ProRealTime

PROGRAMMING GUIDE

Trading Systems (ProBacktest & ProOrder)

V 6.1.0 - 20251117

Table of contents

Introduction to programming trading systems with ProRealTime

The trading system tools in ProRealTime let you create personalized investment strategies via programming or assisted creation (no programming required).

There are two ways to run your systems:

- As backtests to test their performance over historical data of a security.
- As automatic trading systems: orders are placed in real-time from a trading or PaperTrading account.

Trading system programming uses the ProBuilder programming language that is also used to write indicators in ProRealTime with additional instructions that apply only to programming trading systems.

Trading system programs can include instructions to take positions, set stops and risk management of each trading system based on personalized conditions such as:

- Predefined indicators in the workstation or indicators that you have programmed.
- Past performance of your trading system.
- Your trading system's latest orders.

The results of a trading system are presented in the following format:

- The equity curve shows the status of a system's gains and losses on a particular instrument.
- The positions histogram shows the positions of the system (green bars for buying positions and red bars for short selling positions). No bar is shown if there is no position for a particular time period.
- The detailed report in the application indicates the statistics of your system for the security over the time period it was backtested or executed.

In backtesting mode, it is also possible to optimize variables of your trading system to see which values give the best results over the period of history you are examining.

In automatic trading mode, orders placed by your trading systems appear in your portfolio and order list. The portfolio is updated with the gains and losses made.

This manual is organized in the following manner:

- The first part explains how to access the trading system creation features.
- The second part explains the ProBuilder instructions used to program systems.
- The third part explains how to backtest trading systems with ProBacktest.
- The fourth part explains how to execute a trading system automatically.
- The appendices at the end show how trading system results are displayed and also provide some example programs as well as the glossary for the ProBuilder language.

For beginning users, we advise you to first watch the video "**Backtest your strategies without writing a single line of code**".

The ideas expressed in this manual are only to help you learn to write trading systems and test your own ideas. They are not investment advice in any case.

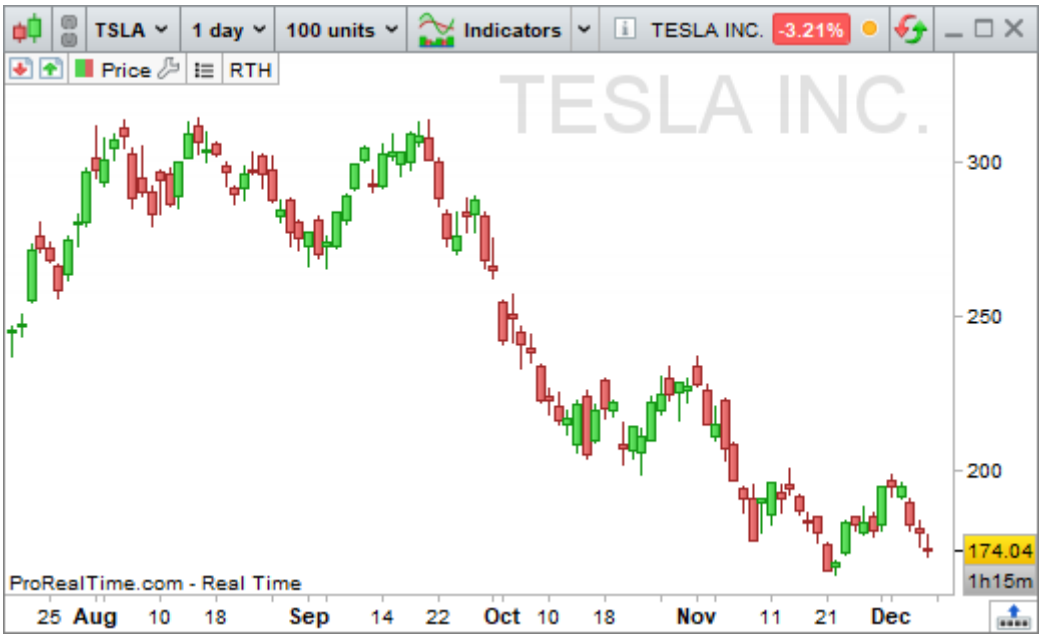
If you have any further questions about how the ProBacktest works, you can ask them to our ProRealTime community on the **ProRealCode forum**, where you will also find **online documentation** with many examples.

We wish you the greatest success in your trading and hope you will enjoy the manual.

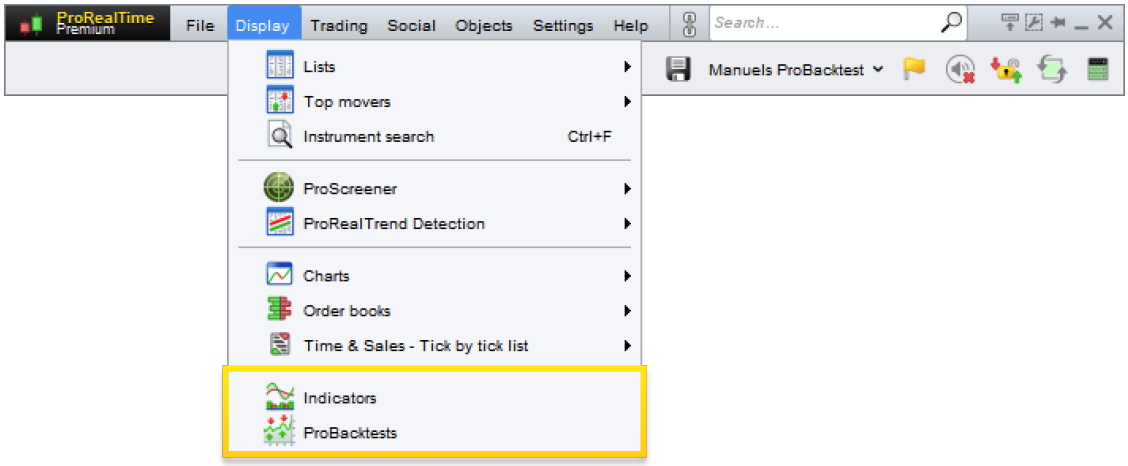
Introduction

Accessing trading system programming

The zone for trading system creation may be accessed with the "Indicators" button located at the top[1] of each of your ProRealTime charts.



You can also access it directly from the "Display" menu[1]:



By default, the "Indicators" tab will be selected. Select the second tab "Backtesting & Automatic trading". You will then be able to:

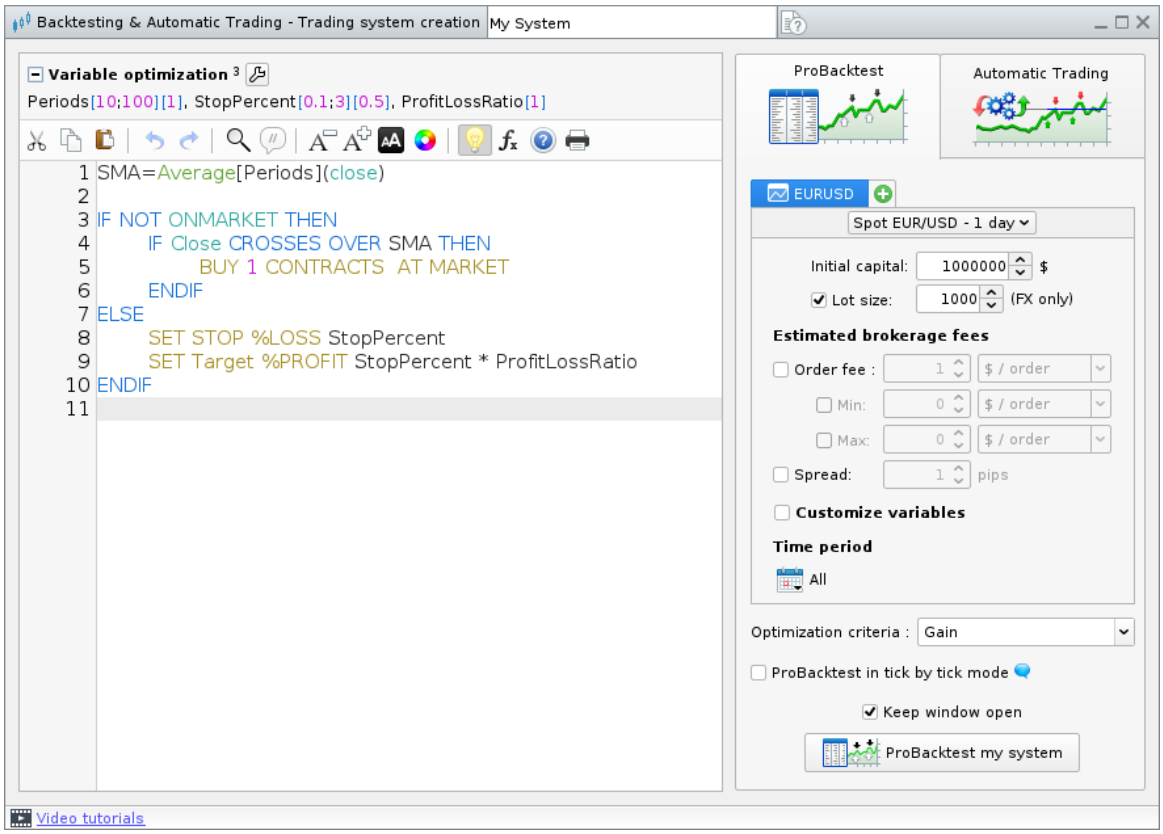
- Access the list of existing trading systems (personal or predefined)
- Create a new trading system or apply an existing system to any security
- Modify or delete an existing trading system
- Import or export trading systems
- Add codes that you purchased on the **ProRealCode Marketplace**

[1] *Since version 12 of ProRealTime*

Trading system creation window

The trading system creation window is composed of two main zones:

- The creation zone (assisted creation or creation by programming) appears on the left.
- The strategy application zone appears on the right. This includes a ProBacktest tab to backtest a trading system with historical data and a ProOrder AutoTrading tab to automatically execute a trading system. The options in the ProBacktest tab are detailed in section **#5.ProBacktest - trading system backtesting** of this manual.



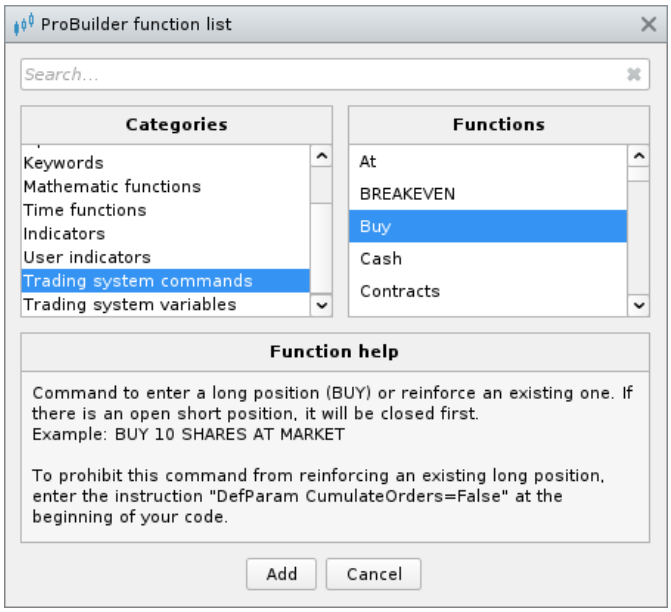
The creation zone allows you to:

- Program the trading system using the text editor
- Use the “insert function” button to open a new window with a list of ProBuilder and trading system commands separated into different categories that give you contextual help while programming. You can see a help text related to the command or function selected in the lower part of the window.

Example:

Let's use the function library by clicking on “insert function”.

Choose the section “Trading system commands”, click “**BUY**” then click “Add”. The command will be added to your program.



Now let’s try to create a full line of code. Suppose we want to buy 10 shares at market.

Proceed as above to find the functions "`SHARES`", "`AT`" and "`MARKET`" (separating each word with a space). Specify between "`BUY`" and "`SHARES`" the number to buy (10).

You will then obtain the instruction "`BUY 10 SHARES AT MARKET`" which is an instruction to buy 10 shares, lots or contracts at market price. The next section presents all the instructions which are available to program trading systems.

To see some examples of complete trading systems, check Annex B at the end of this manual.

Programming trading systems

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

Programming in ProBuilder language

Before learning to program trading strategies, we advise you to first read the **manual concerning how to program indicators in ProBuilder**.

ProBuilder is the programming language used in the workstation. It is very simple to use and offers many possibilities. Some important principles of ProBuilder are:

- The calculations are done at the end of each bar. ProBuilder programs including trading systems and all of their functions are evaluated at the end of each bar from the beginning to the end.
- All instructions to place orders are triggered after calculations on the current candlestick are finished (meaning the orders will be executed at the open of the next candlestick at the earliest).

The rest of this section of the manual focuses on the additional ProBuilder commands that are specific to trading systems (and are not discussed in the indicator programming manual linked above).

Entering and exiting the market

Different instructions are used depending on the type of position:

Long positions:

- `BUY` is an enter long position instruction (Buy financial instruments)
- `SELL` is an exit long position instruction (Sell financial instruments that you own)

`BUY` allows you to open a long position on the market or add to an existing open position. It is associated with the instruction `SELL` which allows you to close or partially close a position. The instruction `SELL` has no effect if there is no long position open.

Short positions:

- `SELLSHORT` is an instruction to enter a short position (short sell financial instruments)
- `EXITSHORT` is an instruction to exit a short position (buy back shorted financial instruments)

These two instructions work similarly to "`BUY`" and "`SELL`". The instruction `EXITSHORT` has no effect if there is no short position open.

It is not possible to take a long and short position at the same time on the same security. In practice, this means it is also possible to close a long position with the `SELLSHORT` command or close a short position with the `BUY` command.

Note: In automatic trading, remember to check that the maximum position size you set when starting the trading system is not smaller than the position size your trading system will try to open (or strengthen). Otherwise, the order will not be transmitted and therefore will not be executed.

Each of these commands may be followed by quantity and at type instructions as shown below:

```
<Order> <Quantity> AT <type>
```

Example:

```
BUY 1000 CASH AT MARKET or SELL 1 SHARE AT 1.56 LIMIT
```

Quantity:

There are two ways to define the quantity:

- **SHARES** corresponds to one unit of the instrument. "**1 Share**" can represent 1 stock, 1 futures contract, or one Forex contract. **SHARES** can be used interchangeably with **SHARE**, **CONTRACT**, **CONTRACTS**, **PERPOINT**, **LOT**, or **LOTS** for any type of instrument. In the case of Forex, the quantity bought will be multiplied by the size of one lot. If no quantity is specified, the following default values are used:
  - 1 unit for a position entry (Ex: **BUY AT MARKET**, buys a quantity of "1" at market price)
  - The entire quantity of a position for an exit (Ex: **SELL AT MARKET** sells the entire long position)
- **CASH** corresponds to a cash amount (like € or $) and this instruction can only be used for buying or selling shares. The quantity of the order will be calculated at the close of the bar and rounded down by default. Brokerage fees are not taken into account when calculating the quantity to buy or sell in cash.

Example: `BUY 1000 CASH AT MARKET`

The instruction **ROUNDEDUP** can be used to round this quantity up and **ROUNDEDDOWN** can be used to round this quantity down.

Example: `BUY 1000 CASH ROUNDEDUP AT MARKET`

Modes

Three modes are available for these types of orders:

`AT MARKET`: The order will be placed at market price at the open of the next bar.

Example: `BUY 1 SHARE AT MARKET`

`AT <price> LIMIT`: A limit order will be placed at the indicated price

`AT <price> STOP`: A stop order will be placed at the indicated price

Example: `BUY 1 SHARE AT 10.5 LIMIT`

Limit and stop orders with specific levels are valid for one bar by default starting at the open of the next bar. They are canceled if not executed.

These orders are different from protection stop and target orders (see next section) which are linked to an open position and valid until the close of that position.

Some orders may be treated as market orders in the following conditions:

- If a `BUY <quantity> at <price> LIMIT` order is placed above the market price, the order is treated as a market order.
- If a `BUY <quantity> at <price> STOP` order is placed below the market price, the order is treated as a market order.
- If an order to `SELLSHORT <quantity> at <price> LIMIT` order is placed below the market price, the order is treated as a market order.
- If an order to `SELLSHORT <quantity> at <price> STOP` order is placed above the market price the order is treated as a market order.

Example:

The following program buys 1 share at market price if the RSI is oversold (RSI < 30) and price is under the lower Bollinger band. It sells if the RSI is overbought (RSI > 70) and price is above the upper Bollinger band.

```
MyRSI = RSI[14](Close)
MyBollingerDown = BollingerDown[25](Close)
MyBollingerUp = BollingerUp[25](Close)
IF MyRSI < 30 AND Close < MyBollingerDown THEN
   BUY 1 SHARE AT MARKET
ENDIF
IF MyRSI > 70 AND Close > MyBollingerUp THEN
   SELL AT MARKET
ENDIF
```

You can set the validity length of limit and stop orders.

The following example shows how it is possible to create a limit order with a validity set to a specific number of bars by using variables. The code places a buy limit order at the close price of the bar on which a moving-average crossover occurred. This limit is valid for 10 bars after the bar of the crossing. If it is not executed during these 10 bars, it is cancelled.

Example:

```
// Definition of the validity length of the order
ONCE NbBarLimit = 10
MM20 = Average[20](Close)
MM50 = Average[50](Close)
// If MM20 crosses over MM50, we define 2 variables "MyLimitBuy" and "MyIndex" containing
the close price at that time and the index of the bar of the cross.
IF MM20 CROSSES OVER MM50 THEN
   MyLimitBuy = Close
   MyIndex = BarIndex
ENDIF

IF BarIndex >= MyIndex + NbBarLimit THEN
   MyLimitBuy = 0
ENDIF

// Place an order at the price MyLimitBuy valid as long as this variable is greater than
0 and we are not in a long position.
// Remember: MyLimitBuy is greater than 0 for the 10 bars after the bar of the crossing.
IF MyLimitBuy > 0 AND NOT LongOnMarket THEN
   BUY 1 SHARES AT MyLimitBuy LIMIT
ENDIF
```

In case the order was not executed, it is possible to replace the expired buy limit order with a buy at market price order. This could be done by adding the following code to the previous one:

```
IF BarIndex = MyIndex + NbBarLimit AND NOT LongOnMarket THEN
   BUY 1 SHARES AT MARKET
ENDIF
```

Stops and targets

You can also define profit targets and protection stops. The syntax is as follows:

```
SET STOP <type> <value> or SET TARGET <type> <value>
```

Example:

```
SET STOP %LOSS 2 // Set a stop loss of 2%
```

Each instruction is detailed in the next paragraphs.

Note the difference between **STOP** commands:

- **AT <price> STOP** is used to ENTER a position. This order is valid during one bar by default.
- **SET STOP LOSS <price>**, is a protection stop used to EXIT a position. This order is valid until the position is closed.

Protection stops

Protection stops let us limit the losses of a position. They can be defined in relative or absolute terms:

SET STOP LOSS x : Set a stop loss to close the position **x** units from entry price.

SET STOP pLOSS x : Set a stop loss to close the position **x** points from entry price.

SET STOP %LOSS x : Set a stop loss to close the position when the loss reaches **x**%, brokerage fees not included.

SET STOP $LOSS x : Set a stop loss to close the position of **x** €,$ (currency of the instrument), brokerage fees not included.

Protective stops can also be placed at the entry price of the position, allowing you to secure a position by preventing loss if the position reaches the break even price (excluding brokerage fees):

SET STOP BREAKEVEN: A protective stop is placed at the entry price of the position.

