https://www.garot.com/trading/

01 – Introduction

This is the informal video blog.

This fifth video of this VLOG series concerns additions to the code to pass the validation testers when I uploaded the code to the MQL5.com's CodeBase. These code additions are important in any EA, so I don't want to give the impression that I'm doing the bare minimum *just* to pass the tests. There are many facets to any EA that I haven't covered in this series because instruction should—I think—start with the simplest concepts first.

One point I will make is that if you download the file from the MQL5.com CodeBase instead of from my website, instead of from my website, you will see that it was run through the MetaEditor Styler. To me, this is just ugly—I prefer to use tabs for indentation with a tabstop of 4 spaces. I understand and appreciate that MetaQuotes wants consistency in their CodeBase, so I am happy to go along with their request.

Anyway, I call this part: **Passing The MQL5 Validation Tests**

(page 2)

In this VLOG we will:

- Remove the netting/hedging mode check
- Check STOPS level
- Normalize the Lot Size
- Check the amount of free margin

Now let's take a look at the additions to the code.

02 – Code View in WinMerge

To show the additions to the code from version 04 to 05, I will do a quick walk-thru summary using WinMerge. After this summary, I will go through the added code in more detail in MetaEditor.

Remember that in WinMerge changed code is highlighted yellow on both sides, and additions (new code) is gray on the left side and yellow on the right side.

Toward the top we note the addition of the AccountInfo MQL5 standard library, which we will use to determine the amount of free margin.

At the top of the OnInit() event, I removed the code to determine if our brokerage account is netting or hedging since, for this EA, it doesn't matter.

The biggest addition to the code was the addition of the CheckStopsLevel() function. This ensures that the distance between the stop loss and price in question meets the brokerage minimum requirement.

This next minor addition is a check to not place a trade if the new SL and existing SL are the same. This removes a few error messages in the log file.

The FixLotSize() function ensures that the minimum, maximum, and granularity of the broker requirements for the lot size occur.

In the OrderStop() function, you will note that volume is no longer a constant.

- Volume is passed through FixLotSize() to ensure compliance with broker settings.
- The available free margin is checked.
- Finally, we ensure the distance from the trigger price to the SL meets the broker requirements.

Finally, in the PositionModify() function, I check the SL stops level again.

03 – Code View in MetaEditor

I'm back in MetaEditor, and we're looking at the code for Rapid Doji EA 05.

I've already mentioned the addition of the AccountInfo standard library.

I've also mentioned the removal of the check for netting vs. hedging.

So let's jump right to the CheckStopsLevel() function. This whole function revolves around the SYMBOL_TRADE_STOPS_LEVEL setting on the brokerage server. The documentation describes this setting as:

> Minimal indention in points from the current close price to place Stop orders

This means two things to us:

1. For our pending orders, it is the minimum distance between the target price and the stop loss.
2. For our modify orders (for the trailing SL), it is the minimum distance between price (ask/bid) and the stop loss.

We first have a bunch of short-hand constants to make the code easier to read.

Then I define a variable called distancePts, which I immediately set to a ridiculous value to trap for errors in my coding.

Let's look at ORDER_TYPE_BUY_STOP first. First I check if the trigger price is less than the ask. If it is, I give a warning message and exit. This is a lunacy test, and it helps check for coding errors prior to calling this function.

The next code block calculates the distance between the trigger price and the SL. If it isn't as big as the server demands, we get out.

The code for ORDER_TYPE_SELL_STOP does exactly the same check with a SL on the other side of the trigger price.

Let's take a look at the check for ORDER_TYPE_BUY. This will be true when issuing the ModifyOrder() function on an existing position. In this case we check the distance between the current price, in this case the bid, and the SL. It's as simple as that.

Now let's look at the additions to the function FixedTrailingStop(). As I scroll toward the bottom, I find where I have added another code block. This is a simple test to ensure that the new SL and the existing SL are not essentially identical. If the MathAbs() of the difference is less than an epsilon, we return. This will save a few error messages in your log file.

Let's take a look at the FixLotSize() function. Basically this normalizes the lot size to a granularity acceptable to the server. For example, the server may have a SYMBOL_VOLUME_STEP of 0.01, and in our calculations we might land on 0.007, which the server won't accept. So, we normalize it.

Then we pass the normalized value through checks to ensure compliance of the minimum and maximum

lot sizes. In this case, if either extreme is breached, we set the lot size to that level.

Now let's jump down to the OrderStop() function. We do a few new things here:

1. Normalize the lot size and limit to the min or max allowable by the server settings.
2. Check the free margin using the CAccountInfo standard library to ensure we have enough money to place the trade. Note the ternary operator used to convert the order_type to one the FreeMarginCheck() function accepts.
3. Call the CheckStopsLevel() function to ensure the distance between the trigger price and stop loss is acceptable.

Finally, we'll scroll just a little more to PositionModify() where we once again call the CheckStopsLevel() function to ensure the distance between the current price and stop loss is acceptable.

04 – Wrapping up

(read from slide)

Final Notes (not in video)

1.
2.