高斯贝尔Nvod软件需求规格

Version 0.2

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2016.08.09 | 0.01 | 完成和功能相关的章节: Chapter 8, Chapter 9 | 刘浩 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Project Drivers

# The Purpose of the Project

## The User Business or Background of the Project Effort

## Goals of the Project

# The Client, the Customer, and Other Stakeholders

# Users of the Product

Project Constraints

# Mandated Constraints

## Solution Constraints

### Constraints on Operation System Compatibility

#### Description

本软件必须兼容的OS平台包括Windows 10(64 Bits)，Windows 7(32 Bits)。

#### Rationale

用户家里有2台电脑,1台运行64位Windows 10, 另1台运行32位Windows 7，公司的工作电脑运行32位Windows 7。为了在公司和家里都能够操作/调试本软件， 程序的代码必须兼容这2种OS。

## Anticipated Workplace Environment

### Constraints on Different Workplace for Identical Account

#### Description

允许用户在不同的地点的电脑上运行本软件。 为了程序的简单性和正确性，如果用户在同一个时间点上运行了多个本程序的实例，对于任何一个交易账户，只可能有1个本软件的进程的的实例可以为此交易账户产生订单并保存交易数据道数据库。

最后启动的那个本软件的实例退出（正常或异常）以后， 其它更早启动的那些进程也不在处理相关交易帐户的交易。 这样做的目的是为了避免程序bug可能带来的经济损失，同时也确保程序的简单性。

#### Rationale

通常的用户都拥有多台电脑，并且希望在多个地点使用不同的电脑来运行本软件，本软件应该满足用户的这个需求，并且保证数据的正确性。

## Partner or Collaborative Applications

### Constraints on Dealing Platform

#### Description

本软件必须与当前主流的几家bitcoin交易平台兼容，包括:

1. http://www.btctrade.com
2. https://www.okcoin.cn
3. https://www.huobi.com
4. https://www.chbtc.com

#### Rationale

1. 本软件需要同时与多个交易平台交互，以赚取多个交易平台间的价格差。
2. 本软件需要同时在多个交易平台做低进高出的短线操作。

## Budget Constraints

### Description

在需求明确之后，初步计划用40个小时的工作时间来完成第一版程序的设计、编码、测试工作。

### Rationale

根据目前了解的情况，本软件**可能**会有经济价值，但是需要进一步验证。为了避免耗费太多的精力开发没有经济价值的软件，用于验证方案可行性的第一个版本的开发时间应该进可能的短。

# Naming Conventions and Definitions

## Definitions of All Terms, Including Acronyms, Used in the Project

### Ask Price

卖方出价

### Bid Price

买方应价

### Cost Price

成本价

### Selling Price

售价

### Expected Price Range

BuyLowSellHigh-0 算法的预期价格波动范围。 robot在此范围内买入的操作。也就是说，Expected Price Range 以买入价为标准。

### Buying Price Point Number

Buying Price Point Number = Expected Price Range / Expected Profit threshold

为了减少需求分析和代码的复杂度，我们规定Buying Price Point Number的值必须大于等于5。

### Waiting Area

每次价格行情波动超出当前Waiting Area的范围， Robot就必定会为此做应对的动作。在此定义术语Waiting Area的主要目的是为了更清晰的描述订单列表，订单列表中每个状态都牵涉到一个Area。

Waiting Area的位置和取值范围如下：

* Order List 中只有买单，Waiting Area位于买单的上方，取值范围为：Expected Profit threshold。
* Order List 中只有卖单，Waiting Area位于卖单的下方，取值范围为：Expected Profit threshold。
* Order List中既有买单也有卖单。Waiting Area位于买单和价格最低的卖单之间，取值范围为：Expected Profit threshold \* 2。

### Max Disposable Amount

单个robot实例可操作某一个账户的资金的上限。如果当前robot关联了多个账户，此robot可操作的资金的总数为：Max Disposable Amount \* 关联账户的数量。

### Price Difference Unit (Pdu)

（卖价 - 买价）/ 买价，Pdu以1/10000 为单位计量。 如买价位9元，卖价为10元， 则Expected Profit threshold = (10 - 1) / 10 = 1000 / 10000, 即1000 Pdu。

BuyLowSellHigh-0 算法：单次买-卖操作的预期收益。

Hedging-0 算法：单此买-卖操作的预期收益，由于收到价格信息到挂出买卖单，再到实际产生交易之间3者之间的时间差，robot有可能在交易完成前就修改挂出的买卖价。所以Hedging-0 不能保证每组交易最终获得的利润等于预期收益。

### Expected Profit Threshold

BuyLowSellHigh-0 算法：单次买-卖操作的预期收益。

Hedging-0 算法：单此买-卖操作的预期收益，由于收到价格信息到挂出买卖单，再到实际产生交易之间3者之间的时间差，robot有可能在交易完成前就修改挂出的买卖价。所以Hedging-0 不能保证每组交易最终获得的利润等于预期收益。

### Buy Low Sell High Robot (Blshr)

### Margin Level Number (Mln)

### Hedge Robot (Hr)

### Order Pricing Mode (Opm)

### Final Profit threshold

### Client, Server

在本文档中，Client指的是客户端软件的一个实例（进程），Server指的是服务端软件的一个实例（windows平台的一个后台服务，可以通过windows的services.msc工具查看/管理其运行时状态）。

### Trading Platform

交易平台， 如：btctrade.com, okcoin.com。

### Order State Inquiry Interval（Milliseconds）

Robot挂单完成后，每2次查询订单状态之间的时间间隔，

### Process Account State

当前交易账户在当前进程中的状态, 可能有以下几种状态:

1. Checking: 检测当前时间点当前账户是否在其他进程中处于Active状态。
2. Active：当前进程可以为当前账户下单，并保存交易信息到数据库。
3. Deactivate： 当前进程不能为当前账户下单，对数据库中的信息只有读权限。

## Data Dictionary for Any Included Models

### Buy/Sell Order

买/卖bitcoin的订单， 包含以下信息：

1. Id， 订单Id。
2. Post Time，挂单时间。
3. Order Type， Buy or Sell。
4. Amount，买卖bitcoin的数量，以BTC为单位。
5. Unsold，此订单还未完成的Amount。

### Transaction Records

A list of Buy/Sell Order。

# Relevant Facts and Assumptions

Functional Requirements

# The Scope of the Work



# The Scope of the Product



## Ur-1, Configuration

### Brief Description

Client可以是某个CLI命令行程序，也可以是WEB应用。 每一种Client和Server都是单独的应用程序， 重启某个Client不会影响到Bitcoin的交易， 也不会对其他的Client的使用产生影响。 用户可通过Client修改本软件的运行时参数。

完整的配置参数列表，请参考相关章节。

为了让整个程序正确运行，用户必须知道以下几点：

1. Command descriptions use these conventions:

* Commands and keywords are in **boldface** text.
* Arguments for which you supply values are in *italic*.
* Braces ({}) group required choices, and vertical bars ( | ) separate the alternative elements.
* Braces and vertical bars within square brackets ([{ | }]) mean a required choice within an optional element.

