Dear Developers,

These are the major job request for this project:

1. Modify my order process (OppositeDirection and SameDirection).
2. Modify my SameDirection function
3. Add Automatic Exit function
4. Add Continuous Trading Function
5. Add Flatten Everting function
6. Add Default Strategy Setting function

**>>>>>>>>>>>> Fixing My Order Process.**

I am reaching out to request your expertise in refining a strategy I have developed for NinjaTrader 8. The focus of this enhancement is primarily on two methods: **OnBarUpdate** and **OnExecutionUpdate**. I have attached my scource code for you to have an idea of how my script works and what I expected you to work on.

I intend to provide a detailed explanation below regarding the specific modifications needed for the script. Your assistance in optimizing these functions will be greatly appreciated.

My Script has 2 directional order executions:

1. **OppositeDirection**
2. **SameDirection**
   1. **OppositeDirection=** This section works more like a close and reverse function. i.e. if a buy is opened and the price falls to a particular level i.e. Recovery Gap, it closes the buy and opens a sell based on a set quantity sequence.
   2. **SameDirection:** This section works more like a stoploss re-entry function. i.e. if a buy is opened and the price falls to a particular level i.e. Recovery Gap, it closes the buy and waits for the price to reverse to the original entry to open a buy based on a set quantity sequence.

**The Problem with my Script**

a. **Issue with BuyStop Orders**: I’ve frequently encountered an issue that prompts me to halt my script’s operation. The error message displayed is: “Rejected: The BuyStop price cannot be less than the current market price.” This typically happens during market openings when volatility is high, causing the market price to jump below the predetermined BuyStop level. As a result, the script attempts to place a BuyStop order at a price that’s already been surpassed by the market, leading to the order’s rejection and, unfortunately, financial losses for me.

b. **Issue with SellStop Orders**: Similarly, I face a challenge with SellStop orders. The error message I receive is: “Rejected: The SellStop price cannot be higher than the current market price.” This occurs during periods of significant volatility at market open, where the price leaps above the SellStop level. Consequently, the script tries to initiate a SellStop order at a price that the market has already exceeded, resulting in the order being rejected and causing considerable losses for me.

I hope this revision clarifies the issues you’ve been experiencing. If you need further assistance in addressing these problems within your script, please let me know.

**Recommended Solution to my Script**

To achieve this, you have to do the following:

**Define the following functions:**

* ShortMarketOrderPlaceHolderLine() : This is a function that draws a horizontal line below the Long Entry price. It’s major function is to serve as a reference point for a **price placeholder** without initiating a short stop-order entry.
* LongMarketOrderPlaceHolderLine(): This is a function that marks a horizontal line above the Short Entry price. It’s major function is to serve as a reference point for a **price placeholder** without initiating a long stop-order entry.

**>>>>>>>>>>>> IF THE OPPOSITEDIRECTION is Selected in the Trade Direction**

Step 1: Enter Long

Recovery Gap

Step 3: Long Actions

Step 2: Short Actions

**/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////**

Assuming an initial Long Entry, however the price us falling, meaning it a losing trade; so this will trigger the following actions:

**Initial Trade Entry Actions:**

After the execution of a Long Entry order:

* + Do not set a stop loss.
  + Do not set a short stop order entry.
  + Set a take profit level/limit.
  + Draw two horizontal lines: one at the Long entry point and the other below the Long entry point using the ShortMarketOrderPlaceHolderLine() function as a **price placeholder,** which is defined to draw a line, and hold a price value. The space between the long entry and the ShortMarketOrderPlaceHolderLine() is the RecoverGap already existing in the script.
  + **DO NOT** use ShortMarketOrderPlaceHolderLine()  to set a short stop order entry. NO, NO, NO!!! Only as a **price placeholder,** to draw a line, and hold a price value.

**Labeling:**

Whenever ShortMarketOrderPlaceHolderLine() or LongMarketOrderPlaceHolderLine(): is drawn, label the line as **“Active Trade”** so as to indicate it’s holding a placeholder price value.