Protective stops can lastly be placed to secure a gain when the position is currently winning. They are then placed in the direction of the gain of the position:

SET STOP PROFIT x : A protective stop is placed at **x** units from the entry price in the position.

SET STOP pPROFIT x : A protective stop is placed at **x** points from the entry price in the position.

SET STOP %PROFIT x : A protective stop is set at a level corresponding to a gain of **x**%, brokerage fees not included.

SET STOP $PROFIT x : A protective stop is placed at a level corresponding to a gain of **x** € or $ (currency of the instrument), brokerage fees not included.

It is possible to set a stop directly at a defined price, this stop can be placed in the direction of a loss or gain for the position:

SET STOP PRICE x : A protective stop is placed at the price of **x** on the instrument.

The direction (buy or sell) as well as the quantity of the stop order are automatically adapted to the current position. A protective stop is necessarily linked to a given position. If no position is open, the associated protection stop will not be activated.

To deactivate a stop, use one of the following instructions:

```
SET STOP LOSS 0, SET STOP [p/%/$]LOSS 0, SET STOP BREAKEVEN 0, SET STOP PROFIT 0, SET STOP [p/%/$]PROFIT 0, SET STOP PRICE 0
```

Be careful when placing position protection stops, as for position opening orders of type **STOP** and **LIMIT** seen previously, a protection stop placed on the wrong side of the price will be automatically transformed into an at market price order to close the position.

Target orders (Limits)

Profit Target instructions let you exit a position when gains attain a target amount.

SET TARGET PROFIT x : Set a profit target to close the position **x** units from the average position price.

SET TARGET pPROFIT x : Set a profit target to close the position **x** points from the average position price.

SET TARGET %PROFIT x : Set a profit target to close the position when profit reaches **x**% (brokerage fees not included).

SET TARGET $PROFIT x : Set a profit target order to close the position when the gain reaches **x** €,$ (currency of the instrument, brokerage fees not included).

In the case of a losing position, you can set a target order at the position entry price, which will allow you to exit the position without loss (excluding brokerage fees):

SET TARGET BREAKEVEN : A limit is placed on the entry price of the position.

It is also possible to limit losses with a target order when a position is currently losing. The limit is then placed in the direction of the loss of the position:

SET TARGET LOSS x : A limit is placed at **x** units from the position entry price.

SET TARGET pLOSS x : A limit is placed at **x** points from the position entry price.

SET TARGET %LOSS x : A limit is set at a level corresponding to a loss of **x**%, brokerage fees not included.

SET TARGET $LOSS x : A limit is set at a level corresponding to a loss of **x**€ or $ (currency of the instrument), not including brokerage fees.

Finally, as with protective stops, it is possible to place a target limit directly at a set defined price:

SET TARGET PRICE x : A limit order is placed at the price of **x** on the instrument.

The quantity and direction (buy or sell) of the target order are automatically adapted to the type of position currently open. All targets are linked to a position. If there is no open position, the target order is not active.

To deactivate a target in the code, the following instruction can be used:

```
SET TARGET PROFIT 0, SET TARGET [p/%/$]PROFIT 0, SET TARGET BREAKEVEN 0, SET TARGET LOSS 0, SET TARGET [p/%/$]LOSS 0, SET TARGET PRICE 0
```

Be careful when placing target limits, as with STOP and LIMIT orders to open seen earlier and the position protection stops, a target limit placed on the wrong side of the price will be automatically transformed into an at market price order to close the position.

Trailing stops

A trailing stop is a stop order whose price changes depending on the evolution of the price. For long positions, when price increases, the level of a trailing stop increases, but if the price decreases, the level of the trailing stop remains constant. Trailing stops on short positions work in the opposite manner: when price decreases, the level of the trailing stop decreases, but if price increases, the level of the trailing stop remains constant.

Like protection stops, trailing stops can be defined in relative or absolute terms:

`SET STOP TRAILING` **y :** Sets a trailing stop **y** units from average position price.

`SET STOP pTRAILING` **y :** Sets a trailing stop **y** points from average position price.

`SET STOP %TRAILING` **y :** Sets a trailing stop **y**% from average position price, brokerage fees not included.

`SET STOP $TRAILING` **y :** Sets a trailing stop **y** € or $ (currency of the instrument) from average position price, brokerage fees not included.

The quantity and direction (exit long or exit short position) of the trailing stop order are automatically adapted to the type of position currently open. All trailing stops are linked to a position. If there is no open position, the trailing stop is not active.

If the quantity of the position changes, the level of the stop is re-initialized.

To deactivate a trailing stop in the code, the following instruction can be used:

```
SET STOP TRAILING 0, SET STOP pTRAILING 0, SET STOP %TRAILING 0, SET STOP $TRAILING 0
```

Example:

A long position is taken on the DAX at 6000 points and a trailing stop is placed at 50 points:

```
SET STOP pTRAILING 50
```

The stop is initially placed at 5950. Price increases to 6010 then decreases to 5980; the stop will increase 10 points to 5960, then stay there until price increases higher than 6010. It will be triggered if the price reaches 5960.

Use of "`Set Target`" and "`Set Stop`" with conditional "`IF`" statements

It is possible to change the type of target or stop set in your code depending on personalized conditions by using conditional `if` statements.

Example:

```
// Use a stop loss of 10% if the gain of the previous trade was at least 10%, otherwise use a stop loss of 5%.
IF PositionPerf(1) > 0.1 THEN
   SET STOP %LOSS 10
ELSE
   SET STOP %LOSS 5
ENDIF
```

Multi-level stops and limits

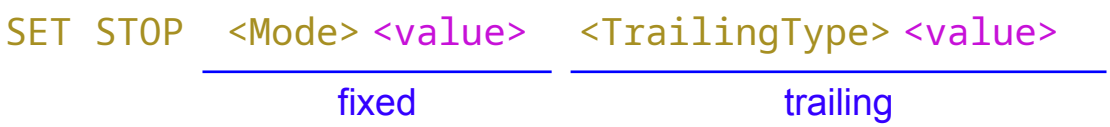
Only one "**Set Stop**" and one "**Set Target**" command can be active at a time under normal circumstances. If there are successive "**Set Stop**" or "**Set Target**" commands in a code, the last command replaces the previous command.

Multi-level stops allow you to combine a classic stop and a trailing stop at the same time on a position, the closer of the two stops will actually be placed on the market, with the possibility that the trailing stop will automatically take the place of the classic stop depending on the market movements. Multi-level stops can be used in backtesting and in automatic trading on our PaperTrading module only.

Example:

```
SET STOP %LOSS 10 // Set a stop loss of 10%
SET TARGET PROFIT 50 // Set a profit target of 50 units
SET TARGET %PROFIT 5 // Removes the previous target of 50 units and replaces it with a profit target of 5%
SET STOP %TRAILING 2 // Removes the previous 10% stop loss and replaces it with a trailing stop of 2%
```

However, it is possible to combine fixed stops and trailing stops or stop losses and trailing stops with a single instruction shown below:



Mode: **LOSS**, **pLOSS**, **%LOSS**, **$LOSS**

Trailing Type: **TRAILING**, **pTRAILING**, **%TRAILING**, **$TRAILING**

This instruction appears in the following form:

```
SET STOP[LOSS/pLOSS/$LOSS/%LOSS] <value> [TRAILING/pTRAILING/$TRAILING/%TRAILING] <value>
```

Examples of use:

SET STOP LOSS x TRAILING y : A stop loss is placed at **x units** from the entry price and it becomes a trailing stop of **y** units if the trailing stop level becomes closer to the current price than the stop loss level (when price varies favorably by **y units – x units**).

SET STOP LOSS x pTRAILING y : A stop loss is placed at **x units** from average position price and it becomes a trailing stop of **y** points if the trailing stop level becomes closer to the current price than the stop loss level (when price varies favorably by **y points – x units**).

SET STOP LOSS x $TRAILING y : A stop loss is placed at **x units** from average position price and it becomes a trailing stop of **y $ or €** (currency of the instrument) if the trailing stop level becomes closer to the current price than the stop loss level.

SET STOP LOSS x %TRAILING y : A stop loss is placed at **x units** from average position price and it becomes a trailing stop of **y**% if the trailing stop level becomes closer to the current price than the stop loss level.

Example:

```
SET STOP LOSS x TRAILING y :
```

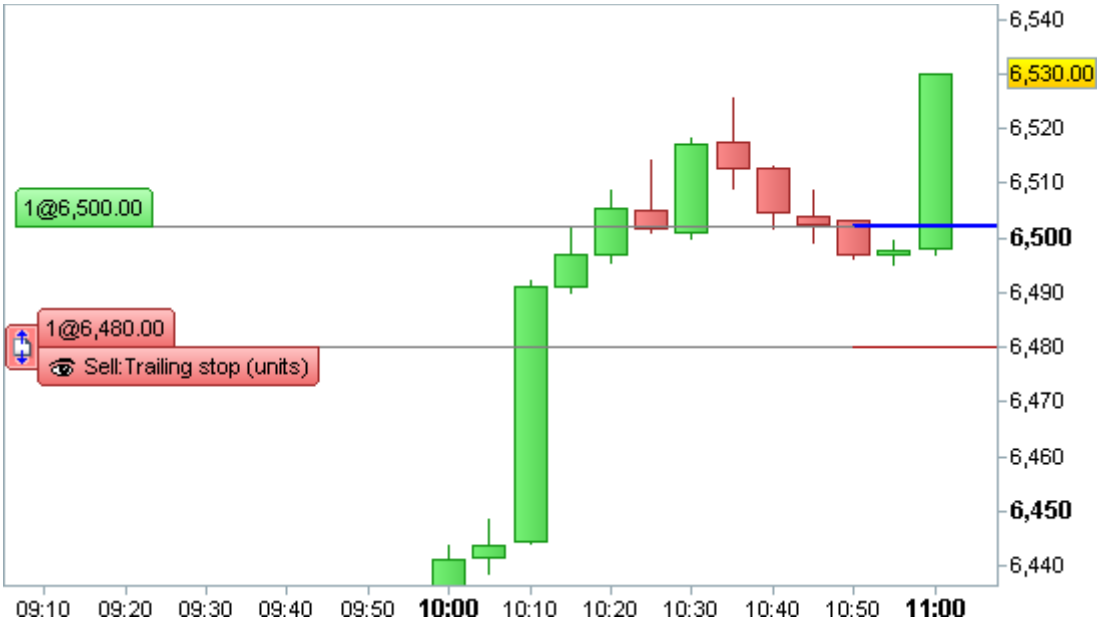
A stop is placed at **x** units from average position price and it becomes a trailing stop of **y** units if the trailing stop level becomes closer tothe current price than the stop loss level.

For example, if you enter a long position on the DAX future at 6,500, the following code places a stop loss **20** units from the average position price which becomes a **50**-unit trailing stop if price passes 6,530.

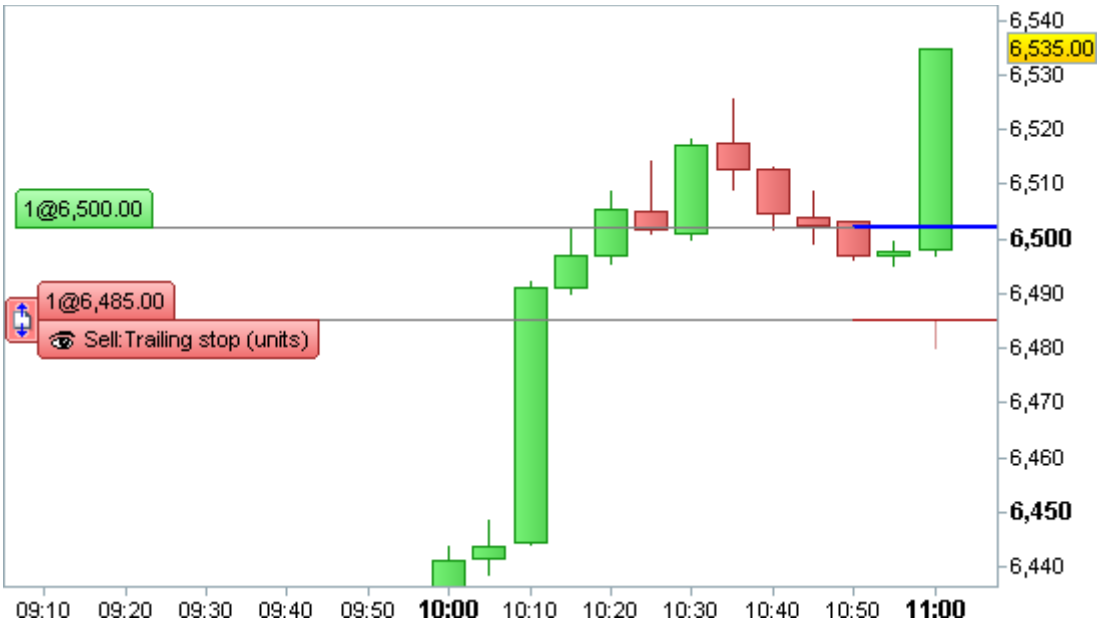
```
SET STOP LOSS 20 TRAILING 50
```

The following images illustrate the example:

The initial stop is placed at a fixed level 20 units below the position opening price (6480)



Only if the price reaches 6,530 (=6,500 + (50-20)), does the stop becomes 50-unit trailing stop.



If price rises to 6,535 as shown in the image above, the trailing stop would rise to 6,485.

Note concerning the use of points or units for the distance of stops and limits

Point size may vary depending on the type of instrument you are looking at, whereas unit size (variation of 1 unit on the chart) is always constant. Depending on your code and the instruments which it is applied to, you may prefer to use distances in points or units. For Example:

On the EuroDollar (EUR/USD), 1 point = 0.0001 units on the chart

On index Futures (DAX, FCE), 1 point = 1 unit on the chart

On futures on European interest rates, 1 point = 0.01 units on the chart

This information is available in our language using the keyword **POINTSIZE**.

Stopping a trading system with QUIT

The "**Quit**" instruction lets you stop a trading system. The stop takes effect when the current bar is closed. Pending orders will be cancelled and all open positions will be closed. This lets you stop a trading system in case of high losses or after a certain date for example.

Example:

```
If Date > 20300101 THEN // Stop the strategy after January 1st, 2030
   QUIT
ENDIF
```

Trading Variables

Position status variables

3 variables allow you to check the status of your trading system's positions:

ONMARKET: is equal to 1 if you have an open position, or zero otherwise.

LONGONMARKET: is equal to one if you have a long position, or zero otherwise.

SHORTONMARKET: is equal to 1 if you have a short position, or zero otherwise.

They can be used with brackets. For example **ONMARKET[1]** is equal to 1 if you had an open position at the close of the previous bar, or 0 otherwise.

These variables are usually introduced with **IF** commands prior to entering a position:

Example:

```
// Observe crossings of the MACD histogram
c1 = MACD[12,26,9](Close) CROSSES OVER 0
// BUY: if there is no long position and MACD > 0, buy 3 lots.
IF NOT LONGONMARKET AND c1 THEN
   BUY 3 SHARES AT MARKET
ENDIF
```

Position size variables

These 3 variables allow you to know the quantity of an open position:

COUNTOFPOSITION: size of the position (in lots, shares, contracts...). It has a positive value if there is a long position open and a negative value if there is a short position open.

COUNTOFLONGSHARES: size of a long position (in lots, shares, contracts...) if there is a long position open. 0 otherwise.

COUNTOFSHORTSHARES: size of a short position (in lots, shares, contracts...). It has a positive value if there is a short position open, 0 otherwise.

These variables are usually introduced with **IF** commands prior to entering a position.

Tip on using status variables: The code is evaluated at the end of each bar, and the orders are placed at the open of the next bar.

For example, in the following block of code, the variable "long" will not be equal to 1 at the close of the first candlestick, but only at the close of the second candlestick because the first buy orders is placed at the open of the second bar.

```
BUY 1 SHARE AT MARKET
IF LONGONMARKET THEN
   long = 1
ENDIF
```

LongTriggered

The **LONGTRIGGERED[n]** command is used to find out if a long position has been opened on the nth previous bar.

LongTriggered[n^{th} previous candlestick]

Note: It is possible to use **LongTriggered** without associating it with a bar number defined between brackets. In this case, the program will consider the bar of the candlestick being calculated, as if you had written : **LongTriggered[0]**

Example:

```
// We reopen a long position only if we did not have an opening/closing position on the same candlestick:
IF NOT LONGONMARKET AND LongTriggered = 0 THEN
   Buy AT MARKET
ENDIF
```

ShortTriggered

The **SHORTTRIGGERED[n]** command is used to find out if a short position opening has taken place on the nth previous candlestick.

ShortTriggered[n^{th} previous candlestick]

Note: It is possible to use **ShortTriggered** without associating it with a bar number defined between brackets. In this case, the program will consider the bar of the candlestick being calculated, as if you had written : **ShortTriggered[0]**

Example:

```
// We reopen a short sale only if we did not have an opening/closing position on the same candlestick:
IF NOT SHORTONMARKET AND ShortTriggered = 0 THEN
   SellShort AT MARKET
ENDIF
```

TradeIndex

The command **TRADEINDEX(n)** lets you access the bar index of the nth previous executed order:

TRADEINDEX(n^{th} previous order)

Note: It is possible to use **TradeIndex** without a number between parenthesis. In this case, the program considers the bar of the last executed order : **TradeIndex(1)**

TradeIndex is usually used conjointly with **BarIndex**.

Example:

```
// Close a long position if it has been open for at least 3 bars
IF LONGONMARKET AND (BarIndex - TradeIndex) >= 3 THEN
   SELL AT MARKET
ENDIF
```

TradePrice

The command **TRADEPRICE(n)** lets you find the price of the previously executed transaction.
The syntax is as follows:

$$\text{TRADEPRICE}(n^{th}\ \text{previous order})$$

If n is not specified, the price of the last executed order is referenced : **TradePrice=TradePrice(1)**

Example:

```
// Close a long position if price is greater than the price of the previous order plus 2%.
IF LONGONMARKET AND CLOSE > 1.02 * TRADEPRICE THEN
   SELL AT MARKET
ENDIF
```

PositionPerf

The instruction **POSITIONPERF(n)** returns:

- The performance (ratio gain/cost of the position) of the nth last position closed if n>0 (not including brokerage fees)
- The performance (ratio gains/cost of the position) of the currently open position if n=0 (not including brokerage fees)

The syntax is as follows:

$$\text{POSITIONPERF}(n^{th}\ \text{previous position})$$

If n is not specified, we suppose that n=0. **PositionPerf=PositionPerf(0)**

Example:

```
// BUY if the previous trade made at least 20% Gain.
IF NOT ONMARKET AND PositionPerf(1) > 0.2 THEN
   BUY 1000 CASH AT MARKET
ENDIF
```

PositionPrice

The command **PositionPrice** lets you know the average purchase price of the currently open position.

$$\text{POSITIONPRICE}[n^{th}\ \text{previous candlestick}]$$

It is calculated as the sum of all entry prices weighted by the quantity of each order. Only adding to a position will change the value of **PositionPrice**.

This instructions may be used with brackets to introduce an offset: **POSITIONPRICE[1]** returns the value of **PositionPrice** at the close of the previous bar.

Example:

If you buy one stock at a price of 5 € and buy the same stock again when the price is 10€ and buy the same stock again when the price is 15€, **PositionPrice** would be equal to: (5 + 10 + 15)/3 = 10 €.

If you then sell one share at a price of 20 €, **PositionPrice** would still be equal to 10 € (no change).