1. 所有可配置的参数分为Client参数（如Server Ip Addr配置）和Server参数，其中Server参数又分为交易无关参数（如Database Ip Addr）和交易相关参数（如Profit Benchmark， Trading Platform Ip Addr）。

如果是CLI Client，用户配置的Client参数保存在CLI Client可执行程序所在目录，文件名为client.xml。Server交易无关参数保存在Server可执行程序所在目录，文件名为server.txt。交易相关参数保存在Database。

通过同一个Client可配置所有的参数。

1. 一旦用户按下回车键，没有语法错误的配置立即生效，不需要重启软件。但是，只有输入了save命令的情况下，之前输入的配置才会被永久保存，否则，软件重启后将退回到上一次输入save命令是所对应的配置。

### Configuring Server Global Parameter

|  |  |  |
| --- | --- | --- |
| Step | Command | Description |
| Step 1 | **set ip-server** *ip-addr port* | 配置Client连接到Server所需的Ip 地址和 Tcp 端口号。 |
| Step 2 (opt) | **show ip-server** | 验证Server连接配置。 |
| Step 3 | **set price-query-interval** *millisecond* | 设置market查询间隔 |

### Save Configuration

|  |  |  |
| --- | --- | --- |
| Step | Command | Description |
| Step 1 | **save** [{**all** | **client** | **server**}] | 保存all/client/server/ 配置， 其中all等同于分别保存client和server的配置。没有输入[{**all** | **client** | **server**}] 的情况下，缺省值为all。 |

### Reboot Server

|  |  |  |
| --- | --- | --- |
| Step | Command | Description |
| Step 1 | **reboot server** | 重启与当前Client相连的Server， 其效果等同于在Windows 命令行中输入services.msc，然后windows服务管理工具中停止、启动服务。 |

### Configuring Database Connection

|  |  |  |
| --- | --- | --- |
| Step | Command | Description |
| Step 1 | **set database** *ip-addr port username password {mysql | sqlserver}* | 配置Server连接到database的Ip地址， Tcp 端口好，用户名，用户密码以及数据库类型。 |
| Step 2 (opt) | **show database** | 验证数据库连接配置。 |

### Configuring Dealing Platform Account

|  |  |  |
| --- | --- | --- |
| Step | Command | Description |
| Step 1 | **create account** *userid platform-id* [**key** *public-key private-key*] *[description]* | 创建Server可操作的账户。  **key**: 通过http和交易平台交互时所需要的公钥和密钥。 |
| Step 2 (opt) | **set account key** *public-key private-key* | … |
| Step 3 (opt) | **del account** *userid* | 删除Server可操作的账户。  如果robot曾经在当前账户上做过买或卖的操作，则此账户不能被删除。  如果存在正在操作当前账户的robot，必须先删除/终止robot后，再删除账户。 |
| Step 5 (opt) | **show account** | 验证数据库连接配置。 |

### Configuring Robot

|  |  |  |
| --- | --- | --- |
| Step | Command | Description |
| Step 1 | * hedging-0   **create robot hedging-0** *robot-id* {**btc** | **ltc**} [**userid** *userid1 userid2*][**order-query-interval** *millisecond*]  [**profile-threshold** *expected-margin-threshold final-profile-threshold*]  **[max-disposable-amount** *amount*]   * buy-low-and-sell-high-0   **create robot buy-low-and-sell-high-0** *robot-id* {**btc** | **ltc**} [**userid** *userid*]  **profile-threshold** *expected-margin-threshold* | 创建自动交易机器人，并将需要操作的账户绑定到当前机器人。  Robot的算法只能在创建时指定，将来不能修改。  **hedging-0**:采用hedging-0算法的机器人。  **buy-low-and-sell-high-0**:采用buy-low-and-sell-high-0算法的机器人。  **order-query-interval**: 2次查询订单状态的间隔  **max-disposable-amount**: 参考术语”Max Disposable Amount”。 |
| Step 2 (opt) | **set robot-associated-account** *robot-id* **userid** *userid1* [*userid2*] | 配置与当前robot关联的交易账户 |
| Step 3 (opt) | **set robot-max-disposable-amount** *robot-id amount* | 配置与当前robot可支配的RMB金额。  为了确保配置过程不对交易产生负面影响，我们要求运行当前命令的前提为：robot状态只能是Initiation 或 Ready。 |
| Step 4 | **start robot** *robot-id* | 启动robot。 |
| Step 5 (opt) | **stop robot** *robot-id* | 停止robot。 |
| Step 6 (opt) | **reset robot transaction** *robot-id* | 命令robot快速退回到Ready状态，执行此命令时，robot必需出于Pausing状态。  我们使用此命令让当前robot快速的进入到刚配置好参数还未参与交易时的状态。即，取消所有买入币的订单，将当前robot买入的币全部卖出，然后切换到Ready状态。 |
| Step 7 (opt) | **del robot** *robot-id* | 删除自动交易机器人。  如果当前robot已经产生过买或卖的交易，则robot不能被删除。 |
| Step 8 (opt) | **show robot** | 验证自动交易机器人配置。 |

### Exporting Configuration

|  |  |  |
| --- | --- | --- |
| Step | Command | Purpose |
| Step 1 | **export configuration** *file-name* | 导出当前服务端运行时的配置到本地xml文件。 |

## Ur-2, Transaction

本软件支持同时运行多个robot，可以为不同的robot设置相同或不同的算法，每个robot都需要关联1个或多个交易账户。同1个交易账户可以同多个robot关联，但是在任何时刻，最多只能有1个robot处于Running状态。



本章我们将基于算法详细说明robot自动化交易的处理过程，将1种算法视为本用例的1个子用例。

### Ur-2-1，Hedging-0

#### Brief Description

一个robot同时关联到多个账户，每个账户属于不同的交易平台，假设为PlatformA，PlatformB, 账户A、B， 其中有一个帐户为Primary Account，其它的帐户都为Subordinate Account。 主账户和一个Subordinate Account 配置为1对Hedging 账户组合，有多少个Subordinate Account就会配置多少对账户组合。每个帐户可分别配置可用于交易的钱和币的数量。 robot 每隔t秒查询查询1次价格，如果PlatformA 的委托卖价高于PlatformB的委托买价, 且价格差大于expected-margin-threshold, robot此时在PlatformA卖出, PlatformB买入， 我将此次操作的卖价-买价定义为 FirstDealingMargin。Robot继续查询2个平台的价格， 当PlatformB的委托卖价比PlatformA的委托买价相对上升，并且 FirstDealingMargin + （PlatformB委托卖价 – PlatformA委托买价）>= final-profile –threshold时，robot 在PlatformA做买操作， 在PlatformB做卖操作。 然后，Robot又回到初始状态，等待下一次交易的时机的到来。