**Response to Losing Long Entry:**

* If the Long Entry is losing and the price falls to <= ShortMarketOrderPlaceHolderLine() :
  + Execute a Short Entry Market Order with a quantity equal to the sum of the current trade quantity and the next trade quantity. **(See examples 1 and 2 below)**
  + Only Use Market Order at placeholder, NO STOPS for entry
  + Only Use Market Order at placeholder, NO STOPS for entry
  + Only Use Market Order at placeholder, NO STOPS for entry
  + Set a new short take profit level.
  + Draw a LongMarketOrderPlaceHolderLine(): above the Short entry as a new placeholder.
  + Remove the previous Long Take Profit line.

**Example 1**

This mean if the existing/current losing long quantity = 5

And the next quantity in the iteration = 5,

New total quantity = (5 + 5) = 10 Therefore, enter next trade with this new TOTAL QUANTITY.

This means there will be a **(AS NET POSITION ADJUSTMENT)** of 5 long positions

**Example 2**

This mean if the existing/current losing long quantity = 2

And the next quantity in the iteration = 4, due to increment

New total quantity = (4 + 2) = 6 Therefore, enter next trade with this new TOTAL QUANTITY.

This means there will be a **(AS NET POSITION ADJUSTMENT)** of 4 long positions.

**Handling a Failing Short Entry:**

* If the Short Entry is losing and the price rises to >= LongMarketOrderPlaceHolderLine() :
  + Execute a Long Entry Market Order with a quantity equal to the sum of the current trade quantity and the next trade quantity. **(See examples 1 and 2 below)**
  + Only Use Market Order at placeholder, NO STOPS for entry
  + Only Use Market Order at placeholder, NO STOPS for entry
  + Only Use Market Order at placeholder, NO STOPS for entry
  + Set a new long take profit level.
  + Draw a ShortMarketOrderPlaceHolderLine() below the Long entry as a new placeholder.
  + Remove the previous Short Take Profit line.

**Example 1**

This mean if the existing/current losing short quantity = 10

And the next quantity in the iteration = 20, due to increment

New total quantity = (20 + 10) = 30. Therefore, enter next trade with this new TOTAL QUANTITY.

This means there will be a **(AS NET POSITION ADJUSTMENT)** of 20 long positions

**Example 2**

This mean if the existing/current losing long quantity = 7

And the next quantity in the iteration = 10, due to increment

New total quantity = (10 + 7) = 17 Therefore, enter next trade with this new TOTAL QUANTITY.

This means there will be a **(AS NET POSITION ADJUSTMENT)** of 10 long positions.

**Continuation:**

* Repeat the above process as necessary, adjusting for market conditions and trade outcomes.
* Apply the same logic for a Short Entry scenario.

**>>>>>>>>>>>> IF THE SAMEDIRECTION is Selected in the Trade Direction**

Step 1: Enter Long

Recovery Gap

Step 3: Long Actions

**Step 2: Exit Loss with Market Order**

**STEP 1: Handling a Long Entry Without Stop Orders.**

After executing a Long Entry in anticipation of a price increase:

* Do not set a stop-loss order.
* Do not initiate a short stop order entry. Instead, take the following actions:
  + Set a take-profit level/limit at the time of Long Entry execution.
  + Draw a horizontal line at the entry price and another at the calculated ShortMarketOrderPlaceHolderLine() level.
  + Label any line drawn as **“Active Trade”** to indicate it’s currently in use or reserved.

**STEP 2:** Adjusting Strategy When Price Moves Against the Long Entry

If the market price drops to or below the ShortMarketOrderPlaceHolderLine() level:

* Cancel the existing Long trade.
* Do not place any buy-stop orders. Wait for the market price to return to the original Long Entry level indicated by LongMarketOrderPlaceHolderLine(). If the price recovers to or exceeds this level:
* Execute a new Long trade at the market price, adhering to the predefined parameters.

STEP 3: Applying the Same Logic to Short Trades Implement the same procedures for Short Entries when trading in the same direction.

**>>>>>>>>>>>> “SAMEDIRECTION” FUNCTION MODIFICATION TO A NEW ENTRY LEVEL**

1. This Modification is for the “Trade Direction Input”
2. This Modification is **ONLY FOR THE “SAMEDIRECTION” OPTION** of the trade direction.