StrategyProfit

This command returns the gains or losses (in absolute and in the currency of the instrument, not including brokerage fees) realized since the beginning of the trading system. It is typically used with "**QUIT**" to stop a trading system that has lost too much.

STRATEGYPROFIT[n^{th} previous candlestick]

This instruction can be used with brackets: **StrategyProfit**[1] gives the profit at the close of the previous bar.

Example:

```
IF STRATEGYPROFIT < -500 THEN
   QUIT
ENDIF
```

Note:

As a reminder, the code of your systems is **evaluated at the close of each candlestick**. In the example above, losses may be greater than 500 € in case of a large loss during a single candlestick or in case of a gap.

As a result, a user who wanted to stop a trading system after 500 € of loss should first set a **STOP LOSS** to limit the losses, then use the block of code above to stop the system.

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

Definition of parameters of execution of trading systems

Additional parameters may be defined with the instruction **DEFPARAM**.


Reminder about **DEFPARAM** instructions: they should always be at the beginning of your code.

Cumulate orders

The parameter **CumulateOrders** lets you authorize or forbid accumulating orders to enter the market or add to a position. This parameter is set to "**True**" by default for codes created by programming which means that a trading system may add to an existing position at every bar where the conditions to enter that position are true. It is also possible to have multiple limit or stop orders to enter the market active at the same time in this case.

To prevent a strategy from increasing the size of an already open position, the following instruction must be set at the beginning of the code:

```
DEFPARAM CumulateOrders = False
```

DEFPARAM instructions remain valid for the entire execution of the trading system. It is not possible to change this trading system's setting during its execution.

Examples:

```
// This code will buy 1 share every bar, up to a maximum of 3.
DEFPARAM CumulateOrders = True
If CountOfPosition < 3 THEN
   Buy 1 shares at market
Endif

// This code will buy 1 share at a price of 2 and an additional share at a price of 3
DEFPARAM CumulateOrders = True
If CountOfPosition < 2 THEN
   Buy 1 shares at 2 Limit
   Buy 1 shares at 3 Limit
Endif

// This code will buy 5 shares only once
DEFPARAM CumulateOrders = False
Buy 5 shares at market
```

It is possible to have several orders to exit the market in the same direction even while the parameter **CumulateOrders** is set to "**False**".

Example:

```
// This code will buy 5 shares. Up to 3 shares will be sold if price crosses under the 40-period moving average. All shares will be sold in case of a 10% loss.
DEFPARAM CumulateOrders = False
Buy 5 shares at market
IF Close CROSSES UNDER Average[40] THEN
   SELL 3 SHARES AT MARKET
   Set stop %Trailing 10
ENDIF
```

Note on stops and target levels while CumulateOrders is active("True"): If you use the instructions to set a stop loss, trailing stop or profit target with **CumulateOrders** activated, the level is calculated based on your positions average entry price and is recalculated each time the position's quantity is modified.

Example:

If you buy 1 share at $10.00 and set a 10% stop loss and a 150% profit target, the initial levels would be: Stop at $9.00 and target at $25.00. If you buy a second share at a price of $20, the average entry price would be $15 and as a result the new levels would be: Stop at $13.50 and target at $37.50 (for the entire position).

Note on codes created with assisted creation mode: **CumulateOrders** is set to false by default for trading systems created in assisted creation mode (the instruction " **DEFPARAM CumulateOrders = False**" will be present at the beginning of these codes.

Pre-loading of historical data

The **DEFPARAM PRELOADBARS** instruction lets you configure the maximum amount of bars that are preloaded prior to the start of a trading system for the calculation of indicators used in the system prior to the system's start (personal or predefined indicators). By default this parameter is equal to 1000. It cannot be less than 1 or higher than 10 000. The system needs to run a calculation on at least one historical candlestick in order to initialize your code on our servers. Therefore, using **DEFPARAM PRELOADBARS = 0** will have the same effect as **DEFPARAM PRELOADBARS = 1**, there will be one preloaded candlestick so that your trading system can start.

The actual amount of preloaded bars is also limited by the available history for the instrument.

Example:

```
DEFPARAM PRELOADBARS = 300
a = ( Close + Open ) / 2
If Close CROSSES OVER Average[250](a) THEN
   BUY 1 SHARE AT MARKET
Endif
```

If the parameter **PRELOADBARS** is set to 300, it means that a moving average of 250 bars would be defined at the very first bar after a trading system started. This would not be the case if only 200 bars were preloaded.

Note that the value of 300 is a maximum: If less than 300 bars are available prior to the start of the trading system, only the available number of bars will be preloaded.

In the example where 300 bars are preloaded, the **BarIndex** of the first bar after the start of the trading system is equal to 300. On the other hand, if 1 bar is preloaded, the **BarIndex** of the first bar after the start of the strategy would be equal to 1.

Warning: a variable assignment preceded by "**ONCE**" will be executed only once at the first reading of this line of code (including on the history loaded with "**PRELOADBARS**")

In the example below, the strategy will not place any order (because tmp variable is at 1 during the **PRELOADBARS**, and no order can be placed during this period).

Example:

```
DEFPARAM PreLoadBars = 200
ONCE tmp = 1
IF tmp = 1 THEN
   BUY 1 SHARE AT MARKET
   tmp = 0
ENDIF
```

FlatBefore and FlatAfter

```
DEFPARAM FlatBefore = HHMMSS
DEFPARAM FlatAfter = HHMMSS
```

HHMMSS is a time where **HH** indicates the hour, **MM** indicates the minutes and **SS** indicates the seconds.

These instructions let you cancel any pending orders, close all positions and prevent placement of additional orders before a certain time of day in the case of **FlatBefore** or after a certain time of day in the case of **FlatAfter** in the time zone of the strategy.

The parameter **FlatBefore** must always be later than the market opening time (customized or not), and **FlatAfter** must be earlier than the market close time (customized or not), otherwise they would have no effect. If the chosen time is not a multiple of the main timeframe of the trading system, (it occurs in the middle of a candlestick), the instruction **DEFPARAM FlatAfter** will take effect at the close of that candlestick and the instruction **DEFPARAM FlatBefore** will be applied until the close of the candlestick .

Orders are restricted during this period, meaning that no orders will be placed and any such orders will not be placed at the opening of the next period when the trading system is authorized to place orders. As a result "OnMarket" type variables will always be false during these times.

Example:

```
DEFPARAM FlatBefore = 093000 // Cancel any pending orders, close any positions and prevent placement of additional orders by the trading system before 9:30:00 in the time zone of the strategy.
DEFPARAM FlatAfter = 160000 // Cancel any pending orders, close any positions and prevent placement of additional orders by the trading system after 16:00:00 in the time zone of the strategy.
```

NoCashUpdate (backtest only)

```
DEFPARAM NoCashUpdate = True
```

If this option is activated, the available cash is not updated with gains, losses and brokerage fees, otherwise it's updated by default.

Example:

Initial capital 10,000 €, with **DEFPARAM NoCashUpdate = True**. The maximum investment will be limited to 10,000€, whatever the gains and losses realized for the entire duration of the backtest.

Reminder:

Parameters defined with the **DEFPARAM** instruction must be defined in the first lines of the code (after any comments).

MinOrder and MaxOrder (backtest only)

```
DEFPARAM MinOrder = n
DEFPARAM MaxOrder = p
```

This option lets you block all orders whose quantity (in lots, contracts or shares) is below n or above p.

Example:

```
DEFPARAM MinOrder = 100
Buy 1000 cash at market
```

If the current price is above 10€, the order quantity will be below 100 therefore the order will not be placed.

Calling indicators

ProRealTime indicators

All of the functions including ProRealTime indicators available for programming your own indicators are also accessible to program trading systems (see the glossary at the end of the manual for a complete list).

We advise you to check the ProBuilder manual for more detail about these functions.

The quantity of historical data necessary to calculate an indicator depends on the type of indicator.

For example, to calculate an exponential moving average over n periods (**ExponentialAverage[N]**), we generally consider that 20*N bars are necessary to obtain a precise result.

If the beginning of the backtest is very close to the beginning of the chart, additional history may be provided by the server for the calculation of the trading system so that the indicators have values at the beginning of the backtest.

Personal indicators

It is possible to call ProBuilder indicators that you have programmed using the "**CALL**" instruction in a trading system.

Example:

```
a,b = CALL "HistoMACD"[5,6] // 5 and 6 being the parameters passed as input to the HistoMACD function
```

To learn more about optimal use of the **CALL** function, read the section below on how to optimize your programs.

Programming tips

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

The time required to calculate a trading system is strongly dependent on the complexity of the indicators used and the way they are called. The following paragraph provides some simple advice to optimize your codes from a programming point of view.

Reduce the number of calls to indicators

If you use the same indicator more than once in a program, stock the indicator in an intermediary variable (avg40 in the example below) rather than calling the indicator again. This will speed up the execution significantly.

NON-OPTIMAL CODE	OPTIMAL CODE

`IF NOT LONGONMARKET AND Close > Average[40](Close) THEN`
`   BUY 1 SHARE AT MARKET`
`ENDIF`
 	
`IF NOT SHORTONMARKET AND Close < Average[40](Close) THEN`
`   SELLSHORT 1 SHARE AT MARKET`
`ENDIF`
	`avg40 = Average[40](Close)`
	`IF NOT LONGONMARKET AND Close > avg40 THEN`
	`   BUY 1 SHARE AT MARKET`
	`ENDIF`
	
	`IF NOT SHORTONMARKET AND Close < avg40 THEN`
	`   SELLSHORT 1 SHARE AT MARKET`
	`ENDIF`

This is also valid if you want to use the same indicator several times but with a different offset.

NON-OPTIMAL CODE	OPTIMAL CODE

`a = ExponentialAverage[40](Close)`
`b = ExponentialAverage[40](Close[1])`
`c = ExponentialAverage[40](Close)`
 	
`IF a > b THEN`
`   BUY 1 SHARE AT MARKET`
`ENDIF`
 	
`IF a < c[1] THEN`
`   SELLSHORT 1 SHARE AT MARKET`
`ENDIF`
	`a = ExponentialAverage[40](Close)`
	
	`IF a > a[1] THEN`
	`   BUY 1 SHARE AT MARKET`
	`ENDIF`
	
	`IF a < a[1] THEN`
	`   SELLSHORT 1 SHARE AT MARKET`
	`ENDIF`

Calls to personal indicators

Calling personal indicators with the **CALL** instruction is slower than using ProRealTime indicators. For ProRealTime indicators, we know in advance what calculations are necessary and can control how they are done. This lets us increase the speed of calculations which is not possible with personal indicators which the user programs themselves.

To improve the execution speed of a trading system with the **CALL** instruction, it is important to use **CALL** as efficiently as possible in the program.

Limit the number of identical calls:

As for ProRealTime indicators, limit the number of times an indicator is called in the program.

NON-OPTIMAL CODE	OPTIMAL CODE

`F1 = CALL "My Function"`
 	
`IF NOT LONGONMARKET AND Close > F1 THEN`
`  BUY 1 SHARE AT MARKET`
`ENDIF`
 	
`F2 = CALL "My Function"`
`IF NOT SHORTONMARKET AND Close < F2 THEN`
`  SELLSHORT 1 SHARE AT MARKET`
`ENDIF`
	`F = CALL "My Function"`
	
	`IF NOT LONGONMARKET AND Close > F THEN`
	`  BUY 1 SHARE AT MARKET`
	`ENDIF`
	
	`IF NOT SHORTONMARKET AND Close < F THEN`
	`  SELLSHORT 1 SHARE AT MARKET`
	`ENDIF`

Limit nested calls:

If you are using a personal indicator in the code of your backtest, check that this personal indicator does not have a **CALL** instruction in its code.

Calling a personal indicator which calls another personal indicator is costly in terms of calculation time. If possible, always duplicate the code of the personal indicator you want to call directly in your ProBacktest rather than using the **CALL** function so that your backtest codes only use standard ProRealTime indicators.

NON-OPTIMAL CODE	OPTIMAL CODE

Code of the trading system:
`LinReg = CALL "MyDCLOSEMix LinearReg"`
 	
`IF NOT LONGONMARKET AND Close > LinReg THEN`
`  BUY 1 SHARE AT MARKET`
`ENDIF`
 	
`IF NOT SHORTONMARKET AND Close < LinReg THEN`
`  SELLSHORT 1 SHARE AT MARKET`
`ENDIF`
 	
Code of "MyDCLOSEMix LinearReg":
`dayclosemix = CALL "MyDCLOSEMix"`
 	
`RETURN LinearRegression[5](dayclosemix)`
 	
Code of "MyDCLOSEMix":
`mix = (DClose(0) + 3.5 * DClose(1) + 4.5 * DClose(2) + 3 * DClose(3) + 0.5 * DClose(4) - 0.5 * DClose(5) - 1.5 * DClose(6)) / 10.5`
 	
`RETURN mix`
	Code of the trading system:
	`dayclosemix = (DClose(0) + 3.5 * DClose(1) + 4.5 * DClose(2) + 3 * DClose(3) + 0.5 * DClose(4) - 0.5 * DClose(5) - 1.5 * DClose(6)) / 10.5`
	`LinReg = LinearRegression[5](dayclosemix)`
	
	`IF NOT LONGONMARKET AND Close > LinReg THEN`
	`  BUY 1 SHARE AT MARKET`
	`ENDIF`
	
	`IF NOT SHORTONMARKET AND Close < LinReg THEN`
	`  SELLSHORT 1 SHARE AT MARKET`
	`ENDIF`

Limit nested loops:

For all conditional statements (**IF**...**THEN**...**ENDIF**), it is always preferable in terms of calculation time to use one condition that verifies n conditions rather than to use n instructions as shown below.

NON-OPTIMAL CODE	OPTIMAL CODE

IF Close >= 0.14 THEN
  IF Close <= 0.47 THEN
  IF IntradayBarIndex >= 5 THEN
  IF IntradayBarIndex <= 20 THEN
  IF NOT SHORTONMARKET THEN
  BUY 1 SHARES AT MARKET
  ENDIF
  ENDIF
  ENDIF
  ENDIF
ENDIF
	cClose = Close >= 0.14 AND Close <= 0.47
	cIndex = IntradayBarIndex >= 5 AND IntradayBarIndex <= 20
	IF cClose AND cIndex AND NOT SHORTONMARKET THEN
	  BUY 1 SHARES AT MARKET
	ENDIF

Use of FOR loops:

Use of **FOR** loops is sometimes necessary, but it's better to limit their use when possible as they increase calculation time.

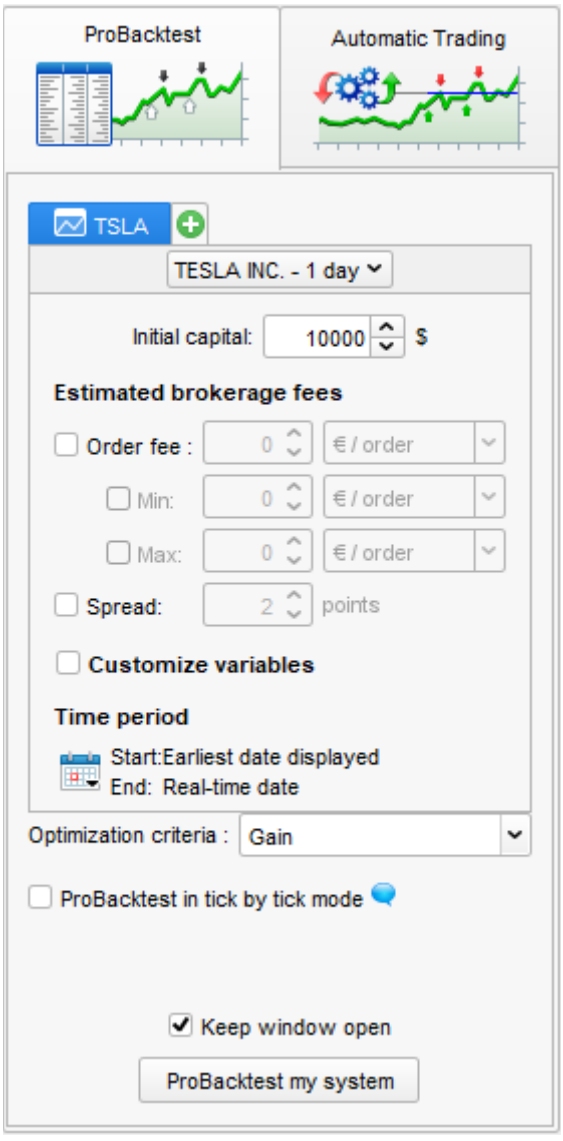
Here are several separate examples where using a **FOR** loop is avoided:

```
// Determine if the condition C1 is true at least once over the last n candlesticks:
IF HIGHEST[n](c1) = 1 THEN
   ...
// Determine if the condition c1 was always true over the last n candlesticks:
IF LOWEST[n](c1) = 1 THEN
   ...
// Determine the number of times the condition c1 was verified over the n last candlesticks:
num = SUMMATION[n](c1)

// Determine the number of bars since c1 was true:
IF c1 THEN
   lastoccurrence = BarIndex
ENDIF
timesince = BarIndex - lastoccurrence

// Find the maximum of variables (a,b,c,d,e,f,g):
top = MAX(a , MAX(b , MAX(c , MAX(d , MAX(e , MAX(f , g) ) ) ) ) )
```

ProBacktest - trading system backtesting

The “ProBacktest” tab of the trading system creation window lets you configure the parameters of the system:

- Instrument(s)
- Graph on which to apply the backtest
- Initial capital
- Lot size (forex)
- Brokerage parameters
- Optimization variable model
- Simulation period



Money Management

Initial capital

This section lets you assign the capital that is available for the trading system to trade with. The maximum authorized investment of the system depends on this.

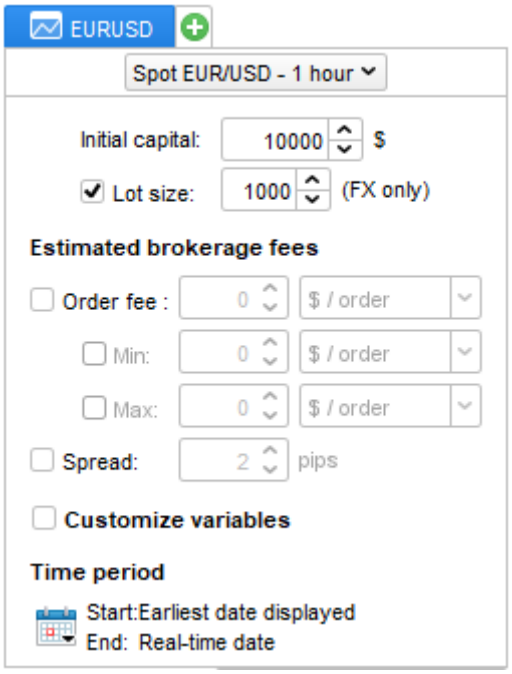
During the execution of a backtest, gains, losses and brokerage fees affect the amount of capital available to a trading system (unless you activate the `NoCashUpdate` setting). See the section **Reinvest gains** for more information. For automatic trading systems, the capital available to your systems is the capital in your portfolio.

If a backtest does not place any trades, try increasing the amount of initial capital.

Brokerage fees and spread

You can customize these parameters to accurately reflect the fees and other parameters of your broker. Any kind of brokerage fee can be applied to any type of instrument. The types of brokerage fees available include:

- Cash per order: Fixed amount of cash (in the currency of the instrument) applied every time an order is executed. You can also specify a minimum and a maximum value per order.
- % Transaction: percentage of the transaction (in the currency of the instrument) applied every time an order is executed.
- Cash per lot: Fixed amount of cash (in the currency of the instrument) applied per lot or contract.
- Lot size (Forex only): It's the minimum order quantity on the instrument. Every order quantity entered in `BUY`/`SELL` instruction is multiplied by the lot size.
- Spread (in pips): the value added to the mid price to reflect the bid-ask spread.