我们将从A卖币，从B买币的价格差记为 A->B expected-margin-threshold, 可将->理解为币的移动方向。

Robot 需要动态的调节Expected Profit Threshold。 有2种调节机制：

1 Horizontal Comparison Expected Profile Threshold调节机制。

2 Vertical Comparison Expected Profile Threshold调节机制。

名词解释：

HorizontalMarginAdjustInterval: Seconds，0表示不执行Horizontal利润调节

VerticalMarginAdjustInterval: Seconds，0表示不执行Vertical利润调节

HedgeVerticalMarginLevelNumber: 奇数， >= 3

HedgeOrderPricingType: 枚举值为Ask/Bid Price Type，Last Price Type。 Robot下买单时，可根据当前行情的的Ask/Bid Price下单或用Last Price下单。

HedgeOrderPricingStrategyType: 取值范围为AskPriceOnly， LastPriceOnly, AskAndLastPrice， LastAndAskPrice。 如果配置为AskPriceOnly， LastPriceOnly，则robot只创建单一HedgeOrderType类型的买卖单。 如果位置为AskAndLastPrice，robot会为钱币创建Ask Price Type和Last Price Type（一小部分）的买卖单。

HrPlatformPair State Machine, 假设ApiAverageResponseTime PlatformA > PlatFormB。

OpmAbab: 表示一组买卖交易中， 在其买币的Platform Opm为AskBidPrice， 在其卖币的Platform Opm也是AskBidPrice。

OpmAbal: 表示一组买卖交易中， 在其买币的Platform Opm为AskBidPrice， 在其卖币的Platform Opm为AskLastPrice。

Untouched Coin Number (ucn): 假设robot已经下了买/卖单，每次查询到买/卖单完成的数量大于平台minTradingUnit时，我们会立即在Pair Platform 下相应的卖/买单。Ucn 就是在A平台已经买入/卖出的币且没有在B平台卖出/买入的币的数量。

|  |  |
| --- | --- |
| 事件 | 处理过程 |
| Buy at NoOrder | acc.availableMoney -= order.limitedPrice \* order.coinNumber |
| Sell at NoOrder |  |
| Cancel Step0-Buy | acc.availableCoinNumber += order.filledCoinNumber  acc.availableMoney += order.limitedPrice \* order.filledCoinNumber  acc.residualCoinNumber += order.filledCoinNumber  acc.residualCoinCostPrice = order.filledCoinNumber |
| Cancel Step0-Sell |  |
| Cancel Stop-Loss Buy | acc.availableCoinNumber += order.filledCoinNumber  acc.availableMoney += order.limitedPrice + order.filledCoinNumber |
| Cancel Stop-Loss Sell |  |

1. PlatFormA Opm = AskBidPrice, PlatFormB Opm = AskBidPrice

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 当前状态 | 事件 | 处理过程 | | | | | 目标状态 |
| No Order | Ptm0 and Ptm1 market info received | Ptm1.BidPrice – Ptm0.AskPrice < margin && Ptm0.BidPrice – Ptm1.AskPrice < margin, do nothing | | | | | No Order |
| Ptm1.BidPrice – Ptm0.AskPrice >= margin, 在ptm0下买单, | | | acc0.availableMoney < price \* minTradeUnit, 仅记录日志 | | No Order |
|  | | Buy Created |
| Buy Created  (At PtmA) | PtmA market info received | ~~AskPrice < Order Price, 可以肯定Buy Order 已经filled, 且PtmB的BidPrice还没有大的下跌（否则当前应该是Canceling Buy Order状态）。 Create Sell Order with Price = Last PtmB Bid Price - minPriceSize~~ | | | | | ~~Buy Order Filled, Sell Created~~  ~~(At PtmB)~~ |
| GetBuyPrice(ptmAMarket, true) > Order Price  => PtmA 的价格上涨  1 CancelOrder()  2 FetchOrder() | | PtmA.GetTotalVolume() >= ptmB.minTradingUnit | | 1 create sell order | Sell Order Created |
| PtmA.GetTotalVolume() < ptmB. minTradingUnit | | 1 刷新cookie->residualCoin, cookie-> concludedPrice  2 CastEvent(BuyOrderCancled)  3 automaton.HandleMarket() | unknown |
| PtmB market info received | 1 PtmB Bid Price – Last PtmA Ask Price >= margin, do nothing | | | | | - |
| 2 PtmB Bid Price – Last PtmA Ask Price < margin, Cancel Buy Order | | | | | Canceling Buy Order |
| Order info received | Order unfilled， do nothing | | | | | - |
| Order filled, 可以肯定PtmB的BidPrice还没有大的下跌（否则当前应该是Canceling Buy Order状态）。Create Sell Order with coin number = buy order number, price = Last PtmB Bid Price - minPriceSize | | | | | Buy Order Filled, Sell Created  (At PtmB) |
| Order part filled, 此处需要计算ucn | | 1 ucn >= minTradingUnit 并且 unfilled coin number >= minTradingUnit, Create Sell Order with price = Last PtmB Bid Price - minPriceSize | | | BuyAndSell Order Created |
| 2 ucn < minTradingUnit 或 unfilled coin number < minTradingUnit, do nothing | | | - |
| Sell Created  (At PtmA) | PtmA market info received | BidPrice > Sell Order Price, 可以肯定Sell Order 已经filled， 且PtmB的AskPrice没有大的上涨， （否则当前应该是Canceling Sell Order状态）。 Create Sell Order with Price = Last PtmB Bid Price – minPriceSize。 Create Buy Order with price = Last PtmB Ask Price + minPriceSize | | | | | Sell Order Filled, Buy Created  (At PtmB) |
| BidPrice < Sell Order Price - minPriceSize, 说明PtmA的价格下跌了， Cancel Sell Order（在价差仍然高于margin的情况下是否应该立即创建一个新的Sell order？） | | | | | Canceling Sell Order |
| PtmB market info received | 1 Last PtmA Bid Price - PtmB Ask Price >= margin, do nothing | | | | | - |
| 2 Last PtmA Bid Price - PtmB Ask Price < margin, do nothing, Cancel Sell Order | | | | | Canceling Sell Order |
| Order info received | Order unfilled， do nothing | | | | | - |
| Order filled, Create Buy Order with coin number = sell order number, price = Last PtmB Ask Price + minPriceSize | | | | | Sell Order Filled, Buy Created  (At PtmB) |
| Order part filled, 此处需要计算ucn | 1 ucn >= minTradingUnit 并且 unfilled coin number >= minTradingUnit, Create Bue Order with price = Last PtmB Bid Price + minPriceSize | | | | BuyAndSell Order Created |
| 2 ucn < minTradingUnit 或 unfilled coin number < minTradingUnit, do nothing | | | | - |
| Buy Canceled | PtmA market info received |  | | | | | Buying Plus |
| Buying Plus |  |  | | | | |  |