**Variables and Conditions**

Condition1 = [(Bar2\_isBullish && Bar1\_isBearish) && (Bar\_2\_Low > Bar\_1\_Close)]

Condition2 = [(Bar3\_isBullish && Bar2\_Bearish && Bar1\_isBearish) && (Bar\_3\_Low > Bar\_1\_Close)]

Condition3 = [(Bar2\_Bearish && Bar1\_isBullish) && (Bar\_2\_High < Bar\_1\_Close)]

Condition4 = [(Bar3\_Bearish && Bar2\_isBullish && Bar1\_isBullish) && (Bar\_3\_High < Bar\_1\_Close)]

Here’s what each condition means:

* **Condition1:** A bullish pattern is present in the second-to-last bar, and a bearish pattern in the last bar, with the low of the second-to-last bar being higher than the close of the last bar.
* **Condition2:** A sequence of a bullish pattern followed by two bearish patterns over the last three bars, with the low of the third-to-last bar being higher than the close of the last bar.
* **Condition3:** A bearish pattern is present in the second-to-last bar, and a bullish pattern in the last bar, with the high of the second-to-last bar being lower than the close of the last bar.
* **Condition4:** A sequence of a bearish pattern followed by two bullish patterns over the last three bars, with the high of the third-to-last bar being lower than the close of the last bar.

1. **SELL CONDITION Opportunity 1 (Current Code in Strategy)**
   * + 1. This is the current Short Position code existing in the strategy.
2. **SELL CONDITION Opportunity 2**
3. Trade Direction Logic for ‘SameDirection’ Selection:
   1. If the Trade Direction dropdown has the SameDirection option selected.
   2. Upon opening a Sell trade:
      1. Monitor the market price. If it rises and either Condition1 or Condition2  becomes True, and the price is above the initial Buy entry price, then:
         1. Adjust the Sell re-entry level to match the current market price.
         2. Execute ONLY Market Sell Order at this new level.
         3. Execute ONLY Market Sell Order at this new level.
         4. Execute ONLY Market Sell Order at this new level.
         5. Apply the input variables (arithmetic increment, recovery gap level, take profit level, and stop loss level) to this new re-entry level, maintaining the standard procedure.
      2. Repeat this process:
         1. Continue until the take profit target is met or the trade is manually closed.
   3. Note:
      1. A new Sell re-entry level is set only if all conditions are satisfied and the current price is more than the previous Sell entry price.
      2. If not, maintain the previous entry level as the re-entry point.

A graph with red and green lines

Description automatically generated

1. **BUY CONDITION Opportunity 1 (Current Code in Strategy)**
   * + 1. This is the current Long Position code existing in the strategy.
2. **BUY CONDITION Opportunity 2**
3. Trade Direction Logic for ‘SameDirection’ Selection:
   1. If the Trade Direction dropdown has the SameDirection option selected.
   2. Upon opening a Buy trade:
      1. Monitor the market price. If it drops and either Condition3 or Condition4 becomes True, and the price is below the initial Buy entry price, then:
         1. Adjust the Buy re-entry level to match the current market price.
         2. Execute ONLY Market Buy Order at this new level.
         3. Execute ONLY Market Buy Order at this new level.
         4. Execute ONLY Market Buy Order at this new level.
         5. Apply the input variables (arithmetic increment, recovery gap level, take profit level, and stop loss level) to this new re-entry level, maintaining the standard procedure.
      2. Repeat this process:
         1. Continue until the take profit target is met or the trade is manually closed.
   3. Note:
      1. A new Buy re-entry level is set only if all conditions are satisfied and the current price is less than the previous Buy entry price.
      2. If not, maintain the previous entry level as the re-entry point.

A graph of a stock market

Description automatically generated with medium confidence

**>>>>>>>>>>>> AUTOMATIC EXIT FUNCTION**

* + Define a variable called **HighestTotalPnL()**
    - This function is only used to store the highest profit of the day.
    - If the CurrentPnL is higher than the HighestTotalPnL() stored, we update HighestTotalPnL() with the new value.