Futures:

For futures, Brokerage fees are typically defined as a fee per lot and per transaction.

The margin is the cash necessary to buy a contract. The value of a point is automatically calculated by the workstation for the future on which you are backtesting the trading system.

Here are the point values of the main futures.

CONTRACT CODE	FUTURE NAME	POINT VALUE
MNQ	Micro E-mini NASDAQ100	2$
MES	Micro E-mini S&P 500	5$
DXS	Micro DAX	1€
MYM	Micro E-mini DJ30	0.5$
DXM	Mini DAX	5€
M2K	Micro E-mini Russel 2000	5$
FCE	CAC40 FCE	10€
MGC	Micro Gold	1$
NQ	Mini NASDAQ100	20$
ES	Mini S&P 500	50$

This information is available in the ProBuilder programming language using the keyword `POINTVALUE`.

Forex:

The spread, lot size and margin can be defined and are applied to each order.

Example with EURUSD:

- Lot size: 100,000
- Spread: 2 pips

The instruction `BUY 1 LOT AT MARKET` on EURUSD buys 1 lot of 100,000 with a spread of 2 pips. (0.0002).

Stocks:

The brokerage fees are usually defined in fees per order in € or in % of the transaction. It's also possible to define minimum or maximum brokerage fees per transaction.

Variable optimization

Variable optimization lets you test different combinations of variables in a backtest to see which combinations give the best results for a given instrument and timeframe over the period of historical data tested.

To learn more and see an example, we suggest you watch the video "**Money management, stops and optimization**".

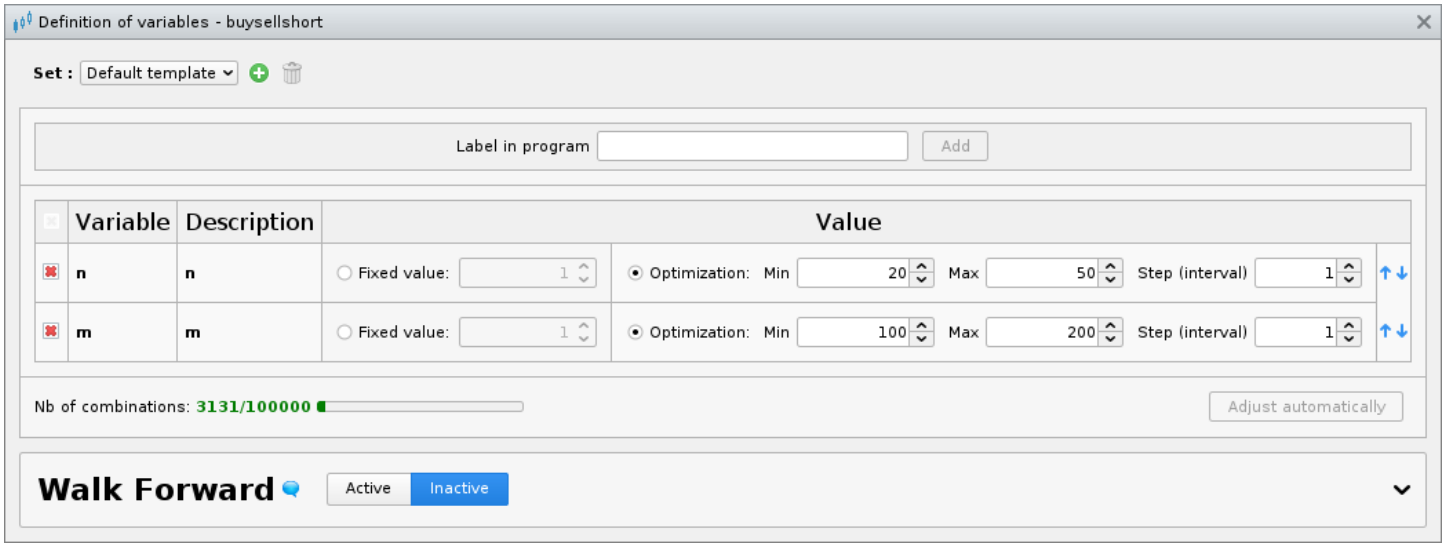
The result of the optimization is presented in an “Optimization report”. You will see the statistics of the best combinations of variables tested and use this information to determine which variable you want to use in your trading system.

Here is an example of a program that can be optimized with 2 moving averages with periods of n and m.:

```
AVGm = ExponentialAverage[m](Close)
AVGn = ExponentialAverage[n](Close)
IF AVGm CROSSES OVER AVGn THEN
   BUY 100 SHARES AT MARKET
ENDIF
IF AVGm CROSSES UNDER AVGn THEN
   SELL 100 SHARES AT MARKET
ENDIF
```

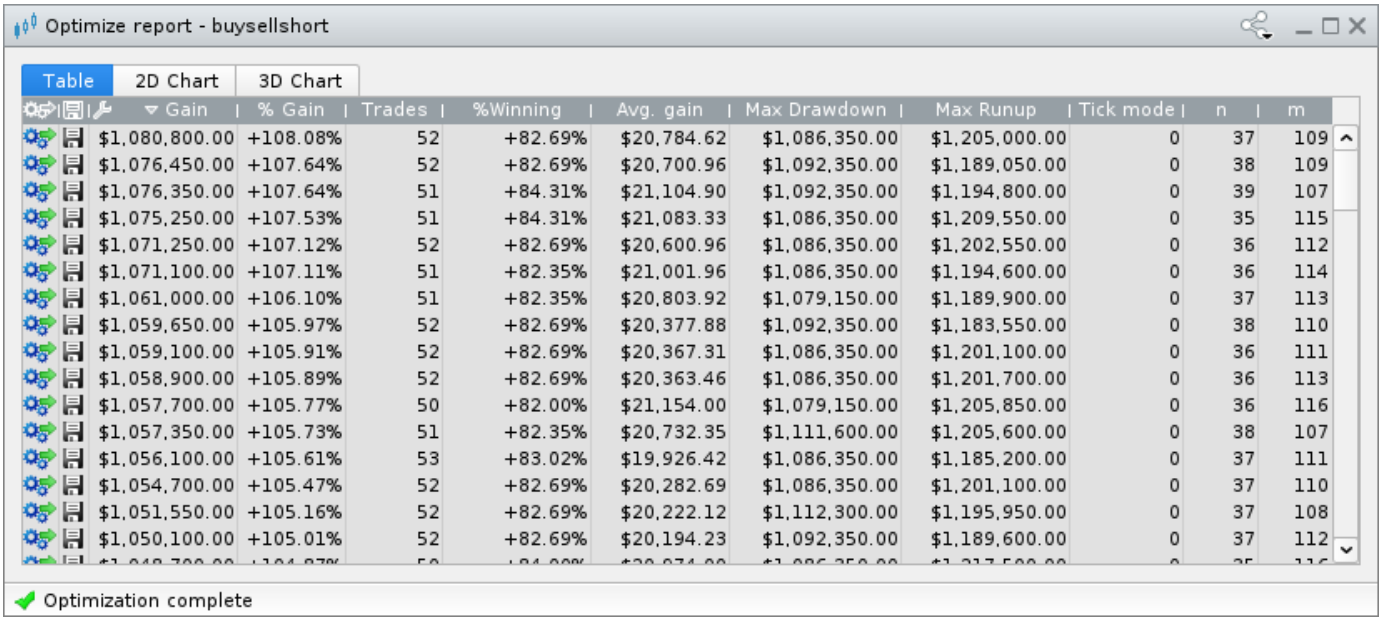
The variables n and m can then be defined by clicking on the “Add” button in the optimization section:

The following window then opens where you can set up the optimization:



- **"Label in program"** is the name of the variable in the code (n and m in this case).
- **"Fixed value"** is used to set the variable to a defined value.
- **"Min." and "Max."** are the limits of the variable for the optimization test.
- **"Step"** is the interval of variables to be tested in the optimization.

Here is an example of an optimization report:



Gain	% Gain	Trades	%Winning	Avg. gain	Max Drawdown	Max Runup	Tick mode	n	m
$1,080,800.00	+108.08%	52	+82.69%	$20,784.62	$1,086,350.00	$1,205,000.00	0	37	109
$1,076,450.00	+107.64%	52	+82.69%	$20,700.96	$1,092,350.00	$1,189,050.00	0	38	109
$1,076,350.00	+107.64%	51	+84.31%	$21,104.90	$1,092,350.00	$1,194,800.00	0	39	107
$1,075,250.00	+107.53%	51	+84.31%	$21,083.33	$1,086,350.00	$1,209,550.00	0	35	115
$1,071,250.00	+107.12%	52	+82.69%	$20,600.96	$1,086,350.00	$1,202,550.00	0	36	112
$1,071,100.00	+107.11%	51	+82.35%	$21,001.96	$1,086,350.00	$1,194,600.00	0	36	114
$1,061,000.00	+106.10%	51	+82.35%	$20,803.92	$1,079,150.00	$1,189,900.00	0	37	113
$1,059,650.00	+105.97%	52	+82.69%	$20,377.88	$1,092,350.00	$1,183,550.00	0	38	110
$1,059,100.00	+105.91%	52	+82.69%	$20,367.31	$1,086,350.00	$1,201,100.00	0	36	111
$1,058,900.00	+105.89%	52	+82.69%	$20,363.46	$1,086,350.00	$1,201,700.00	0	36	113
$1,057,700.00	+105.77%	50	+82.00%	$21,154.00	$1,079,150.00	$1,205,850.00	0	36	116
$1,057,350.00	+105.73%	51	+82.35%	$20,732.35	$1,111,600.00	$1,205,600.00	0	38	107
$1,056,100.00	+105.61%	53	+83.02%	$19,926.42	$1,086,350.00	$1,185,200.00	0	37	111
$1,054,700.00	+105.47%	52	+82.69%	$20,282.69	$1,086,350.00	$1,201,100.00	0	37	110
$1,051,550.00	+105.16%	52	+82.69%	$20,222.12	$1,112,300.00	$1,195,950.00	0	37	108
$1,050,100.00	+105.01%	52	+82.69%	$20,194.23	$1,092,350.00	$1,189,600.00	0	37	112

The optimization report gives 5 statistics for each combination of variables tested. These statistics are as follows:

- **"Gain"**, is the gain or loss realized by the trading system. The calculation formula is:

Gain = Final capital – Initial capital

This statistic lets you evaluate the absolute gain potential with the trading system defined for the historical period tested and for each variable combination.

Note: Brokerage fees as defined in the “Brokerage parameters” section are taken into account in this calculation.

- **"%Gain"**, is the gain or loss in %. The calculation formula is:

%Gain = 100 x Profit / Initial capital

This indicates the relative performance of the backtest configured with the corresponding variables.

- **"Trades"** indicates the number of positions opened during the backtest.

- **"% winning"** indicates the % of winning positions. It is calculated as:

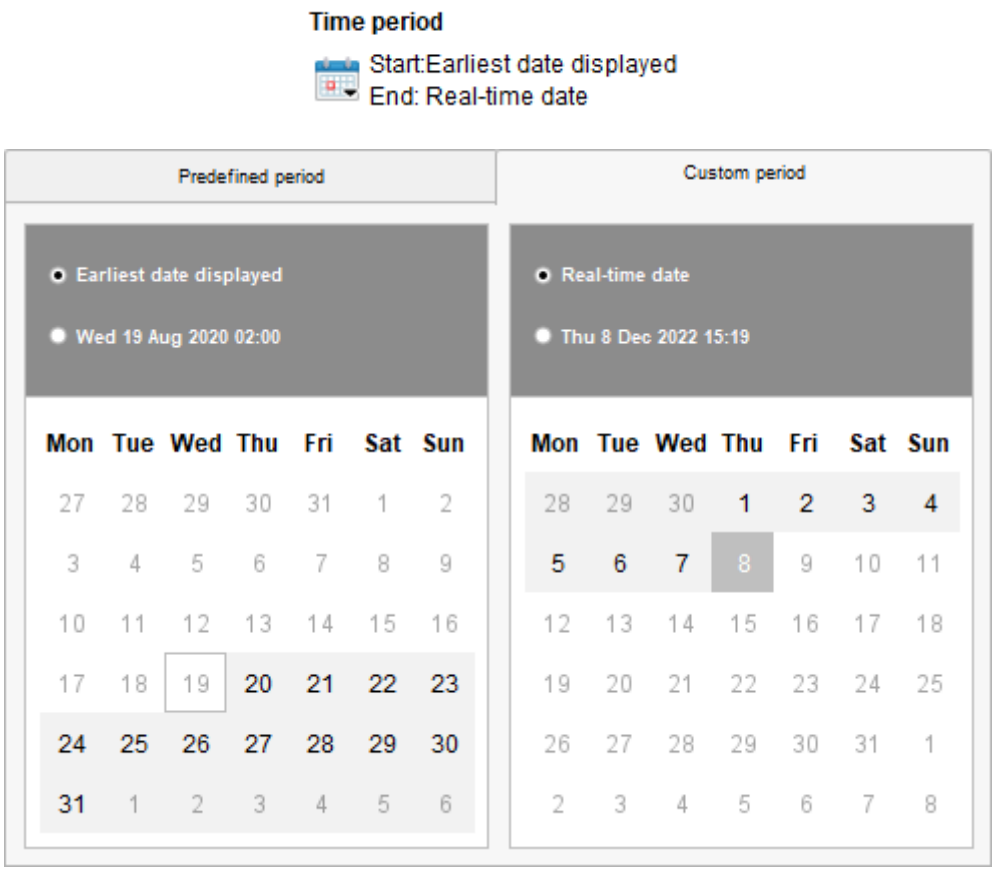
% winning = (100 x number of winning positions) / Number of positions

- **"Avg. gain"** is the average gain per position. It can be useful to determine efficiency of orders placed. It is defined as:

Avg gain per position = Gain / Number of positions

Note: The results of optimization reports may change for a given trading system depending on the security, timeframe or amount of historical data used.

Definition of the period of execution of the backtest



This area lets you define the beginning and end of the backtest. Note that the amount of data that can be used for the backtest is generally limited to the amount of data displayed on the chart. You can increase the amount of historical data loaded on your chart by using the left dropdown menu of each chart. You need to load the amount of data on which you want to run the test before executing the backtest.

In the case of a “real-time” backtest, the orders appear on the chart whenever a signal is triggered. It is also possible to associate these orders with popups or sounds by selecting “Alert and sound configuration” from the “Options” menu.

If an ending date is defined, all positions opened at that date are closed.

Note:

If your backtest takes a long time to execute, you can reduce the period of execution: the amount of time it takes to backtest a trading system is proportional to the amount of historical data on which the trading system is tested.

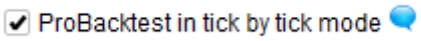
After executing a backtest, the results are displayed in the following manner:

- Equity curve showing the gains and losses of the trading system
- Positions histogram
- Detailed report

For more information about the display of results of backtests, check Annex A at the end of this document.

Tick by tick mode:

You have the option to run your backtest in tick-by-tick mode to solve execution sequence issues when two orders can be executed in the same candlestick.



Using this option can impact the computation time of your trading system. The more positions that need to be checked, the more the impact on the computational performance of your code will be visible. In order to avoid excessive calculation times, a limit on the number of positions that can be checked in "tick by tick" mode has been defined (1,000 by default, 10,000 for premium platforms).

It should also be noted that the use of this option can impact the results of your trading system's positions and thus change the overall performance of the system.

Optimizations are not run in tick by tick mode but the “Tick mode” of an optimization report indicates the number of cases detected.

Optimize report - buysellshort

Table | 2D Chart | 3D Chart

Gain	% Gain	Trades	%Winning	Avg. gain	Max Drawdown	Max Runup	Tick mode	n	m
$44,000.00	+4.40%	55	+60.00%	$800.00	$26,300.00	$52,000.00	19	34	101
$42,250.00	+4.22%	61	+59.02%	$692.62	$24,350.00	$46,250.00	15	23	101
$40,000.00	+4.00%	54	+59.26%	$740.74	$30,300.00	$52,000.00	19	34	100
$38,250.00	+3.82%	60	+58.33%	$637.50	$22,400.00	$42,250.00	13	23	100
$38,250.00	+3.82%	60	+58.33%	$637.50	$23,800.00	$46,250.00	14	22	101
$36,000.00	+3.60%	63	+57.14%	$571.43	$28,000.00	$68,000.00	19	20	100
$34,200.00	+3.42%	55	+58.18%	$621.82	$28,000.00	$46,200.00	18	35	101
$32,000.00	+3.20%	56	+57.14%	$571.43	$27,250.00	$40,000.00	19	33	100
$32,000.00	+3.20%	54	+57.41%	$592.59	$30,300.00	$44,000.00	17	35	100
$30,250.00	+3.02%	60	+56.67%	$504.17	$31,800.00	$50,050.00	14	22	100
$28,000.00	+2.80%	53	+56.60%	$528.30	$27,250.00	$36,000.00	18	33	101
$24,000.00	+2.40%	52	+55.77%	$461.54	$28,000.00	$44,000.00	19	36	100
$24,000.00	+2.40%	64	+54.69%	$375.00	$28,000.00	$56,000.00	18	20	101
$24,000.00	+2.40%	52	+55.77%	$461.54	$28,000.00	$44,000.00	19	36	101
$20,000.00	+2.00%	49	+55.10%	$408.16	$20,000.00	$36,000.00	13	46	100
$20,000.00	+2.00%	55	+54.55%	$363.64	$36,900.00	$49,650.00	20	32	101

Optimization complete

After clicking on one line, the associated backtest will start and solve each conflict to know which of the `STOP` or `TARGET` was first hit, giving you more accurate results for your system.

Note: the results can differ between the optimization window and the backtest results as the tick by tick is not applied directly on the optimization.

Customization of trading hours for backtesting

The menu "Time zones and trading hours" of the setting windows lets you define customized trading hours for a market.

Customized trading hours: If you define reduced trading hours for a market, that means that only data during these reduced trading hours will be shown on charts (and taken into account by ProBacktest). Note that it is only possible to define reduced trading hours within a given day. For example, a the market is open 24 hours per day, you could choose to take into account only data between 10:00 and 14:00, but it is not possible to take into account data between 21:00 one day and 9:00 the next day. If an order is placed at the customized close of the last bar of the trading day with customized trading hours configured, this order will be placed at the customized open of the next trading day.

The price constants `Dopen(x)/Dhigh(x)/Dlow(x)/Dclose(x)` on the current candlestick (x=0) take into account the custom time frames you set. They use the official market data (without custom time frames) for the previous candlesticks (x>0).

Using the option to apply these settings in the daily and higher timeframes does not affect this behavior.

Notes concerning customized time zones and weekend data:

Some 24-hour markets have options to "use intraday quotes to build daily candles". This option is not taken into account by ProBacktest, which always uses official daily candles based on the standard (local market) time zone.

Some markets (such as Forex) include weekend data. There is a checkbox for these markets in "Options / Set time zones and trading hours" which allows the users do hide weekend data in the charts. Weekend data is always taken into account for backtesting purposes. For the Forex market, Sunday's data is included in the Monday daily candle for backtesting purposes (there is no daily candle for Sunday).

Please note that depending on the conditions set by your broker this rule may be different and Sunday may have its own daily candlestick.