#### Ur-2-1-1 Start Robot

#### Basic Flow

1. 用于第一次启动Robot。
2. Robot读取expected-margin-threshold 初始配置，设置

emt-low = expected-margin-threshold - ExpectedPrifitAdjustSize

emt-middle = expected-margin-threshold

emt-high = expected-margin-threshold + ExpectedPrifitAdjustSize

每一对Hedging 账户组合产生2组{ emt-low， emt-middle，emt-high}。

1. Robot 将Primary Account 的钱，币分成4份， 1/4按照{emt-low, last price / {bid/ask price}}来操作，1/4按照{emt-middle, last price}来操作，1/4按照{emt-middle, bid/ask price }来操作， 剩下的1/4按照{emt-high, last / {bid/ask price}}来操作。

#### Ur-2-1-2 Initial状态,收到Market Info

#### Basic Flow

1. 如果PlatformA Bid-Price - PlatformB Ask-Price > emt-high,。

#### Ur-2-1-3 调整Order Pricing Mode

1. 2个将参与Order Pricing Mode调整运算的HraGroup 分别按照不同的Mode下订单。一段时间之后，orderPricingModeAdjustInterval 超时, 且对应的HraGroup没有未完成的Order时, robot 会启动Order Pricing Mode调整的处理流程。
2. 为了正确的调整Order Pricing Mode，robot通过比较在一定时间段内2个HraGroup的盈利情况：标识为 TrialForOrderPricing 的HraGroup 和 与此具有相同HrMarginLevelNumber 的HraGroup。 假设这2个HraGroup分别为HraGroupA 和 HraGroupB，HraGroupA包含{ HraGroupA-AccountA，HraGroupA-AccountB }， HraGroupB包含{ HraGroupB-AccountA，HraGroupB-AccountB }
3. 因为2个HraGroup分别有3种Order Pricing Mode {AskBidPrice， AskLastPrice， LastBidPrice}，对应9种排列组合。 因为每次只能在2个HraGroup中各选取1中Price Mode来下单产生实际的用于计算的数据，所以我们需要重复多次{按某种方式下单并产生数据，利用数据计算最优的Pricing Mode}。 我们按照如下的算法，逐步找出最优的Order Pricing Mode。

|  |  |  |
| --- | --- | --- |
| 阶段 | 处理流程 | 结果 |
| 1.1 | HedgeRobot正常运行, 4个Account分别按照如下的方式下订单:  AccountAA：AskBidPrice  AccountAB：AskBidPrice  AccountBA：AskBidPrice  AccountBB：AskLastPrice | - |
| 1.2 | 一段时间后，Robot 根据这段时间产生的利润比较2种Order Pricing Mode 组合中相对较优的一个组合 | P1L = AskBidPrice  If Profile(AccountAA) > Profile(AccountAA)  P1L = AskBidPrice  else  P1L = AskLastPrice |
| 2.1 | HedgeRobot正常运行, 4个Account分别按照如下的方式下订单:  AccountAA：AskBidPrice  AccountAB：P1R  AccountBA：AskBidPrice  AccountBB：LastBidPrice | - |
| 2.2 | 一段时间后，Robot 根据这段时间产生的利润比较2种Order Pricing Mode 组合中相对较优的一个组合 | P2L = AskBidPrice  If Profile(AccountAA) > Profile(AccountAA)  P2L = P1R  else  P2L = LastBidPrice |
| 3.1 | HedgeRobot正常运行, 4个Account分别按照如下的方式下订单:  AccountAA：AskBidPrice  AccountAB：P2R  AccountBA：AskLastPrice  AccountBB：P2R |  |
| 3.2 | 一段时间后，Robot 根据这段时间产生的利润比较2种Order Pricing Mode 组合中相对较优的一个组合 | If Profile(AccountAA) > Profile(AccountAA)  P3L = AskBidPrice  else  P3L = AskLastPrice  P3R = P2R |
| 4.1 | HedgeRobot正常运行, 4个Account分别按照如下的方式下订单:  AccountAA：P3L  AccountAB：P3R  AccountBA：LastBidPrice  AccountBB：P3R |  |
| 4.2 | 一段时间后，Robot 根据这段时间产生的利润比较2种Order Pricing Mode 组合中相对较优的一个组合 | If Profile(AccountAA) > Profile(AccountAA)  P4L = P3L  else  P4L = LastBidPrice  P3R = P2R |

1. 经过上表的4个步骤以后，我们已经知道并且开始使用最优的order pricing mode运行hedge robot。 但是行情始终在变化，我们还需要继续根据最新的交易情况调整Order Pricing Mode。 此时hraGroupA 始终以最优的mode运行hedge robot，hraGroupB则每次变更1个account的mode，直道尝试完2个账户多种mode。
2. 重复第4步的过程。

#### Horizontal Comparison Expected Profile Threshold调节

1. Robot 每隔一段时间（Horizontal Comparison Interval）做一次检查， 在A，B账户均有币可卖且有钱可买币的情况下，以emt-middle价格产生的交易次数为基准，如果A卖出币的次数高于从B卖出币的次数，则下调B->A 相关的emt-low，emt-middle， emt-high。 反之依然。

|  |  |  |
| --- | --- | --- |
| 订单完成情况 | 价格变化方式 | 处理方法 |
|  |  |  |

#### Vertical Comparison Expected Profile Threshold调节

1. Robot 每隔一段时间（Vertical Comparison Interval）做一次检查，从A卖出币的情况。 如果emt-low比emt-middle能产生更好的利润，则下调emt-low，emt-middle， emt-high。如果emt-high比emt-middle能产生更好的利润，则上调emt-low，emt-middle， emt-high。

