价单。

3. CTP有内置的服务器停止单，只不过绝大多数期货公司都没有开。IB及很多币圈交易所都支持服务器端停止单。

4. 发送停止单的时候，引擎会自动每个tick检查一次价格。

5. 发送停止单，如果不考虑开盘价，会导致偷价。

三、关于成交状态（TradeData）

（一）vnpy发出去的每一笔委托都会有多个on\_trade的反馈，因为每成交一次就会有一次的推送。委托和成交是两个并行的数据流，是不一样的。

（二）委托成交后，我们会先收到on\_order反馈，在CTP中，或国内绝大部分API都是**先收到on\_order推送，然后收到on\_trade推送**。这个规则是中国期货市场独创的，国际上的FIX协议是on\_order和on\_trade数据同时推送，跟中国不一样的。CTA策略引擎收到on\_trade推送后，会计算并修改self.pos字段值，然后再去调用on\_trade函数。vnpy是参照CTP设计的，因此也只能依照CTP的规则来接收推送。

（三）股票市场的极速柜台，遵循这个规则，但有个别如恒生、鼎点的系统是乱序推送的，可能是order先到，也可能trade先到，也有可能只收到其中之一。数字货币市场，币安的推送是正常的，其他大部分数字货币交易系统使用websocket通讯协议来推送回报，更加不能保证推送顺序。

（四）因此，这就导致我们在写挂撤单的时候，在期货市场中正确的挂撤单逻辑，应用到其他市场可能就会不能正常运行。针对不同的交易系统，如果要了解推送顺序，只能在on\_order或者on\_trade下加上print来逐项观察。

（五）★这种特殊的推送顺序会导致一个关系策略有效性的重要问题：如果在on\_order函数下，收到order推送后立即调用self.pos值，并不能保证此时self.pos是最新值。在on\_order下要做一些新的操作的话，必须首先判断委托是否已经结束，例如已经收到ALLTRADED状态反馈，那么不能在on\_order下发新单子，而是应该把发新单子的命令放到on\_trade函数下。

（六）self.pos要满足两个条件才是准确的：

1. 外部没有活动委托；

2. 在on\_trade下收到最新的成交回报。但对于分钟级别的CTA策略来说，有99.9%的概率，self.pos值就是on\_bar中获取self.pos时的值。

（七）策略持仓--self.pos的计算逻辑位于vnpy/app/cta\_strategy/engine.py中的CtaEngine的process\_trade\_event中，是基于成交回报来计算的。不要混淆self.pos和PositionHolding，PositionHolding是vnpy引擎层组件中的账户级持仓，和这里应用层的策略持仓无关。

（八）如果是期货的本地停止单，在尚未成交时撤销和不撤销没有本质区别，因为委托只是缓存在本地而已。币圈交易所提供服务端停止单，在服务端的停止单可能之前是被挂在比较靠前的位置的，但如果总是cancel\_all，那么会一直排在最后面，滑点就会比较明显。

（九）如果发出去的委托是stoporder，那么收到回调的只能是on\_stop\_order。on\_order收到的是底层接口推送过来的信息，对于CTP来说，只有限价单、市价单等，没有停止单。停止单的功能是vnpy在CTA策略引擎内部模拟实现的，所以如果下了停止单就只有on\_stop\_order能收到推送，如果把逻辑写在on\_order下，就会完全收不到反馈。

（十）在on\_stop\_order下，因为没有办法确定self.pos是否就是最新的数字，所以只能在确定stoporder.status为canceled时才能再接下来写进一步的逻辑。如果stoporder.status为triggered，意味着单子刚发出去，无法判断self.pos此时是否是最新值。

四、关于CTP持仓查询

（一）CTP不提供持仓状态的实时推送。

（二）每隔3秒，vnpy的gateway会执行一次持仓查询，从CTP服务器获取持仓状况（又说：3秒查一次资金，3秒查一次持仓，持仓6秒得到一次，到底是怎样，有待核实）。

（三）在3秒间隔之间，即使有下单操作导致实际持仓发生变化，在gateway执行下一次持仓查询之前，在vntrader界面上显示的持仓是不会发生变化的。但是成交（on\_trade）的数据是实时推送的，因此可以通过成交数据实时计算持仓数据。

1. 通过update\_trade得到self.long\_pos、self.long\_yd、self.long\_td、self.short\_pos、self.short\_yd、self.short\_td这6个字段的信息。

2. 要知道完整的position\_holding信息，还需要用update\_order函数来计算被冻结的持仓量。

3. update\_order\_request，因为主动下单后，系统不会立即发回委托回报，但是为了后续能够准确下单，需要立即更新持仓冻结信息，因此需要自己进行计算。update\_order\_request就是起这个作用。