Notes concerning customized time zones:

The code is always executed in the user's time zone. This means that time-based instructions (`Time`, `Flatafter`, `Flatbefore`) will take this time zone into account for their calculation. The user's timezone means the time zone chosen for the market to which the instrument belongs. It is customizable in the menu Platform Settings > Time zones & Trading hours.

By default, the time zone of the instrument is the time zone of your computer.

It's possible to change the time zone of a market from the Platform Settings > Time zones & Trading hours menu.

Example: On Vodafone (on the LSE – timezone GMT+1 during summer time or BST), I set the chart to Paris time (GMT+2 during summer time or CET), the instruction time will return 1,300 (time of the close of the 15 minute candle beginning at 11:45 BST: 12:00 converted to 13:00 CET).

All intraday time instructions are concerned:

- `Time` and its derivative instructions (`hour`, `minute`...)
- `Opentime` and its derivative instructions (`openhour`, `openminute`...)
- `FlatAfter` and `FlatBefore`
- `IntradayBarIndex` (reset to zero at the open of the market in the user's time zone)

Daily time-based instructions are not affected by the time zone selected:

- `Dopen`, `Dhigh`, `Dlow`, `Dclose`
- `Date` and its derivative instructions (`year`, `month`, `day`)
- `DayOfWeek` and `Days`

These instructions take into account the time zone of the local market.

Reasons a ProBacktest may stop

A ProBacktest may stop in one of the following cases:

- The backtest reaches the end time specified in the programming window. In this case, the end of the backtest is shown only by a black vertical line on the chart.
- There is a "**Quit**" instruction in the code which is executed. In this case, the end of the backtest is shown by the following icon: 
- The available capital is no longer sufficient to cover losses ("estimated" capital is negative). In this case, the end of the backtest will be shown by this icon: 
- An order is rejected due to insufficient cash. This order will appear in the order list of the detailed report. In this case, the end of the backtest will be shown by this icon:

Here is an example of a backtest stopped due to insufficient capital:

Display of the values of backtest variables

Along with the **PRINT** instruction detailled in the **ProBuilder manual**, the ProBacktest Module provides 2 additionnal instructions to help you understand how each variable behave in your code :

- **GRAPH** to use a dedicated panel to visualize your variables.
- **GRAPHONPRICE** to visualize your variables on the price chart directly.

The **GRAPH** instruction lets you display the values of variables you use in your ProBacktest program.

This instruction works in the following way:

```
GRAPH myvariable AS "my variable name"
```

- **myvariable** is the name of the variable in the code
- "my variable name" is the label of the variable which will be displayed on the chart

It is also possible to define a color for the variable with the optional **COLOURED** parameter:

```
GRAPH myvariable COLOURED (r,g,b) AS "my variable name"
```

"r","g", and "b" are whole numbers from 0 to 255 (RGB format)

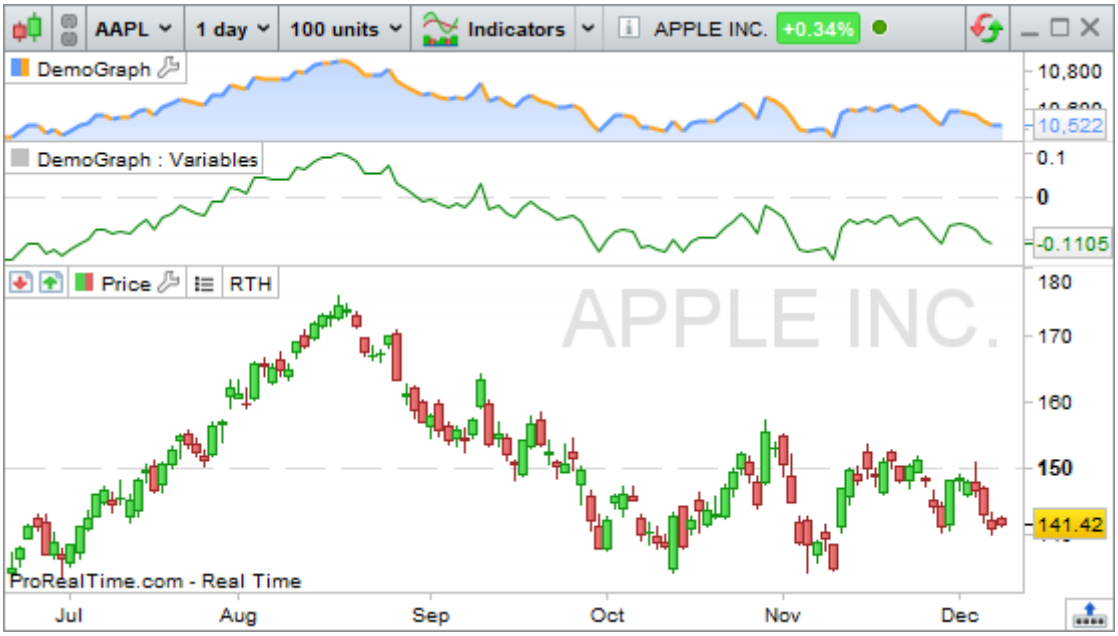
e.g. (255, 0, 0) for a red curve

You can also set the transparency of the line as follows:

```
GRAPH myvariable COLOURED (r,g,b,a) AS "my variable name"
```

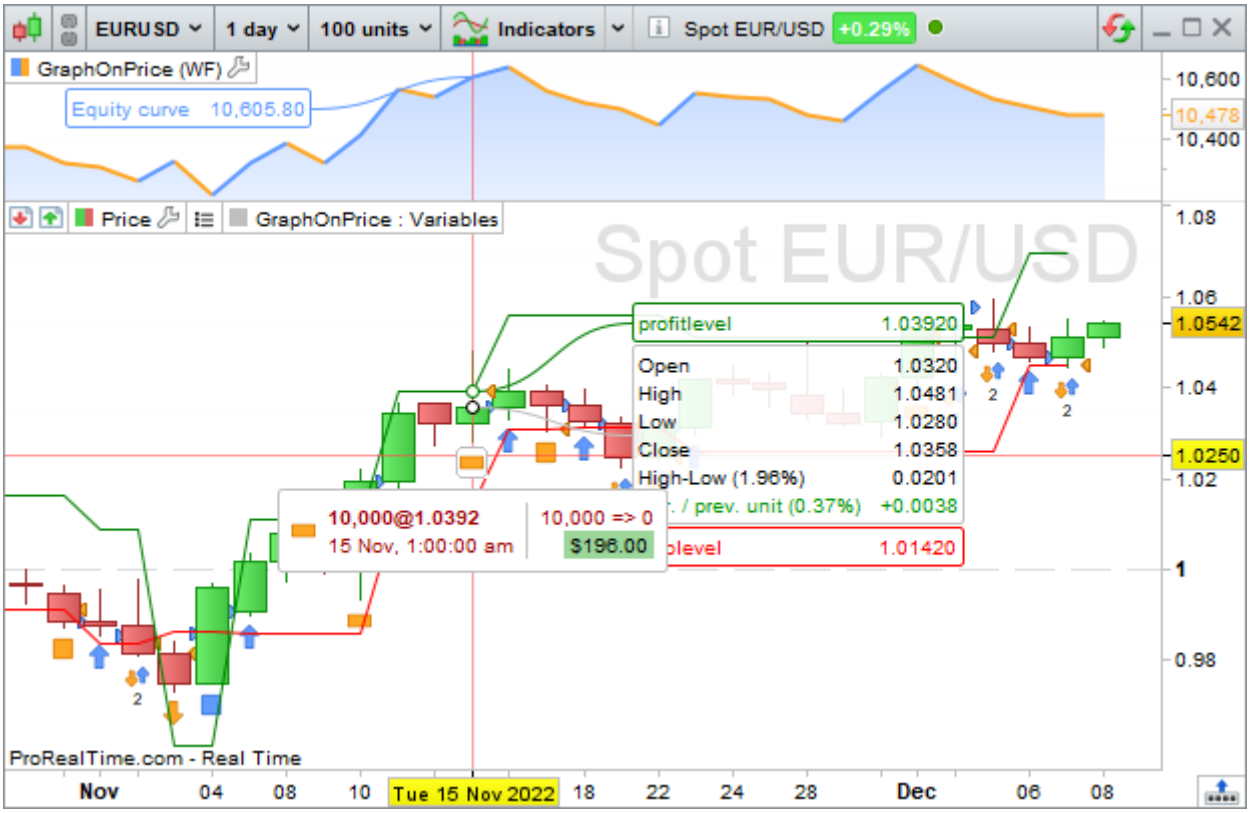
“a” is a whole number between 0 and 255 indicating the level of transparency. (0 for a completely transparent line, 255 for a completely opaque line)

A new chart panel appears under the equity curve panel of your backtest. This panel contains the values of **myvariable** at the close of each bar, as shown in the example below:



It is also possible to view one or more values used in your Backtest program directly on the price chart with the instruction `GRAPHONPRICE`.

In the following example, the Stop and Target levels are displayed directly on the price chart allowing you to follow the placement of these levels.



Note: The `GRAPH`, `GRAPHONPRICE` and `PRINT` instructions cannot be used in automatic trading mode.

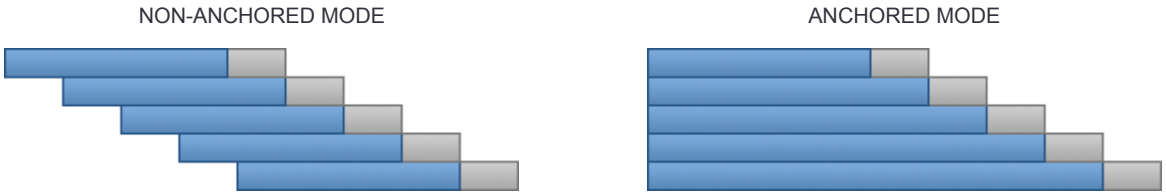
If a ProBacktest does not behave as intended, these 3 instructions are the best way to find the reason why. Each are best used in these case :

- `GRAPH` : When you are interesed in the full history of the variable to display and the variable is far from the price.
- `GRAPHONPRICE` : When you are interesed in the full history of the variable to display and the variable is close to the price.
- `PRINT` : When you are interested in variables only during specific events. Best used within a `IF` statement.

The Walk Forward method

Walk Forward analysis is an essential part of developing a trading strategy. It allows you to optimize a trading system and to validate its robustness and stability over time.

First, the walk forward analysis optimizes a set of variables on an initial period called "in-sample data" or the "optimization period", then it tests the best parameters on the following period called "out of sample data" or the "test period", and repeats the process by shifting forward the time windows. This automated process can be repeated as many times as necessary. The optimization periods (in-sample) may have a single starting point (Anchored mode), or different starting points (Non-anchored mode).



By comparing the results of these 2 periods, the method allows you to:

- Check that the performance of the strategy on the optimization periods (in-sample data: shown in blue) are consistent with the test periods (out-of-sample data: shown in grey). This avoids the risk of "over-fitting" by projecting the strategy on a period of time not used itself in the optimization.
- Check how the strategy behaves in changing market conditions
- Test the strength of the strategy on past data

To determine if a strategy is robust or not, we use the WFE ratio (walk forward efficiency ratio). The walk forward efficiency ratio is a qualitative indicator of the optimization process. It compares the annualized gain of the test period to the annualized gain of the optimization period. This measure of robustness is a critical part of walk forward analysis.

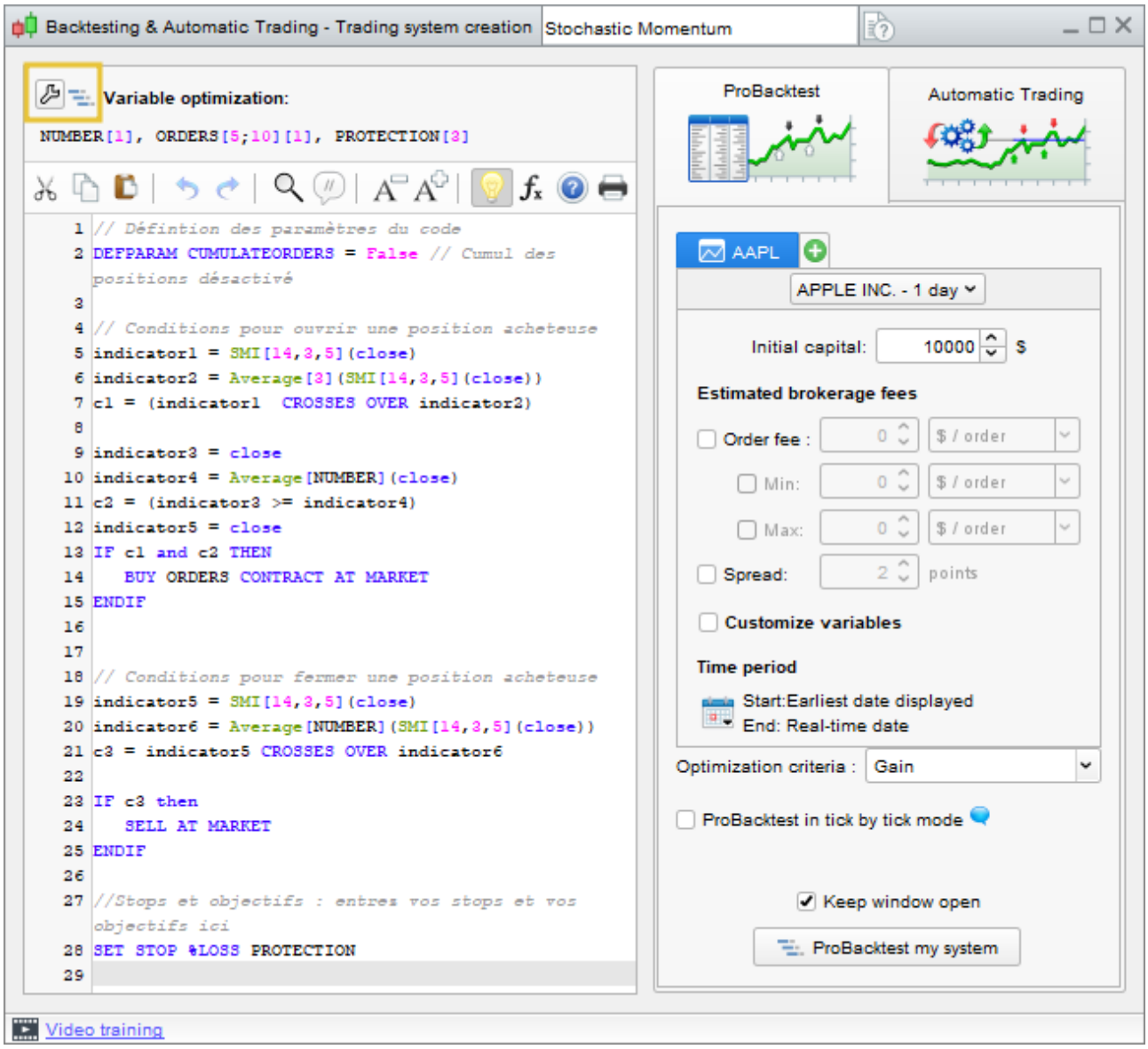
$$\text{WFE Ratio} = \frac{\text{Annualized gain of the test period}}{\text{Annualized gain of the optimization period}}$$

Annualized gains are the gains realized over a given period, expressed in annual terms. They let us compare the gains of different periods on a common basis. Typically, the optimization period (in-sample/blue in the image above) represents 70% of the total analyzed period, and the test period (out of sample/grey) 30%. To measure the reliability of the analysis, the results of the two periods need to be evaluated with a common basis. Studies of this method indicate that if at least 3/5 test periods display a WFE ratio greater than 50%-60%, the strategy can be considered to be robust.

After the walk forward method, it is interesting to analyze the results of several points. It is necessary to examine the regularity of gains from post-optimization periods. If the results show that the strategy shows a risk of being over-optimized (= the gains from test periods are much lower than those of optimization periods), the platform lets you adjust different parameters (variables, stop and target levels, trading periods) and launch a walk forward analysis again, as many times as needed, to obtain a robust strategy.

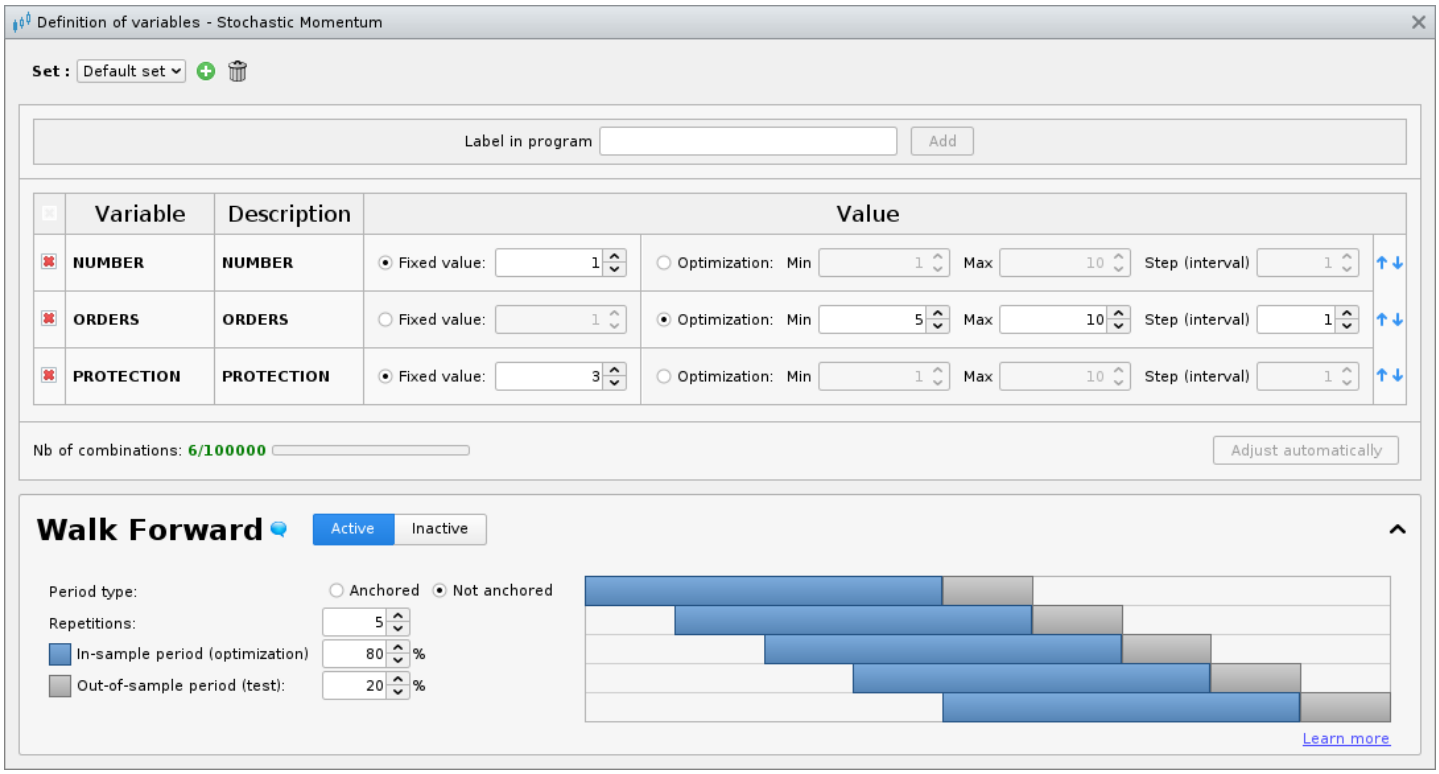
Practical example

This section shows how to optimize an existing trading system using the Walk Forward method.
First click on the wrench button at the top of the code editor window shown below.