### Ur-2-2，BuyLowSellHigh-0

#### Brief Description

每一个robot关联到1个账户。每个robot都有一个预设的Expected Price Range， robot刚启动时将当前Bid Price (设为PriceX0) + Expected Price Range / 2设定为Expected Price Range的上限， 将当前Bid Price - Expected Price Range / 2设定为Expected Price Range的下限， 以当前Bid Price买入(Max Disposable Amount / Expected Price Range) 个币， 并挂单以当前PriceX0 + Expected Profit Threshold元的价格卖出。Robot每隔Order State Inquiry Interval查询一次订单状态，如果订单已经完成，robot以PriceX0做为Bid Price挂出买单。 如果之前的卖单还没有完成的情况下，平台上已经出现了低于PriceX0 - Expected Profit Threshold（设为BpX1）的卖单， robot以BpX1买入(Max Disposable Amount / Expected Price Range) 个币，并以PriceX0的价格挂单卖出，然后继续以Order State Inquiry Interval时间间隔查询订单状态， 并重复上述步骤，即：每完成一个买操作，就挂出一个卖单， 反之，每完成一个卖单，就挂出一个买单。

如果币的价格持续上涨，或下跌，将导致robot无法继续产生交易。此时robot需要做调整Expected Price Range上下限的操作。

本章节接下来的内容中， 假设Expected Price Range的值为5， Expected Profit threshold的值为1，当前robot可以{100, 101, 102, 103, 104}的价格下买单。

**注: 任何时候, 当前robot只可能拥有0~1个挂起的买单, 可能拥有0~多个挂起的卖单。**

**注: 任何时候，如果存在多个挂起的卖单， 这些卖单的卖价必定是连续的。**

**注：任何时候，在Expected Price Range中间值（102）的下方不可能有单独的买单或卖单。 如：挂起单列表中不可能是以下几种状态**

* **{101**→**}, 有101价格的卖单， 必定有102和103的卖单。**
* **{**→**101}，有101价格的买单， 必定有103的卖单。**

**注：任何时候, 一个买单成交, 必定产生一个新的卖单。 同样，任何时候, 一个卖单成交, 必定产生一个新的买单（可能会取消一个价格更低的买单）。**

**注：（用户输入Reset Order取消所有订单的情况除外）任何时候, 取消一个未成交的买单， 必定产生一个新的买单。 同样，任何时候, 取消一个未成交的卖单， 必定产生一个新的卖单。**

**注：任何时候, 只要用户输入reset 命令, 不管有没有可卖的币, 都必须产生一个卖单.**

可能产生一个买单的情况：

* 1个卖单成交 -> Robot 产生对应的买单
* Ask Price价格上涨，且买单上方没有卖单 -> Robot 取消掉一个已有买单， 并以更高的买价生成一个新的买单
* Ask Price价格下跌，且最低价卖单下方没有买单 -> Robot生成一个新的买单