**SELL EXIT 1**

* If the position is in profit and the current price has not yet hit the Sell Take Profit level
  + If ((Bar3\_Bearish && Bar2\_isBullish && Bar1\_isBullish && (Bar1\_Close > Bar2\_Close) && (Bar2\_Close < Bar3\_Open)) && (CurrentPnL > **HighestTotalPnL())**
    - Then Automatically exit the trade if (CurrentPnL > **HighestTotalPnL())**
    - Label “Sell Exited Here” and draw horizontal line on the chart at the exit
    - Note this condition only triggers if the trade is **IN PROFIT** -

A graph showing a price of a stock market

Description automatically generated with medium confidence

**SELL EXIT 2**

* If the position is in profit and the current price has not yet hit the Sell Take Profit level
  + (Bar2\_Bearish && Bar1\_isBullish && Bar1\_Close > Bar2\_Open) && (CurrentPnL > **HighestTotalPnL())**
    - Then Automatically exit the trade if (CurrentPnL > **HighestTotalPnL())**
    - Label “Sell Exited Here” and draw horizontal line on the chart at the exit
    - Note this condition only triggers if the trade is **IN PROFIT** -

A screenshot of a graph

Description automatically generated

**BUY EXIT 1**

* If the position is in profit and the current price has not yet hit the Buy Take Profit level
  + If ((Bar3\_isBullish && Bar2\_isBearish && Bar1\_isBearish && (Bar1\_Close < Bar2\_Close) && (Bar2\_Close > Bar3\_Open) && (CurrentPnL > **HighestTotalPnL())**
    - Then Automatically exit the trade if (CurrentPnL > **HighestTotalPnL())**
    - Label “Buy Exited Here” and draw horizontal line on the chart at the exit
    - Note this condition only triggers if the trade is **IN PROFIT** -

A graph showing a price of stock

Description automatically generated with medium confidence

**BUY EXIT 2**

* If the position is in profit and the current price has not yet hit the Buy Take Profit level
  + (Bar2\_isBullish && Bar1\_isBearish && Bar1\_Close < Bar2\_Open) && (CurrentPnL > **HighestTotalPnL())**
    - Then Automatically exit the trade if (CurrentPnL > **HighestTotalPnL())**
    - Label “Buy Exited Here” and draw horizontal line on the chart at the exit
* Note this condition only triggers if the trade is **IN PROFIT** -

A graph of a candle stick graph

Description automatically generated with medium confidence

**>>>>>>>>>>>> CONTINOUS TRADING FUNCTION**

1. Add an input named (Trading Mode) with the following characteristics:
   * The input has a dropdown menu offering two options:
     + Manual
     + **Automated ( Make this the default)**
   * When Manual is chosen, the script should operate according to the existing code.
   * When Automated is selected i.e the DEFAULT Mode:
     + If the Start button is activated and either the **profit target is reached or the Automated Exit Function is triggered, do the following**:
       - Determine whether the new trade to open is a LongEntry or ShortEntry based on the existing entry condition in the script:
         1. If it is a LongEntry

Then immediately enter an Automatic Long Position with the original parameters from the initial trade

* + - * 1. If it is a ShortEntry

Then immediately enter an Automatic Short Position with the original parameters from the initial trade

* + - * 1. Continue opening trades if any of the conditions above are met, without the need to click on the “Start Button”.
    - If the Stop button is pressed:
      * All open trades should be closed immediately without restarting.

**>>>>>>>>>>>> FLATTEN EVERYTHING FUNCTION**

* + Add a third button beside the Stop called “FLATTEN EVERYTHING”. This will close all the active trade in the POSTION TAB running in the account.

**>>>>>>>>>>>> DEFAULT STRATEGY SETTING FUNCTION**

There are certain things I kep doing each time I am about to use my strategy which, I don’t want to keep doing?

* + Selecting the account I want to trade on.

I don’t want to keep doing these process, that is what machines, scripts and robot were invented for. So I want you to code the following:

* + **Account:** Each time I want to use or install the strategy on the chart, the account should AUTOMATICALLY SELECT THE ACCOUNT CHOSEN ON THE CHART to trade the instrument.