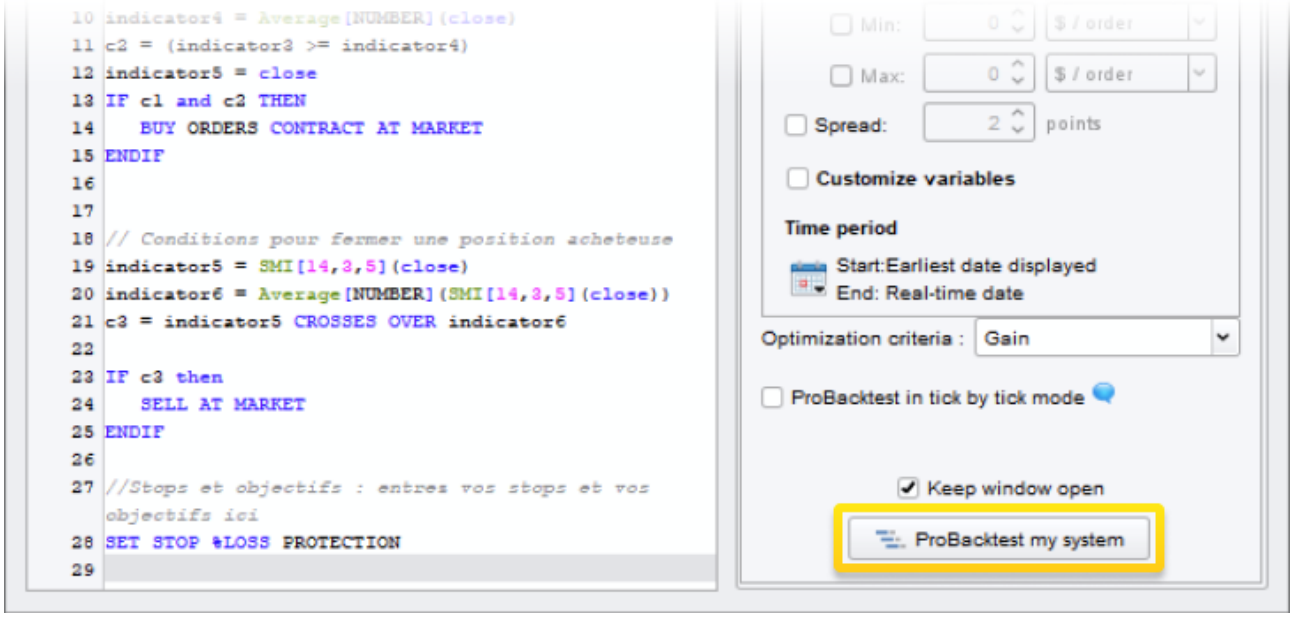
The window below will then appear. Once your variables are defined, you can activate the Walk Forward method by clicking on the "Active" button and choose the Walk Forward parameters. Select Non-Anchored mode for different starting points, or Anchored mode for a single starting point for all test repetitions.

You can also define the number of Walk Forward repetitions (one repetition consists in optimizing the parameters on the optimization period and testing the best set on the test period), and the ratio between the optimization period and the test period.



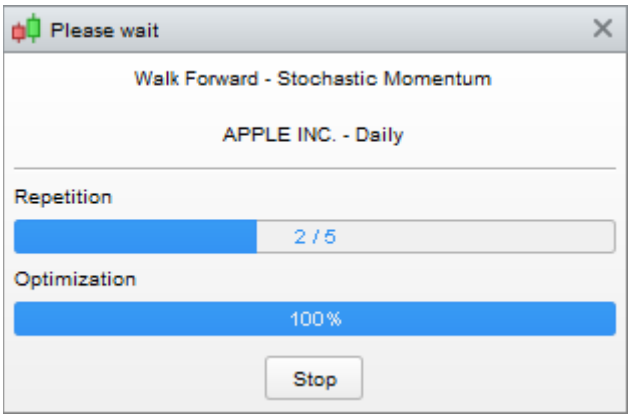
Research suggests an optimization period of 80% of the simulation period and a test period of 20%. It is also recommended to do several tests to increase the reliability of the analysis.

Once the parameters are defined, close the window above and launch the Walk Forward analysis by clicking on "ProBacktest my system".



During the Walk Forward analysis, the platform starts by optimizing the strategy on a first sample to determine the set of variables that generates the highest performance. Then the performance of this set is evaluated on an additional sample that was not included in the optimization sample. The process is repeated 5 times. The aim is to determine whether the optimal set of variables has generated consistent or greater performances in different market conditions.

Depending on the number of variables and repetitions, the calculation can be more or less complex. The backtest execution time will depend on this complexity.



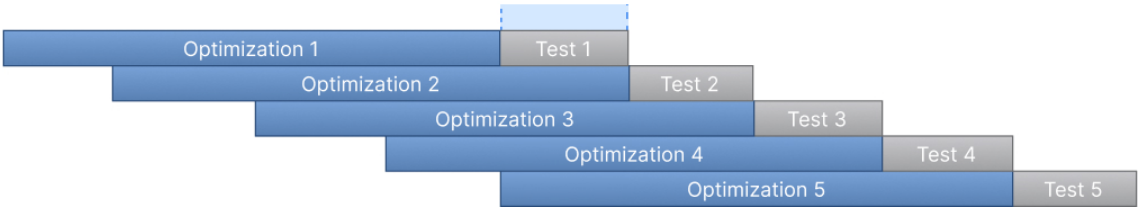
After calculation, a chart containing an equity curve will be displayed together with the detailed report of the system performance. In addition to the usual detailed report tabs, an additional Walk Forward tab compares the results for each test repetition.

Detailed report | ProBacktest (Walk Forward) | Stochastic Momentum | APPLE INC.

Walk Forward statistics | **Stochastic Momentum Daily** | Modify ProBacktest | Start: 24 Jul 2020 **[$10,000.00]** Current: 8 Dec 2022 **[$10,288.65]**

#	IS / OOS	Start	End	Gain	Annualized P...	WF Efficiency	Max drawdown	Max Runup	Variables
1	In-sample	24 Jul 2020	13 Aug 2021	$465.40	$441.63		-$269.70	$0.00	
	Out of sample	16 Aug 2021	16 Nov 2021	$13.70	$53.02	+12.00%	-$171.30	$130.10	
2	In-sample	28 Oct 2020	16 Nov 2021	$220.00	$209.55		-$278.70	$0.00	
	Out of sample	17 Nov 2021	23 Feb 2022	-$187.10	-$718.10	-342.68%	-$198.80	$121.90	
3	In-sample	3 Feb 2021	23 Feb 2022	$260.60	$246.86		-$226.60	$0.00	
	Out of sample	24 Feb 2022	31 May 2022	-$59.40	-$226.38	-91.71%	-$297.80	$202.50	
4	In-sample	11 May 2021	31 May 2022	$153.00	$144.93		-$440.30	$0.00	
	Out of sample	1 Jun 2022	2 Sep 2022	$191.50	$741.07	+511.32%	-$187.10	$379.60	
5	In-sample	16 Aug 2021	2 Sep 2022	-$16.05	-$15.26		-$273.30	$0.00	
	Out of sample	6 Sep 2022	8 Dec 2022	-$135.45	-$519.86	n/a	-$154.60	$19.15	

The statistics above relate to past data. Past performance is not indicative of future results.



The equity curve aggregates and displays the first optimization period (sample 1) and the 5 test periods (out of sample) on the chart.

When moving your mouse over the lower bands, the corresponding period will be highlighted with the parameters used, along with the performances of this time period. On the above screenshot, the mouse is over the first test period. The values shown above for the variables PROTECTION, ORDERS, and NUMBER gave the best results during the 1st optimization period (in-sample #1) and were then applied on the test period (out-of-sample #1), resulting in a gain of $1,159 during test period #1 (highlighted in the image above).

ProOrder Automatic Trading

Warning: If you start an automatic trading system using the ProOrder service, this service will automatically send signals, based on the parameters that you choose in order to send and execute the corresponding orders without any individual validation of each order required from you. Your system will be executed automatically even while your computer is off. It is your responsibility to make sure that you have correctly configured your system in a way that it does not lead you to unrecoverable losses or losses you are not ready to accept.

In all cases, ProRealTime will not be responsible for any losses taken due to the execution of your automatic system.

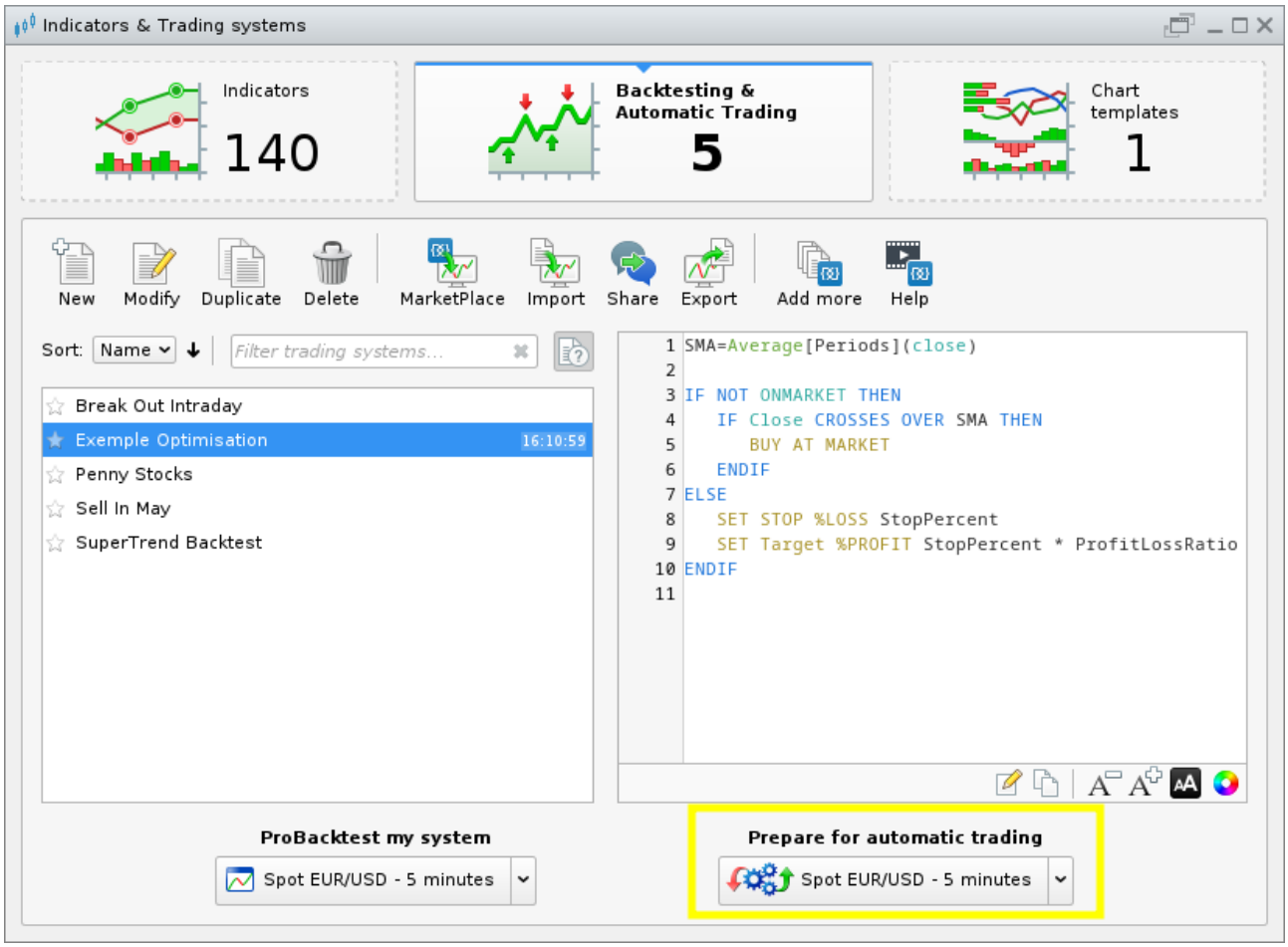
This part of the manual explains how to take trading systems which you have previously backtested and execute them as automatic trading systems.

- The first section explains how to send a trading system to ProOrder to prepare it for execution as an automatic trading system.
- The second section explains how to start a trading system and check the results.
- The third section explains the parameters of trading systems and their conditions of execution.
- The fourth section explains how manual and automatic trading systems coexist in the workstation.
- The fifth section explains implications of executing multiple automatic trading systems on the same instrument.
- The last section contains a list of indicators which cannot be used in automatic trading due to their method of calculation.

It is recommended that you read the entire manual before starting a trading system to learn about the execution of trading systems.

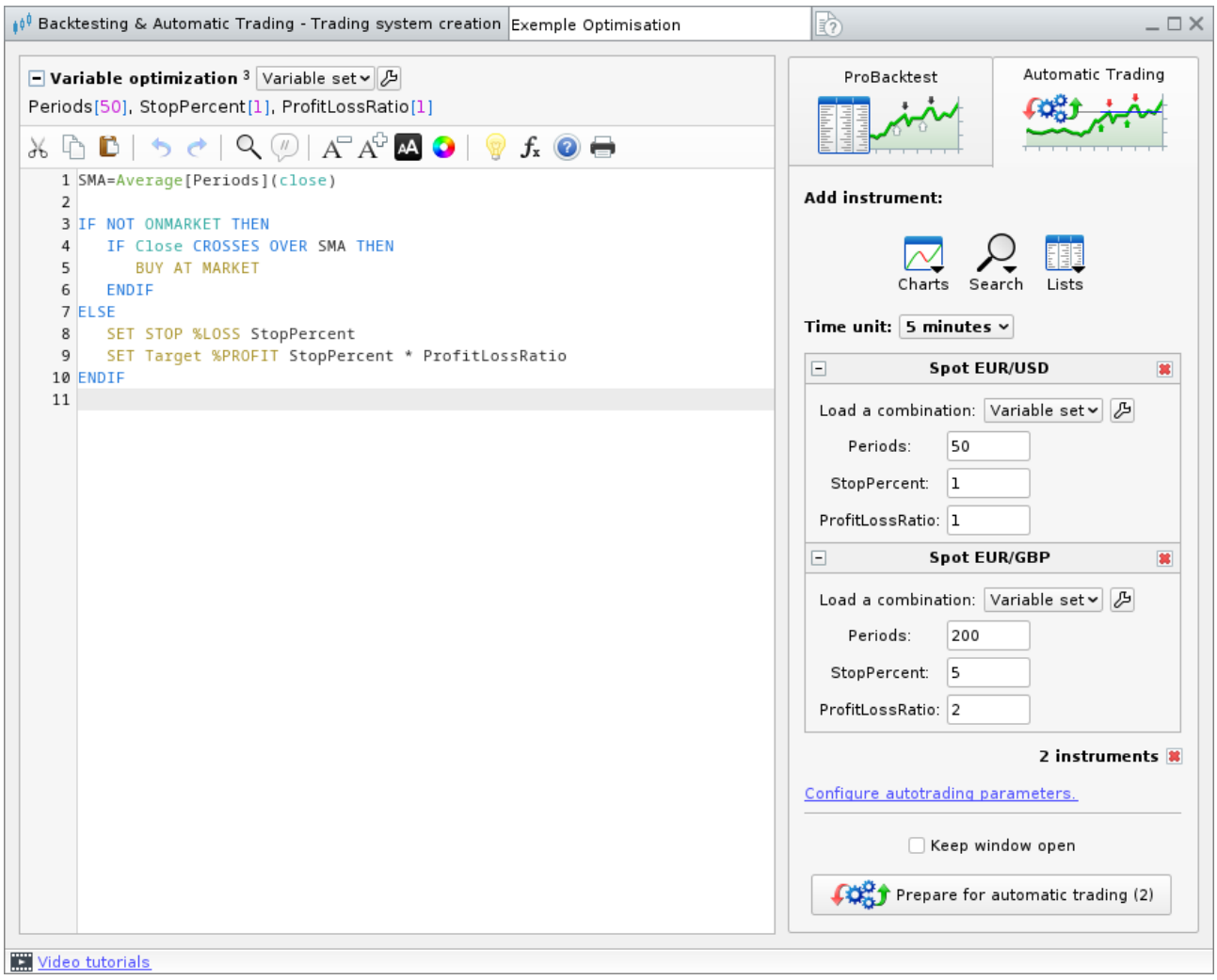
Prepare a trading system for automatic execution

Begin by opening the ProBacktest list from the Display menu:



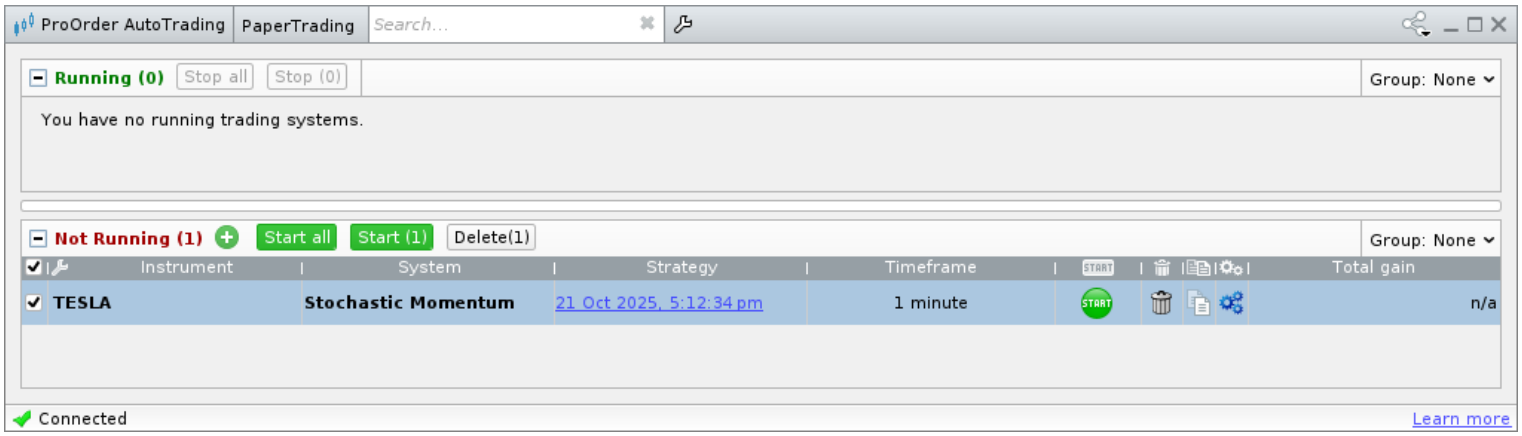
Select the system of your choice and click on "Prepare for automatic trading". The system will then appear in the ProOrder window applied on your chosen financial instrument.

It is also possible to create a trading system from the code editor. It also allows the creation of multiple systems at once, on different instruments and with different variable sets :



How to start a trading system in ProOrder and check the results

Once you have added a trading system to ProOrder, you can define the maximum position size for this system then begin trading with it by pressing the start button:



Be sure to read the content of the confirmation pop-up: you will be asked to confirm the execution of the trading system :

It is important to understand that the maximum position size setting has priority over the buy/sell quantities defined in the trading system's code. The maximum position size for futures and forex is defined in number of contracts or number of lots. This lot size is defined by the broker and therefore cannot be changed.

For example, if your code has an instruction to buy 3 lots and the maximum position size limit is set to 1, the order to buy 3 lots will be ignored.