可能产生一个卖单的情况：

* 1个买单成交 -> Robot 产生对应的卖单
* Reset 命令

程序接到价格变化或订单状态变化的通知时，与robot相关联的订单列表可能为以下几种状态：

|  |  |
| --- | --- |
| 状态编号 | 订单状态, 产生过程 |
| No Order | 无订单 |
| Waiting Orders | 1. 单元测试用例: WaitingSellingOrders01   产生过程: Waiting Selling Orders -> Sell Order Filled  Prerequisite:   * None   Action:   * Create New Buy Order (price = Sell Order’s Price - Expected Profit Threshold)   Order List：  103→ Unfilled  102→ Filled  →101 Unfilled   1. 单元测试用例: WaitingSellingOrders02   产生过程：Waiting Selling Orders -> Market.AskPrice↓  Prerequisite:   * Market.AskPrice < Lowest Sell Order’s Price - Expected Profit Threshold * Lowest Sell Order’s Price > buyPriceLowlimit + Expected Profit Threshold \* 2   Action:   * Create Buy Order (price = Lowest Sell Order’s Price - Expected Profit Threshold \* 2)   Order List：  103→ Unfilled  →101 Unfilled (new order)   1. 单元测试用例: WaitingSellingOrder01   产生过程: Waiting Selling Order -> Market.AskPrice↓  Prerequisite:   * Market.AskPrice < Sell Order’s Price - Expected Profit Threshold   Action:   * Create Buy Order (price = Sell Order’s Price - Expected Profit Threshold \* 2)   Order List：  103→ Unfilled  →101 Unfilled (new order)   1. 单元测试用例: WaitingOrders01   产生过程: Waiting Orders -> Sell Order Filled  Prerequisite:   * Hanging Sell Order’s Number != 0   Action:   * Cancel Buy Order (CanceledByFilledSellOrder / PartCanceledByFilledSellOrder) * Create New Buy Order (price = Sell Order’s Price - Expected Profit Threshold) **(新挂出的买单必须和取消的买单在成交量上保持一致，因为取消掉的买单很可能已经成交了一部分)**   Order List：  104→ Unfilled  103→ Filled  →102 Unfilled (new order)  →101 CanceledByFilledSellOrder / PartCanceledByFilledSellOrder   1. 单元测试编号: WaitingOrdersAtLowLimitArea01   产生过程：Waiting Orders at Low Limit Area -> Sell Order Filled  Prerequisite:   * Hanging Sell Order’s Number != 0 **(在buyingPricePointNumber 配置大于等于5的情况下, Hanging Sell Order’s Number 必须大于0)**   Action:   * Cancel Buy Order (CanceledByFilledSellOrder / PartCanceledByFilledSellOrder) * Create New Buy Order (price = Sell Order’s Price - Expected Profit Threshold)   Order List：  103→Unfilled  102→ Filled  →101 Unfilled (new order)  →100 CanceledByFilledSellOrder |
| Waiting Orders at Low Limit Area | 1. 单元测试用例: WaitingSellingOrdersAtLowLimitArea02   产生过程：Waiting Selling Orders at Low Limit Area -> Sell Order Filled  Action：   * Create New Buy Order (price = buyPriceLowlimit)   Order List：   * [ [ [105→ Unfilled] 104→ Unfilled] 103→ Unfilled ] * 102→ Unfilled * 101→ Filled * →100 Unfilled (new order， price == buyPriceLowLimit)  1. 单元测试用例: WaitingSellingOrders03   产生过程：Waiting Selling Orders -> Market.AskPrice↓  Prerequisite:   * Lowest Selling Order’s Price == buyPriceLowlimit + Expected Profit threshold \* 2 * Market.AskPrice < Lowest Selling Order’s Price - Expected Profit threshold   Action：   * Create New Buy Order (price = buyPriceLowLimit)   Order List：  [ [ [105→ Unfilled] 104→ Unfilled] 103→ Unfilled ]  102→ Unfilled  →100 Unfilled (new order， price == buyPriceLowLimit)   1. 单元测试用例: WaitingSellingOrdersAtLowLimitArea03   产生过程：Waiting Selling Orders at Low Limit Area -> Market.AskPrice↓  Prerequisite:   * 不存在价格为buyPriceUpLimit + Expected Profit Threshold的卖单 * Market.AskPrice < Lowest Sell Order’s Price - Expected Profit Threshold   Action:   * DegradeBuyPriceLimit() * Create Sell Order(price = old buyPriceUpLimit + Expected Profit Threshold, state = CanceledByDecRange, Coin Number = 0) * Create Buy Order (price = new buyPriceLowLimit)   Order List：  105→Create Sell Order(state = CanceledByDecRange, Coin Number = 0)  [[[104→ Unfilled] 103→ Unfilled] 102→ Unfilled]  101→ Unfilled  →99 Unfilled (new order， price == new buyPriceUpLimit) |
| Waiting Orders at Up Limit Area | 1. 单元测试编号: WaitingSellingOrderAtUpLimitArea01   产生过程：Waiting Selling Order at Up Limit Area -> Market.AskPrice↓  Prerequisite:   * Market.AskPrice < Sell Order’s Price - Expected Profit Threshold   Action:   * Create Buy Order (price = buyPriceUpLimit - Expected Profit Threshold \* 2)   Order List：  105→ Unfilled ( price == buyPriceUpLimit + Expected Profit Threshold)  →103 Unfilled (new order)   1. 单元测试用例: WaitingSellingOrders04   产生过程：Waiting Selling Orders -> Sell Order Filled  Prerequisite   * Sell Order’s Price = buyPriceUpLimit   Action:   * Create Buy Order (price = Sell Order’s Price - Expected Profit Threshold)   Order List：  105→ Unfilled ( price == buyPriceUpLimit + Expected Profit Threshold)  104→ Filled ( price == buyPriceUpLimit)  →103 Unfilled (new order) |
| Waiting Buying Order | 1. 单元测试编号: NoOrder01   产生过程: No Order -> First Market Ask Price Information  Prerequisite:   * None   Action:   * Set Robot’s buyPriceLowLimit and buyPriceUpLimit * Create New Buy Order(price = Market’s Ask Price)   Order List：  →102 Unfilled (new order)   1. 单元测试用例: WaitingSellingOrder02   产生过程：Waiting Selling Order -> Sell Order Filled  Prerequisite:   * None   Action:   * Create Buy Order(Sell Order’s Price - Expected Profit Threshold)   Order List：  103→ Filled  →102 Unfilled (new order)   1. 单元测试编号: WaitingBuyingOrder01   产生过程: Waiting Buying Order -> Market.AskPrice↑  Prerequisite:   * Market.AskPrice > Buy Order’s Price + Expected Profit Threshold   Action:   * Cancel Buy Order (CanceledByIncAskPrice / PartCanceledByIncAskPrice) * Create New Buy Order(Old Buy Order’s Price + Expected Profit Threshold) **(新挂出的买单必须和取消的买单在成交量上保持一致，因为取消掉的买单很可能已经成交了一部分)**   Order List：  →103 Unfilled (new order)  →102 CanceledByIncAskPrice / PartCanceledByIncAskPrice   1. 单元测试用例: WaitingOrders02   产生过程：Waiting Orders -> Sell Order Filled  Prerequisite:   * Hanging Sell Order’s Number == 0   Action:   * Cancel Buy Order (CanceledByFilledSellOrder / PartCanceledByFilledSellOrder) * Create Buy Order(Sell Order’s Price - Expected Profit Threshold)   Order List：  103→ Filled  →102 Unfilled (new order)  →101 CanceledByFilledSellOrder / PartCanceledByFilledSellOrder   1. ~~单元测试编号: WaitingOrdersAtLowLimitArea04~~   ~~产生过程：Waiting Orders at Low Limit Area -> Sell Order Filled~~  ~~Prerequisite:~~   * ~~Hanging Sell Order’s Number == 0~~   ~~Action:~~   * ~~Cancel Buy Order (CanceledByFilledSellOrder / PartCanceledByFilledSellOrder)~~ * ~~Create New Buy Order (price = Sell Order’s Price - Expected Profit Threshold)~~   ~~Order List：~~  ~~102→ Filled~~  ~~→101 Unfilled (new order)~~  ~~→100 CanceledByFilledSellOrder~~  **在buyingPricePointNumber 配置大于等于5的情况下, 不可能出现这种情况。**  **注：Waiting Buying Order的状态下， 挂起的买单的买价只可能大于等于 Expected Price Range的中间值（本例中为102）。** |
| Waiting Buying Order at Up Limit Area | 1. 单元测试编号: WaitingBuyingOrder02   产生过程: Waiting Buying Order -> Market.AskPrice↑  Prerequisite:   * Market.AskPrice > Buy Order’s Price + Expected Profit threshold   Action：   * Cancel Buy Order (CanceledByIncAskPrice / PartCanceledByIncAskPrice) * Create New Buy Order (price = buyPriceUpLimit) **(新挂出的买单必须和取消的买单在成交量上保持一致，因为取消掉的买单很可能已经成交了一部分)**   Order List：  →104 Unfilled (new order， price == buyPriceUpLimit)  →103 CanceledByIncAskPrice / PartCanceledByIncAskPrice   1. 单元测试编号: WaitingBuyingOrderatUpLimitArea01   产生过程: Waiting Buying Order at Up Limit Area -> Market.AskPrice↑  Prerequisite:   * Market.AskPrice > Buy Order’s Price + Expected Profit Threshold   Action:   * UpgradeBuyPriceLimit() * Cancel Buy Order (CanceledByIncRange / PartCanceledByIncRange) * Create New Buy Order (price = buyPriceUpLimit) **(新挂出的买单必须和取消的买单在成交量上保持一致，因为取消掉的买单很可能已经成交了一部分)**   Order List：  →105 Unfilled (new order， price = new buyPriceUpLimit)  →104 CanceledByIncRange / PartCanceledByIncRange   1. 单元测试编号: WaitingSellingOrderAtUpLimitArea02   产生过程： Waiting Selling Order at Up Limit Area -> Sell Order Filled  Prerequisite:   * None   Action:   * Create New Buy Order (price = buyPriceUpLimit)   Order List：  105→ Filled  →104 Unfilled (new order， price = buyPriceUpLimit)   1. 单元测试编号: WaitingOrdersAtUpLimitArea01   产生过程： Waiting Orders At Up Limit Area -> Sell Order Filled  Prerequisite:   * None   Action:   * Cancel Buy Order (CanceledByFilledSellOrder / PartCanceledByFilledSellOrder) * Create New Buy Order (price = buyPriceUpLimit) **(新挂出的买单必须和取消的买单在成交量上保持一致，因为取消掉的买单很可能已经成交了一部分)**   Order List：  105→ Unfilled  →104 Unfilled (new order， price = buyPriceUpLimit)  →103 CanceledByFilledSellOrder / PartCanceledByFilledSellOrder |
| Waiting Selling Order | 1. 单元测试编号: WaitingBuyingOrder03   产生过程：Waiting Buying Order -> Buy Order Filled  Prerequisite:   * None   Action:   * Create Sell Order (price = Buy Order’s Price + Expected Profit Threshold)   Order List：  103→ Unfilled (new order)  →102 Filled |
|  |  |
| Waiting Selling Order at Up Limit Area | 1. 单元测试编号: WaitingBuyingOrderatUpLimitArea02   产生过程: Waiting Buying Order at Up Limit Area -> Buy Order Filled  Prerequisite:   * None   Action:   * Create Sell Order (price = Buy Order’s Price + Expected Profit Threshold)   Order List：  105→ Unfilled (new order)  →104 Filled ( price == buyPriceUpLimit) |
| Waiting Selling Orders | 1. 单元测试用例: WaitingOrders03   产生过程：Waiting Orders -> Buy Order Filled  Prerequisite:   * None   Action:   * Create Sell Order (price = Buy Order’s Price + Expected Profit Threshold)   Order List：  [ [ [105→ Unfilled] 104→ Unfilled] 103→ Unfilled ]  102→ Unfilled (new order)  →101 Filled   1. 单元测试用例: WaitingOrdersAtUpLimitArea02   产生过程： WaitingOrdersAtUpLimitArea -> Buy Order Filled  Prerequisite:   * None   Action:   * Create Sell Order (price = buyPriceUpLimit)   Order List：  105→ Unfilled  104→ Unfilled (new order, price = buyPriceUpLimit)  →103 Filled |
|  |  |
| Waiting Selling Orders at Low Limit Area | 1. 单元测试编号: WaitingOrdersAtLowLimitArea02   产生过程：Waiting Orders at Low Limit Area -> Buy Order Filled  Prerequisite:   * None   Action:   * Create Sell Order (price = Buy Order’s Price + Expected Profit Threshold)   Order List：  [ [ [ [105→ Unfilled] 104→ Unfilled] 103→ Unfilled ] 102→ Unfilled]  101→ Unfilled (new order)  →100Filled   1. 单元测试用例: WaitingSellingOrdersAtLowLimitArea01   产生过程：Waiting Selling Orders at Low Limit Area -> Market.AskPrice↓  Prerequisite:   * 存在价格为buyPriceUpLimit + Expected Profit Threshold的卖单 * Market.AskPrice < Lowest Sell Order’s Price - Expected Profit Threshold   Action:   * DegradeBuyPriceLimit() * Cancel upmost Sell Order (CanceledByDecBidPrice / PartCanceledByByDecBidPrice) * Create Buy Order (price = new buyPriceLowLimit + Expected Profit Threshold) **(新挂出的卖单必须和取消的卖单在成交量上保持一致，因为取消掉的买单很可能已经成交了一部分)**   Order List：  105→CanceledByDecBidPrice，PartCanceledByByDecBidPrice  104→ Unfilled  103→ Unfilled  102→ Unfilled  101→ Unfilled  100→ Unfilled (new order， price == new buyPriceUpLimit + Expected Profit threshold) |
|  |  |
| Resetting |  |