（四）关于仓位冻结

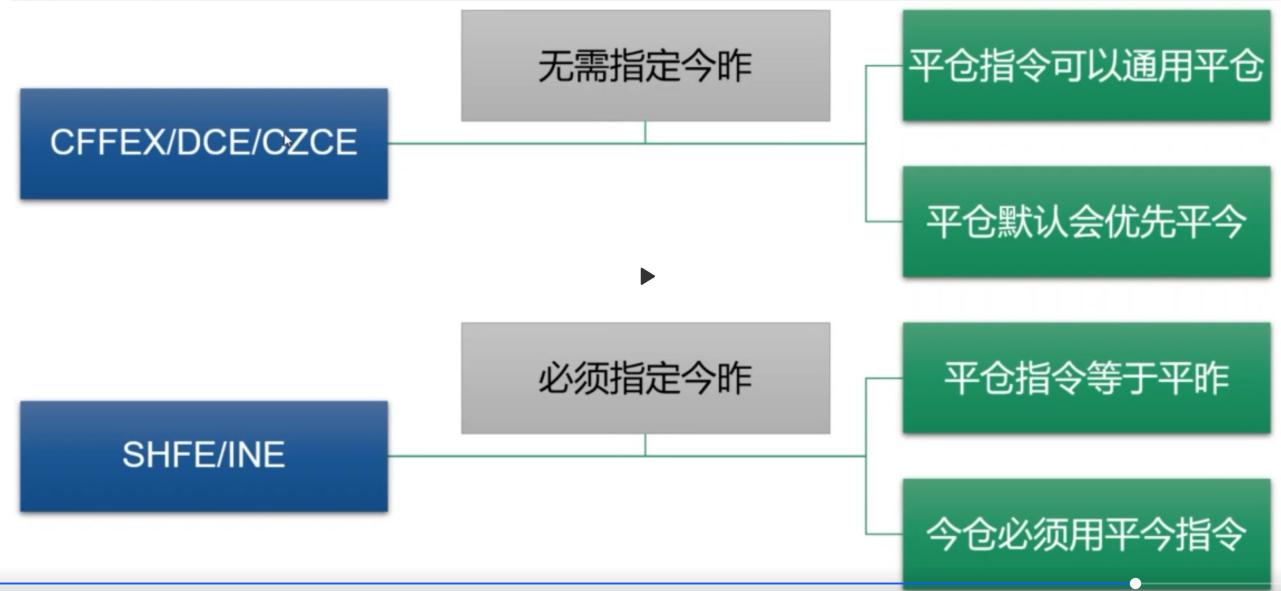
例如持有多仓10手，如下8手平仓委托，那么原来的10手多仓就会有8手被冻结用于等待成交，此时可用的多仓就只有2手，因此如果此时在下3手平仓委托就会被拒单，因为只有2手可用。为了接下来的卖平能够成交，应该先把之前的卖平8手先撤掉，但并不是撤单指令发出去就会立马成交，因此不能撤单之后立即写卖单指令，应等收到撤单成功回报之后再发出卖单指令。

五、关于开平转换

开仓平仓是中国期货市场独有的交易模式。

**（一）上期所模式。**

在上期所交易，如果是开仓请求，不用转换。在平今仓、昨仓时需要分别发出平今、平昨指令。其他交易所支持平仓指令自动转换，优先平今。上期所在默认情况下，平今是有手续费优惠，平昨是没有手续费优惠的。

**（二）锁仓模式。**

今日开仓后，不能反向平仓，否则会有平今惩罚手续费率。采用反向开仓的模式来实现平仓；隔日开仓时，优先平反向的昨仓。用到锁仓的主要是中金所，CFFEX。

六、自定义K线的意义

按照标准时间切分法合成K线，那么大概率会遭遇大量人员同时下单，会造成成较大成交滑点。因此，为减少滑点，就需要找大多数人都不太常用的时间切分法来合成K线。

**2021年4月27日星期二**

陈老师，因为想请教一个关于委托控制的问题，感觉有点儿复杂，就直接联系您了。

在咱们CTA策略课程里介绍了精细化委托控制的方法，主要是通过创建buy\_vt\_orderids、sell\_vt\_orderids、short\_vt\_orderids、cover\_vt\_orderids这4个缓存列表来实现。我理解这个方法的核心目的，是为了在撤销未成交委托的同时可以立即重新发单，从而减少交易机会的浪费。不知道理解的对不对。

但问题是，因为不能保证在on\_order或者on\_stop\_order函数下self.pos值是当前最新值，所以往往会出现基于错误的self.pos值发出的错误追单，最后交易逻辑就会陷入混乱。

所以，我想请教您的是：

1. 在回测模式下，有没有能规避上述矛盾的方法，来实现更准确的委托控制？

2. 在实盘模式下，应该如何规避这个问题呢？

**答：on\_order收到后，更新当前的全局委托状态，撤单成功直接可以继续发。成交则要在收到on\_trade后去执行相关的操作。然后主要就是对委托回报的流程有完整的概念，在写细粒度挂撤单控制的时候，能把各种可能性都覆盖到，这个就只有靠经验了。**