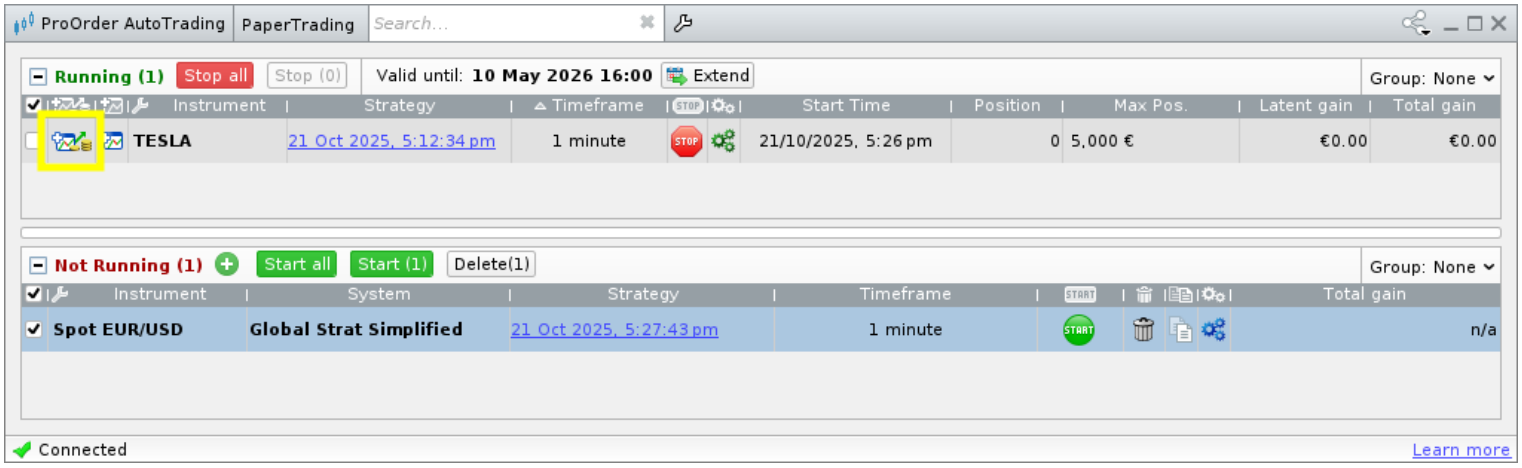
Similarly, if your code calls for buying 1 lot and then shorting 3 lots, the shorting order will be ignored and you will remain in a buy 1 lot position. It is recommended that you always check the maximum position size setting before starting a trading system.

For equities, the maximum position size is expressed in cash amount (brokerage fees not included).

Once launched, the system will be displayed in the "Running" part of the window, as shown below.

Note: We strongly recommend that you carefully read the "Conditions of Automatic Trading" information, accessible via the link in the confirmation pop-up. Those conditions will allow you to fully understand how a system behaves in different cases and for multiple configurations.

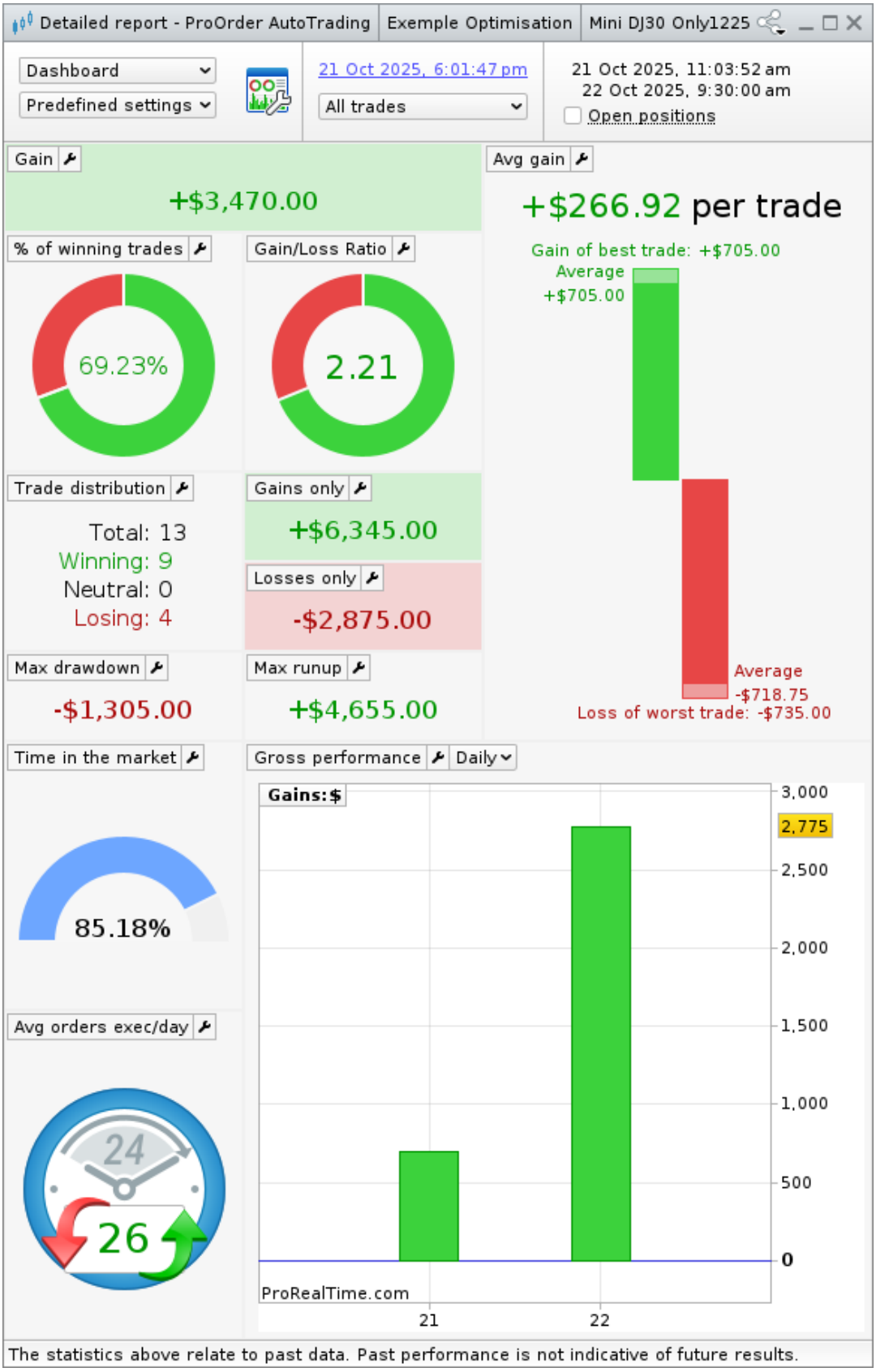
Once a trading system has started, its position, latent gain and total gain will be shown in the ProOrder window. It is possible to click on the link in the "version" section to see a copy of the code of this system.



You can also click the button shown in yellow to see the equity curve of the system and a detailed report on its performance.

Here is an example of an equity curve of a running system

Here is an example of a detailed report of a running system.

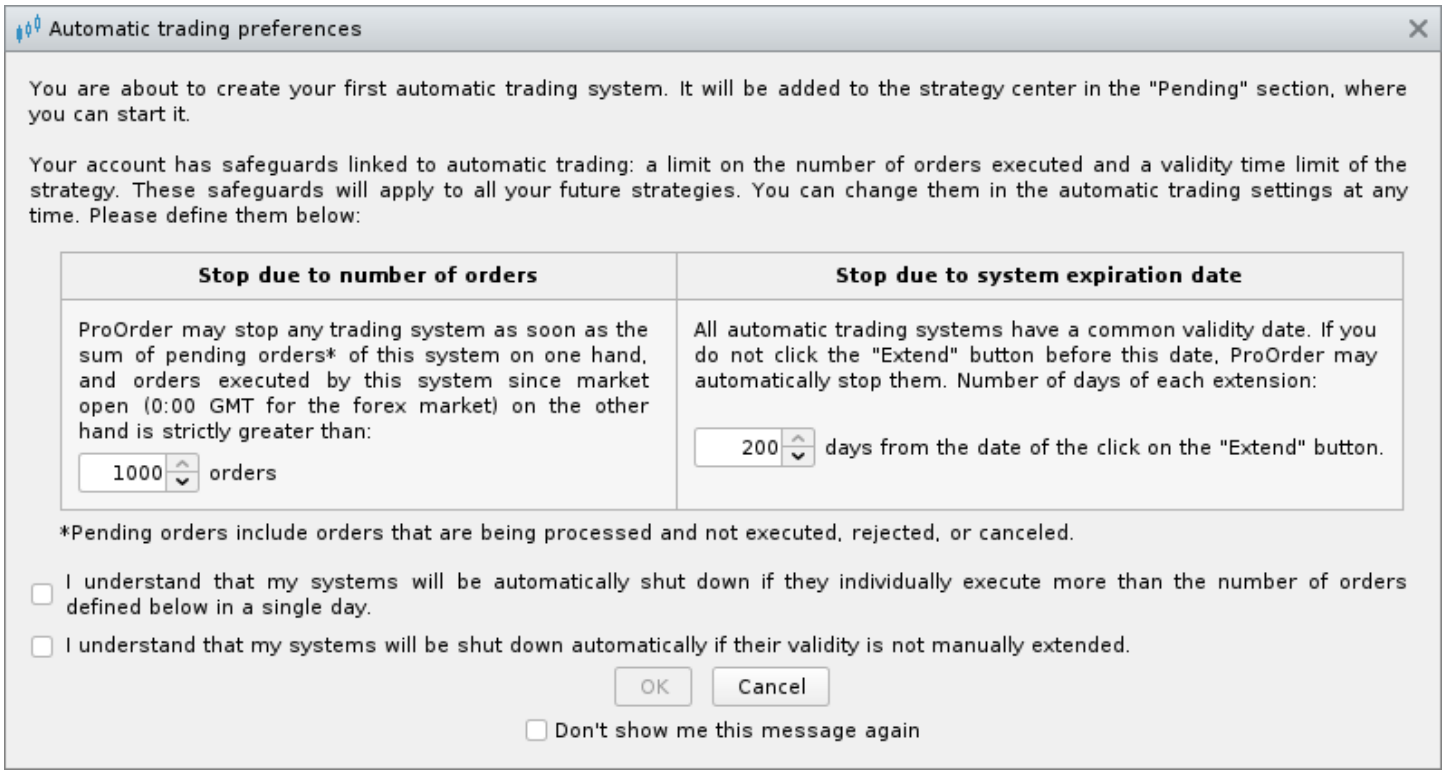


Note: The gain in the section “Statistics of closed positions” may be different from the value of the equity curve because the system is still running and the equity curve takes into account positions that are still open, which is not the case with the detailed report.

Automatic trading parameters and conditions of execution

Trading system parameters

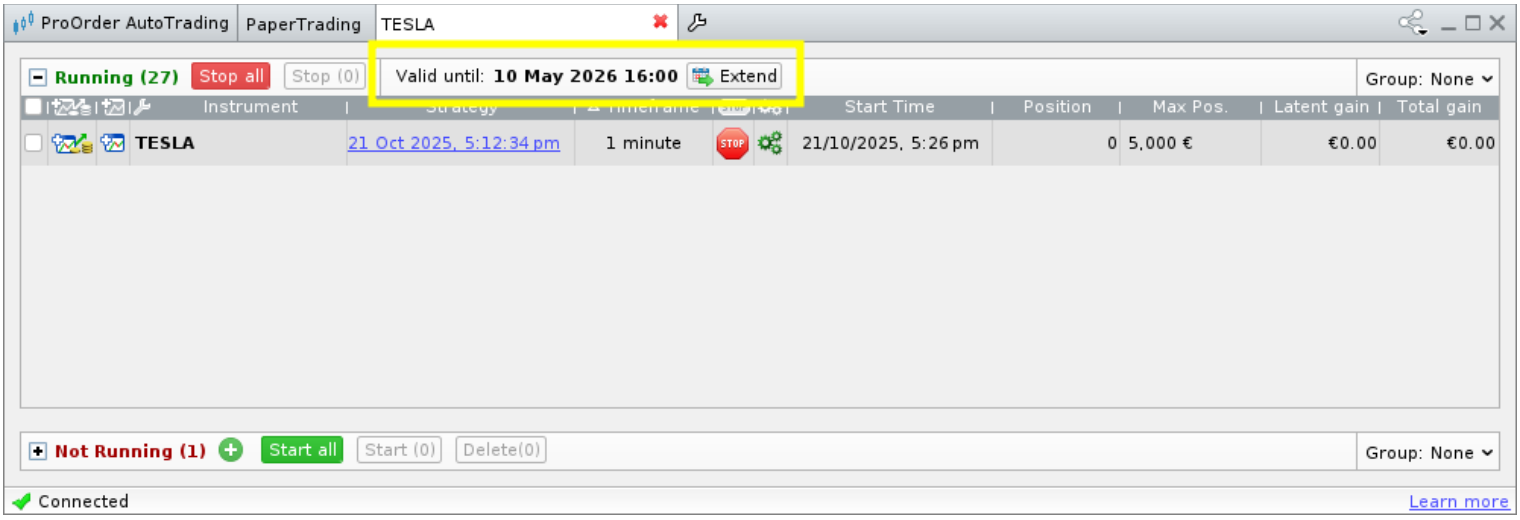
When creating your first Automatic trading system you will be shown this popup asking to set your automatic trading preferences



Automatic stop of trading systems

Validity date: All running trading systems have a common validity date. If you do not click the "Extend" button before this date, ProOrder may automatically stop them. You can view the validity date in the ProOrder window (expressed in your computer's time zone) and extend your trading systems validity via the "Extend" button when a trading system is running:

The extending duration depends on your automatic trading preferences set up in the window from the previous section. You can modify them at any time via the platform's automatic trading settings. The wrench in the title bar is a shortcut that allows you to access these settings with a single click.



Number of orders placed: ProOrder may stop any given trading system as soon as the sum of pending orders placed by this system on one hand and number of orders executed by this system since market open on the other hand (0:00 GMT for the forex market) of this system is greater than or equal to the quantity chosen in the "Automatic trading" tab of the "Trading preferences" window. A pending order is an order that was sent to the broker and not executed, rejected, or canceled.

For example, each "`Set stop`" or "`Set Trailing stop`" or "`Set target`" instruction as long as the corresponding order has not been canceled, rejected, or executed.

In addition, 3 different limit orders or 3 different stop orders that have not been canceled, rejected or executed will count as 3 pending orders. This is true if the 3 orders are on the same price level or on different price levels.

For example, if you choose a maximum level of 8 orders and since the market open 5 orders have been executed by a given trading system and this system has 2 pending orders (one "`set target`" and one "`set stop`") and the system needs to send an additional order to the market: this 8th order will not be sent (5+2+1 reaches the maximum level), this trading system will be stopped with its pending orders cancelled first, and then its position closed.

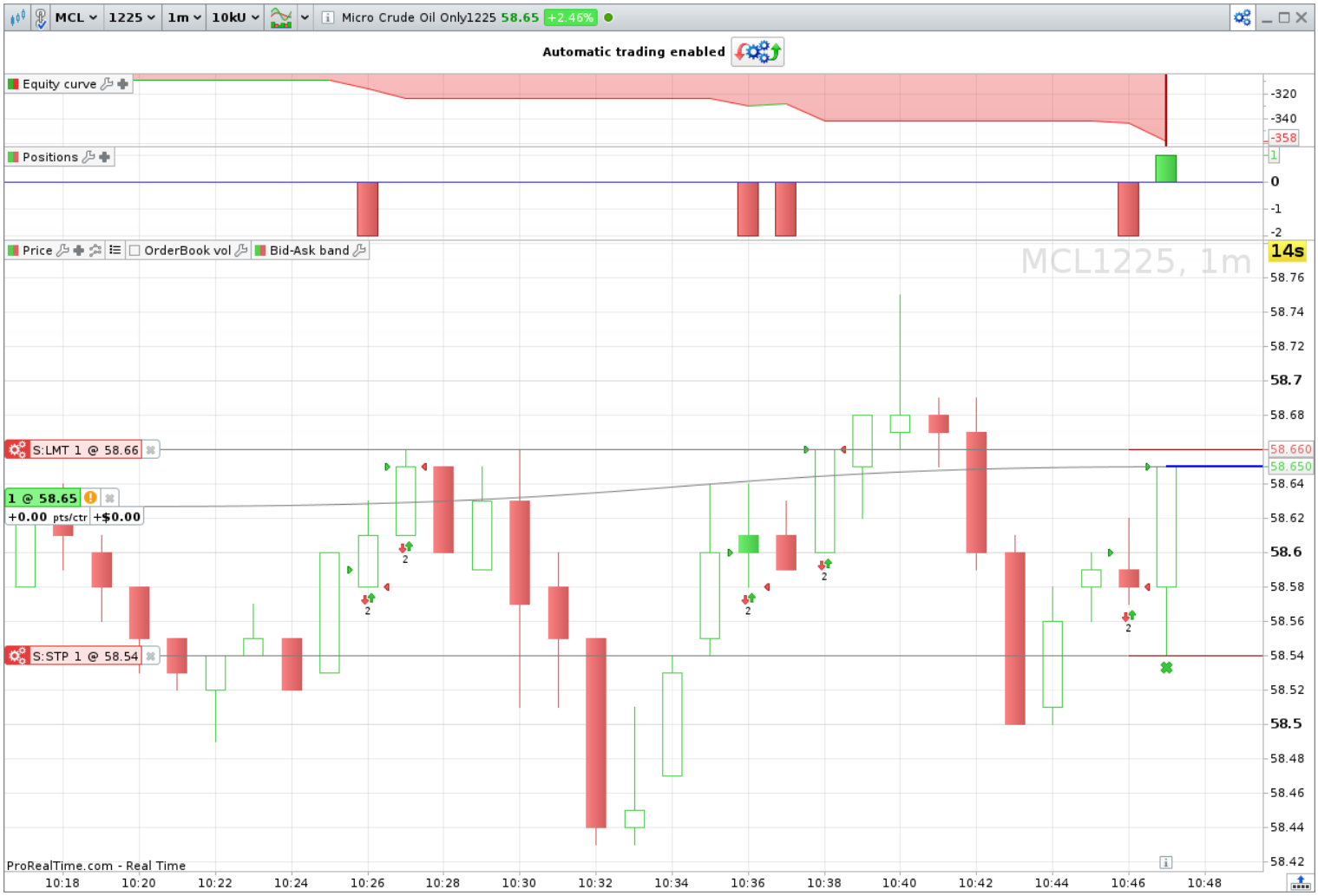
It is possible increase this parameter while you have trading systems running.

Order rejection: ProOrder may stop any given trading system if too many orders of this trading system are rejected.

Co-existence of manual and automatic trading in the workstation

Note : This section is not applicable to all brokers.

When a trading system is running on a financial instrument, manual order placement is no longer available. However, manual trading is still possible on other instruments. A specific button appears instead of the manual order bar on the instrument in question, as illustrated below:



Running multiple trading systems on the same security

Note : This section is not applicable to all brokers.

If you run multiple trading systems on the same instrument, your overall net position is determined by all your systems. Let's take an example where you have 2 trading systems. One places a buy order for 1 lot while the other places a sell order for 1 lot. Your overall net position will then be 0. Only the overall net position is displayed in case several systems are running on the same instrument.

When you display a system's gain & loss curve, you also get its position histogram. Unlike the overall net position, which aggregates all the data from all the systems in the instrument, this histogram represents the positions taken by a single system. The position levels displayed may therefore be different, as shown in the graph below.

When you run multiple trading systems on the same instrument, each system reports its own positions, orders, trades and earnings independently. Therefore, instructions of type `LONGONMARKET` or `SHORTONMARKET` inform about the Long or Short status of the concerned trading system.