#### Pre-condition

* + 为了产生交易，robot 必须处于running状态。
  + robot 处于running状态时，用户不能修改与当前robot关联的账户的userid。
  + robot 处于running状态时，用户不能修改与当前robot关联的账户的可支配资金的金额。

#### Basic Flows

1. 用户输入start robot并命令return。
2. Client解析命令，并通过rpc通知到Server。
3. Server启动用例Ur-3：检查当前robot是否满足自动交易所需的必要条件。
4. 准备工作就绪，Server通过rpc返回异常信息给Client：robot启动成功。
5. Robot 检查并处理掉server停机期间状态产生变化的所有的订单。
6. Server 持续查询价格行情变化情况，和未成交的订单的完成情况。

假设当前的行情为BidPrice0，AskPrice0， robot在第5步按照如下的规则处理停机期间状态产生变化的订单：

#### Exceptional Flows

#### Basic Flows 第3步，Ur-3返回异常信息

1. Server终止当前用例的处理流程，Server给rpc返回robot启动失败的原因。

#### Alternative Flows

#### Basic Flows 第6步，Robot发现价格变化

假设 BuyPrice0 = math.floor(Ask Price)

robot将以下情况视为买入时机：

1. 当前robot没有与其相关联的未成交的订单。
2. 不存在未成交的卖单, 存在未成交的买单（假设其挂单价格为OderPrice0），Ask Price续上涨，BuyPrice0 - OderPrice0 >= Expected Profit threshold, BuyPrice0不高于Expected Price Range的上限。

发现买入时机之后，robot进行以下操作：

1. Server 以BuyPrice0为买价， 挂单买入CoinNumberX0个币。
2. 返回到Basic Flow的第5步继续执行。

#### Basic Flows 第6步，Robot发现有买单已成交

1. 假设价格刚成交的买单的价格为OderPrice0， Server设置：

BuyPrice0 = OderPrice0 - Expected Profit threshold

SellPrice0 = OderPrice0 + Expected Profit threshold

1. Server 以BuyPrice0为买价， 挂单买入CoinNumberX0个币。
2. Server 以SellPrice0为卖价， 挂单卖出CoinNumberX0个币。
3. 返回到Basic Flow的第5步继续执行。

#### Basic Flows 第6步，Robot发现有卖单已成交

1. 假设价格刚成交的买单的价格为OderPrice0， Server设置：

BuyPrice0 = OderPrice0 - Expected Profit threshold

1. Server 以BuyPrice0为买价， 挂单买入CoinNumberX0个币。
2. 返回到Basic Flow的第5步继续执行。

#### Basic Flows 第6步，User下发Reset Order指令

## Ur-3, Collision Detection

### Brief Description

本用例为Transaction的扩展用例，请查看Ur-2了解本用例运行的上下文信息。

### Pre-condition

* + robot 必须关联了正确数量的交易账户， 如：dedging-0要求robot关联交易2个交易账户。
  + 为了启动当前robot，与同一个账户同一种类型的币相关联的其它robot不能处于running状态。

### Basic Flows

1. Server启动本用例。
2. Server检查到数据库的连接是否正常。。
3. Server通过读取数据库的信息检查当前是否有其他处于活动的Server。
4. Server通过读取数据库的信息检查交易账户配置，交易账户状态。
5. Server通过rpc从交易平台获取账户信息，包括：可支配现金，可支配的币，未完成的挂单信息。Server对获取到的信息进行检查。

### Alternative Flows

无。

### Exceptional Flows

### Basic Flows 第2步，Server检测发现到数据库的连接不正常

1. Server终止当前用例的处理流程，给本用例的调用者返回异常信息：数据库连接异常。

### Basic Flows 第3步，Server检测发现其他处于活动状态的Server

1. 当前Server通过通知活动Server，请求接管交易权。
2. 如果能得到活动Server的确认信息，当前Server继续当前用例的处理流程。
3. 否则, Server终止当前用例的处理流程，给本用例的调用者返回异常信息：接管交易权失败。

### Basic Flows 第4步，Server发现当前有其他处于running状态的robot（设为robot-x）正在处理当前robot关联交易账户，并且robot-x和当前robot处理的币的类型一样

1. Server终止当前用例的处理流程，给本用例的调用者返回异常信息：robot-x正在管理当前账户x的当前币种（btc/ltc），同一个用户的同一种币只能被1个robot处理。

### Basic Flows 第5步，Server发现交易账户没有足够的可支配资金

1. 假设当前有其他robot（robot-x）正在处理当前交易账户的其他类型的币。 Server首先计算关联的交易账户上的可支配资金，其算法为：

可支配资金 = 未冻结的现金和币 – robot-x可支配资金余额

robot-x可支配资金余额 = robot-x可支配资金 - robot-x挂单买币的金额 - robot-x已近买入的币的金额

1. Server终止当前用例的处理流程，给本用例的调用者返回异常信息：交易账户X可支配资金余额不足。

# Functional and Data Requirements

## Functional Requirements

### 需求#: 1

需求类型: 9

事件/用况#: Ur-1

描述: Client在执行任何一条需要与Server通讯的CLI命令前，都需要先检到测Server的网络连接是否正常。如果无法连接到Server，CLI Client必须给用户提供准确的信息，以帮助用户解决问题。

理由: 只有在Client与Server的Tcp连接正常的情况下，Client才能正常工作。

来源: Liu Hao

验收标准: CLI命令因为Tcp连接异常而执行失败时， Client应反馈正确的帮助信息给用户。缺少Ip地址配置，Ip地址不通， Server无应答。

依赖关系: 无。

冲突: 无。

支持材料: Product Context in Section 8。

历史: 2016-08-08

### 需求#: 2

需求类型: 9

事件/用况#: Ur-1

描述: Client，服务端都必须正确的保存配置。用户输入的命令立即生效，但是save 命令前输入的配置才会被永久存储。软件重启后，最近一次save命令之后的配置不再有效。

理由: 大多时候，用户希望软件重启后仍然能使用重启前的配置来运行。还有些时候，用户很可能输入几个配置用于测试，测试完成后，用户希望软件重启动后能退回到之前报存过的配置。实现save命令，可以让用户自行决定是否要保存当前配置。

来源: Liu Hao

验收标准: 配置文件的内容始终和save命令执行时的内存中的配置一致。 每次软件重启， Client/服务端软件都会在初始化阶段读取配置文件，并按照配置文件的内容执行。

依赖关系: 无。

冲突: 无。

支持材料: Product Context in Section 8。

历史: 2016-08-08

### 需求#: 3

需求类型: 9

事件/用况#: Ur-1

描述: 用户可以在任何时候重启（或关闭然后再打开）Client和服务端。 重启Client不会对自动化交易产生任何影响。 重启服务端的效果相当于暂停一段时间的自动化交易，重启后，在网络正常的情况下所有robot能在2秒钟内恢复到重启前的状态，并继续自动化交易。如果重启的对象是服务端，需要确保所有的Client能得到正确的提示，以免用户误以为软件异常。

理由: 用户需要重启软件的功能，如为了测试的目的，或暂时关闭软件过段时间再打开。

来源: Liu Hao

验收标准: 用户可以通过关闭然后再打开的方式重启Client，也可以通过Client的命令重启服务端。

依赖关系: 无。

冲突: 无。

支持材料: Product Context in Section 8。

历史: 2016-08-08

### 需求#: 4

需求类型: 9

事件/用况#: Ur-1

描述: 服务端在接收到任何配置命令和重启服务端的命令后，都会先保存日志，记录谁因为什么原因在什么时候执行了什么命令以及执行结果，执行结果的缺省值为失败。除了重启命令，服务端需要在命令执行完成以后记录命令执行的结果。服务端初始化完成之后，查看日志，如果日志里面最后一跳命令是重启命令且时间和当前时间差异小于60面，服务端修改重启命令的日志，将其执行结果记录为成功。

理由: 记录用户操作命令的日志可以帮助将来软件出现异常时检查异常出现的原因。

来源: Liu Hao

验收标准: 用户在对服务端做了配置之后，在没有保存配置的情况下如果服务端crash，用户可以通过比对操作日志和之前导出的配置文件来得到crash时服务端运行时的配置。

依赖关系: 无。

冲突: 无。

支持材料: Product Context in Section 8。

历史: 2016-08-09

### 需求#: 5

需求类型: 9

事件/用况#: Ur-1

描述: robot 在做买入和卖出操作前，需要记录日志。操作完成后需要记录操作结果，成功或失败。

理由: 交易日志可以用来帮助分析软件的bug，也可以用来辅助分析交易算法。

来源: Liu Hao

验收标准: 交易日止必需和交易平台上看到的交易日志一致。

依赖关系: 无。

冲突: 无。

支持材料: Product Context in Section 8。

历史: 2016-08-09

### 需求#: 6

需求类型: 12

事件/用况#: Ur-2

描述: 任何时候, platform 返回不可恢复的错误信息时, 错误处理程序必须立即停止当前robot,并且为错误信息记录日志, 在用户清除了错误的根源后, robot可以立即被重启, 不需要人工修改数据库来恢复robot的运行环境。 其他不受关联的robot的运行不能受到影响.

理由: 记录日志可以帮助程序员检查错误, 也可以为下一步恢复robot的运行提供有用的帮助。

来源: Liu Hao

验收标准: 余额不足错误, cancel order失败, 交易过于频繁等的错误时, 错误处理程序应该按照本需求的描述来执行。 用户在排除了错误的原因后, 在不对数据库做任何修改的情况下,可以立即重启robot.

依赖关系: 无。

冲突: 无。

支持材料: Product Context in Section 8。

历史: 2016-10-21

## Data Requirements

Nonfunctional Requirements

# Look and Feel Requirements

# Usability and Humanity Requirements

# Performance Requirements

## Speed and Latency Requirements

## Safety-Critical Requirements

## Precision or Accuracy Requirements

## Reliability and Availability Requirements

## Robustness or Fault-Tolerance Requirements

## Capacity Requirements

## Scalability or Extensibility Requirements

## Longevity Requirements

# Operational and Environmental Requirements

无。

# Maintainability and Support Requirements

无。

# Security Requirements

无。

# Cultural and Political Requirements

无。

# 参考资料

无。