Your overall net position may differ from the position of a given system. The same applies to variables such as `COUNTOFLONGSHARES`, `COUNTOFSHORTSHARES`, `COUNTOFPOSITION`, `POSITIONPRICE`, `STRATEGYPROFIT`, `TRADEINDEX`, `TRADEPRICE` and `POSITIONPERF`, which are specific to a given trading system.

Indicator restrictions

The following indicators may not be used for automatic trading because their mode of calculation does not allow for real-time usage:

- `ZigZag`
- `ZigZagPoint`
- `DPO`

Note concerning personalized time zones and trading hours

When a trading system is sent to ProOrder, the time zone and trading hours which were defined for the market of the instrument are associated with the strategy. These parameters are applied at each launch of the strategy. To modify the time zone and trading hours of a strategy, you need to delete the strategy from ProOrder, modify these parameters in the Options > Time Zones & Trading hours menu, then send the strategy to ProOrder again.

See the section **Customization of trading hours for backtesting** for instructions concerned by the time zone of the chart and for an explanation of customized trading hours.

Annex A: Display of trading system results

A trading system's results are displayed in 3 complementary ways.

Equity Curve

The equity curve of a backtest shows the profit and loss of the trading system or backtest:

- The **horizontal blue line** represents the initial capital of the system. In the case of automatic trading, this line is always set to 0.

- The **color shading of the equity curve** is blue if the performance is positive (gain since the starting point). It is orange if the performance is negative.
- The **line of the equity curve** is blue when it has increased from the previous point and orange when it has decreased.

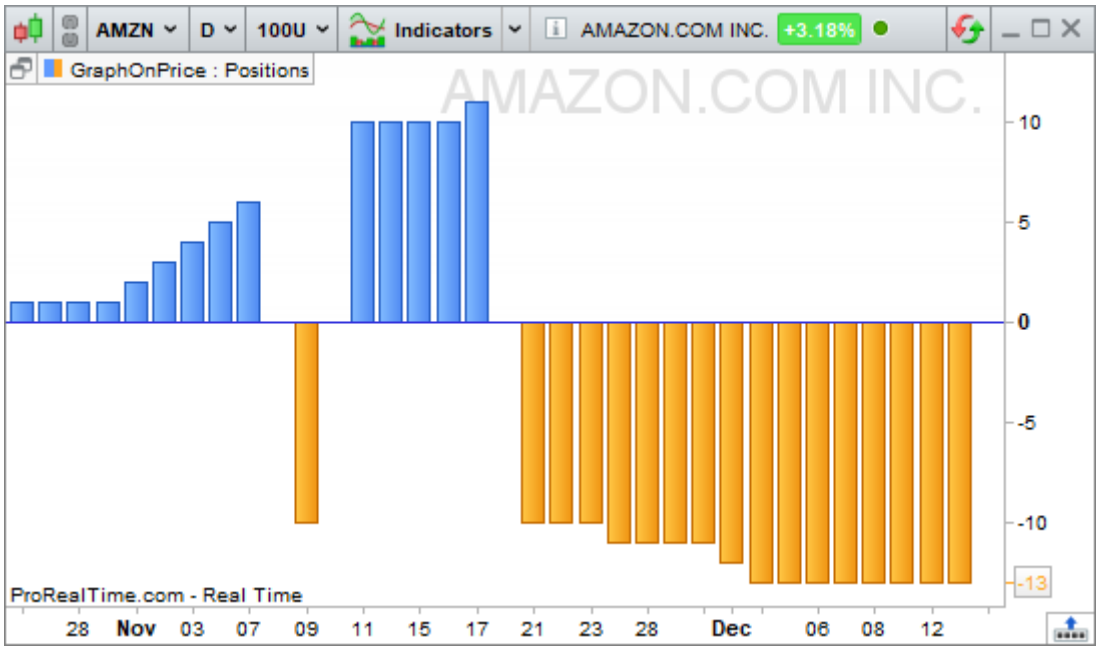
Positions chart

The positions histogram allows you to show in histogram form the evolution of your positions during the trading system simulation.

- A blue bar indicates an open long position.
- An orange bar indicates an open short position.
- No bar indicates no open position.

Several consecutive bars of the same color indicate the position(s) is still open.

On the vertical axis on the right-side of the chart, you will see how many positions you have open currently.



Detailed report

The detailed report lets you view the statistics of your trading system and the details of each position and order:

Detailed report - ProOrder AutoTrading | Exemple Optimisation | Mini NASDAQ100 Only1225

Statistics of closed trades | 21 Oct 2025, 5:44:58 pm | All trades

Start: 21 Oct 2025, 11:03:49 am
Current: 5 Nov 2025, 10:36:00 am
Open positions

Profits and losses	All trades	Long trades	Short trades
Gain	+$7,930.00	+$7,930.00	+$0.00
% Gain	∞%	∞%	n/a
Gains only	+$84,525.00	+$84,525.00	+$0.00
Losses only	-$76,595.00	-$76,595.00	+$0.00
Gain/Loss ratio	1.1	1.1	n/a
Avg gain	+$25.09	+$25.09	n/a
Avg gain of winning trades	+$515.40	+$515.40	n/a
Avg loss of losing trades	-$503.91	-$503.91	n/a
Gain of best trade	+$525.00	+$525.00	+$0.00
Loss of worst trade	-$570.00	-$570.00	+$0.00
Max runup	+$19,455.00	+$19,455.00	+$0.00
% Max runup	-597.7%	-597.7%	n/a
Max consecutive wins	6	6	0
Max drawdown	-$10,830.00	-$10,830.00	+$0.00
% Max drawdown	66.85%	66.85%	n/a
Max consecutive losses	7	7	0
Standard deviation on P&L	+$510.17	+$510.17	n/a
Sharpe ratio	-0.06	-0.06	0
Annualized rate of return	0%	0%	0%
AHPR	-∞%	-∞%	0%
GHPR	∞%	∞%	0%
Risk/Reward ratio	1.02	1.02	n/a
Z-Score	0.65	0.65	0
Brokerage fees	+$0.00	+$0.00	+$0.00

Trades	All trades	Long trades	Short trades
Nbr trades	316	316	0
% of winning trades	51.9%	51.9%	0%
Winning trades	164	164	0
Even trades	0	0	0
Losing trades	152	152	0

Risks	All trades	Long trades	Short trades

Time	All trades	Long trades	Short trades

The statistics above relate to past data. Past performance is not indicative of future results.

The detailed report is presented in a separate window, in which you can choose between several methods of displaying your strategy data:

The **"Statistics of closed positions"** display gives an exhaustive view of performance of your trading system (Gain or loss, number of winning trades, etc) and indicators of risk such as max drawdown. Note that this tab does not include analysis of currently open positions at the time the report is generated (only closed positions are taken into account). Here is the list of statistics:

- **"Gain"** is the gain or loss realized by the trading system. The calculation formula is:

  Gain = Final capital – Initial capital

  This statistic lets you evaluate the absolute gain potential with the trading system defined for the historical period tested and for each variable combination.

  Note: Brokerage fees as defined in the “Brokerage parameters” section are taken into account in this calculation.

- **"%Gain"** is the gain or loss in %. The calculation formula is:

  %Gain = 100 x Profit / Initial capital

- **"Nb positions"** indicates the number of positions opened during the backtest.

- **"% winning"** indicates the % of winning positions. It is calculated as:

  % winning = (100 x number of winning positions) / Number of positions

- **"Avg gain per position"** is the average gain per position. It can be useful to determine efficiency of orders placed. Average gain per position is particularly important when creating a trading system which places a low number of orders. It is defined as:

  Avg gain per position = Gain / Number of positions

- **"Profit best position"** is the maximum gain on a given position and **"Loss worst position"** is the highest loss on a given position opened since the beginning of the trading system. **"Standard deviation on P&L"** is the standard deviation of results of each position.

- **“Drawdown”** is defined as the distance between a given point and the highest point before it on the equity curve:

  $$DD(n)= Max\ t€[0\ ;n]\ P(t) – P(n)$$

- **"Max drawdown"** is calculated as the largest drawdown over the entire history of the trading system.

  $$MaxDD(N) = Max\ n€[0:N]\ (\ Max\ t€[0;n]\ P(t) - P(n)\ )$$

- **“Runup”** is defined as the difference between a given point and the lowest point before it on the equity curve:

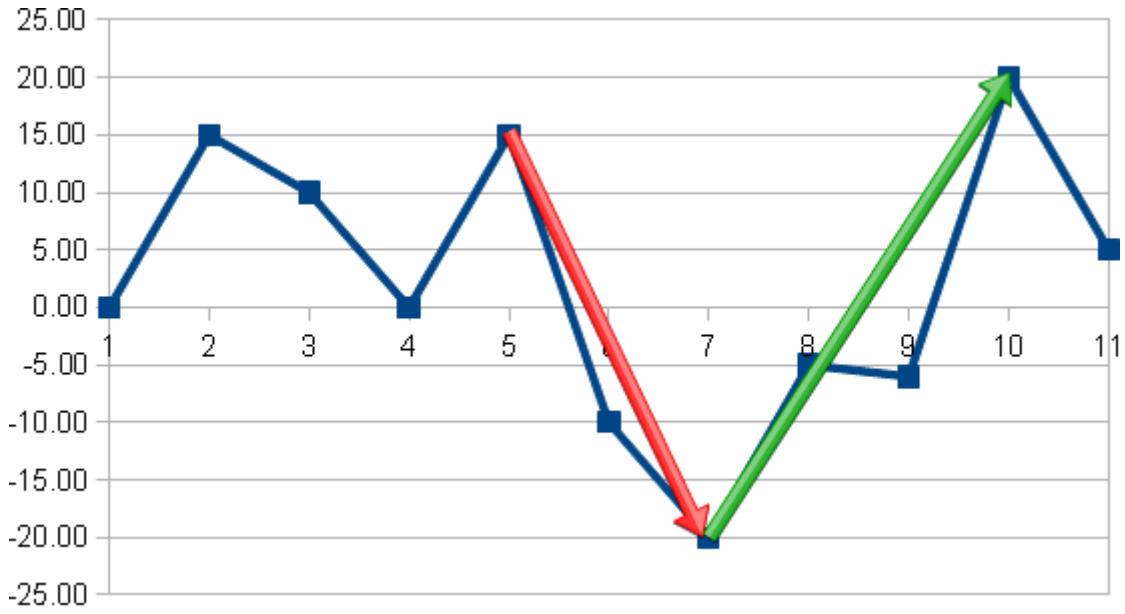
  $$RU(n)= P(n) – Min\ t€[0;n]\ P(t)$$

- **"Max runup"** is calculated as the maximum of this value over the entire history of the trading system.

  $$MaxRU(N) = Max\ n€[0;N]\ (\ P(n) – Min\ t€[0;n]\ P(t)\ )$$

Example:

BARINDEX	P & L	DRAWDOWN	RUNUP
1	0.00	0.00	0.00
2	15.00	0.00	15.00
3	10.00	5.00	10.00
4	0.00	15.00	0.00
5	15.00	0.00	15.00
6	-10.00	25.00	0.00
7	-20.00	35.00	0.00
8	-5.00	20.00	15.00
9	-6.00	21.00	14.00
10	20.00	0.00	40.00
11	5.00	15.00	25.00
	Max:	**-35.00**	**40.00**



"% Max risk exposure": Exposure to risk is the relationship between the maximum loss possible for the position and the current amount of capital. The % max risk exposure is then maximum of this value expressed as a percent. The calculations are as follows for stocks, futures and Forex:

- Stocks:

% Max risk exposure = Max position (quantity * average price / capital) * 100

- Futures:

% Max risk exposure= Max position (quantity * deposit / capital) * 100

- Forex:

% Max risk exposure = Max position (quantity * average price * leverage / capital) * 100

Similarly, **"% avg risk exposure"** is the average percent risk exposure.

"Brokerage fees" count total brokerage fees of each order since the beginning of the trading system. These brokerage fees are defined in the settings in the case of a backtest.

“% time in the market” is calculated as the number of bars with a position open divided by the number of bars of the trading system.

The other selectable display methods give information about orders placed and positions opened and closed closed during the execution of the trading system.

- In **"Order list"** you will find a list of all the orders placed by including their date & time, direction, quantity and price. The times displayed are in the timezone of the instrument.
- In **"Closed positions list"** you will find information about the positions taken by your trading system (long or short, duration in number of bars, performance of each position, opening date and closing date. If there is a position still open at the time the report is generated, it will not be included in this list. In the case of a backtest, if you want to close all positions at the end of the backtest, choose a fixed ending date instead of the real-time date.
- The "**Dashboard"** view gives you a customizable overview of your trading system's performance.
- The "**Advanced Chart**" view allows you to explore graphical representations of the different types of performance indicators in your strategy.

Annex B: Detailed examples of codes

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

You can visit our ProRealTime community on the **ProRealCode forum** to find **online documentation** and many examples.

Heiken Ashi trading system

This trading system generates a buy signal when a green Heiken Ashi candle appears after a red one.

A sell signal is given if a red Heiken Ashi candle appears after a green one.

This backtest reconstructs the Heiken Ashi view from normal candlesticks. It must be applied to a chart using the normal candlestick style (not the Heiken Ashi candlestick style).

```
ONCE IsFirstCandle = 1
XClose = TotalPrice
IF IsFirstCandle THEN
   XOpen = (Open + Close) / 2
   IsFirstCandle = 0
ELSE
   XOpen = (XOpen[1] + XClose[1]) / 2

   IF XClose > XOpen AND XClose[1] < XOpen[1] THEN
       BUY 1 SHARES AT MARKET
   ENDIF

   IF XClose < XOpen AND XClose[1] > XOpen[1] THEN
      SELLSHORT 1 SHARES AT MARKET
   ENDIF

ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

Simple breakout range with trailing stop

This is a basic breakout intraday trading system that takes only long positions. The initial range is determined by the highest and lowest points of the first 2 candlesticks of the day. A support is defined at the lowest point and a resistance at the highest point.

If price crosses over the resistance and the 10-period moving average is increasing, a long position is taken.

A profit target of 1% is defined.

A protection stop is set at the level of the support, if price reaches this level, the position will be closed with a stop order.

The position is also closed at 5 PM local market time in order to not keep any position overnight. Access to intraday data is necessary to test this trading system.

```
DEFPARAM CumulateOrders = False
DEFPARAM FlatAfter = 170000

MM = Average[10](close)
MyTarget = 1

IF INTRADAYBARINDEX = 2 THEN
   MyResistance = highest[2](high)
   MySupport = lowest[2](low)
ENDIF

IF MM > MM[1] AND close CROSSES OVER MyResistance THEN
   BUY 1 SHARES AT MARKET
ENDIF

SELL AT MySupport STOP
SET TARGET %Profit MyTarget
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

Smoothed stochastic trading system

This trading system is based on a smoothed stochastic applied to median price (`sSto`) and an exponential moving average of this value (`EMA`). The trading system is long when sSto is above EMA and is short otherwise.

```
DEFPARAM CumulateOrders = False

sSto = SmoothedStochastic[14,3](MedianPrice)
EMA = ExponentialAverage[9](sSto)

StopLimit = 1

IF sSto >= EMA THEN
   BUY 1 SHARES AT MARKET
ELSE
   SELLSHORT 1 SHARES AT MARKET
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

Swing Trading, ADX and Moving Average

This backtest uses the ADX indicator and its position with regard to the level 30 since at least 5 days, with the goal of reducing false signals and minimizing risk. It must be executed on a daily timeframe.
The trading system has many conditions that limit the number of trading opportunities.

```
DEFPARAM CumulateOrders = False
MyADX12 = ADX[12]
ADXperiods = 5
MyMM20 = Average[20](Close)

// BUY : ADX 12 must be greater than 30 for at least 5 bars
Condition1 = LOWEST[ADXperiods + 1](MyADX12) > 30
// If the 20-period moving average of the current period is between the high and low of the current period and the moving average of the previous period is between the high and low of the previous period
Condition2 = High > MyMM20 AND Low < MyMM20 AND High[1] < MyMM20[1] AND Low[1] < MyMM20[1]
// If the high of the current day is higher than the high of the previous day
Condition3 = Dhigh(0) > Dhigh(1)
IF Condition1 AND Condition2 AND Condition3 THEN
   BUY 1 SHARES AT MARKET
ENDIF

// SHORT : ADX 12 is greater than 30 since at least 5 bars
Condition4 = Condition1
// If the 20-period moving average of the current period is between the high and low of the current period and the moving average of the previous period is between the high and low of the previous period
Condition5 = High > MyMM20 AND Low < MyMM20 AND High[1] > MyMM20[1] AND Low[1] > MyMM20[1]
// If the low of the current day is lower than the low of the previous day
Condition6 = Dlow(0) < Dlow(1)
IF Condition4 AND Condition5 AND Condition6 THEN
   SELLSHORT 1 SHARES AT MARKET
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

Trading system with a position counter

Inverse Fisher transform applied to RSI

This trading system uses the “Inverse Fisher Transform RSI” to place buy or sell orders.

It enters a long position when the Inverse Fisher Transform RSI crosses over 50 and exits a long position when it crosses under 80.

It enters a short position when Inverse Fisher Transform RSI crosses under 50 and exits short when Inverse Fisher Transform RSI crosses over 20.

This trading system can be backtested on futures in 1-hour view or stocks in daily view.

```
// Inverse fisher transform applied to RSI.
// Parameters: n = number of bars for calculation of the RSI.
n = 10
Ind = RSI[n](Close)
x = 0.1 * (Ind - 50)
y = (EXP (2 * x) - 1) / (EXP (2 * x) + 1)
myInverseFisherTransformsRSI = 50 * (y + 1)

IF (myInverseFisherTransformsRSI CROSSES OVER 50) THEN
   BUY 1 SHARES AT MARKET
ENDIF

IF (myInverseFisherTransformsRSI CROSSES UNDER 80) THEN
   SELL AT MARKET
ENDIF

IF (myInverseFisherTransformsRSI CROSSES UNDER 50) THEN
   SELLSHORT 1 SHARES AT MARKET
ENDIF

IF (myInverseFisherTransformsRSI CROSSES OVER 20) THEN
   EXITSHORT AT MARKET
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

Trading system with TRADEINDEX – find inside bar

The following example trading system is based on a frequently used price pattern called an “Inside Bar” and based on 2 candlestick forms:

- The first form occurs if the range of the 2nd candle preceding the current candle is greater than the range of the candle preceding the current candle. The candle preceding the current candle must be white (close > open). In this case, a long position is taken.
- The second form occurs if the range of the 2nd candle preceding the current candle is lower than the range of the candle preceding the current candle and the candle preceding the current one is black (close < open). In this case we take a short position.

All positions are systematically closed 3 bars after they are opened.

```
DEFPARAM CumulateOrders = False
Condition1 = High[1] >= High AND Low[1] <= Low
Condition2 = High[1] <= High AND Low[1] <= Low
Condition3 = Close > Open
Condition4 = Close < Open

IF Condition1 AND Condition3 THEN
   BUY 1 Share AT MARKET
ENDIF

IF LONGONMARKET AND (BarIndex - TRADEINDEX) = 3 THEN
   SELL AT MARKET
ENDIF

IF Condition2 AND Condition4 THEN
   SELLSHORT 1 share AT MARKET
ENDIF

IF SHORTONMARKET AND (BarIndex - TRADEINDEX) = 3 THEN
   EXITSHORT AT MARKET
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

Money management strategies

A backtest's result can be improved by using money management strategies.

These strategies are sometimes formalized in “martingales”. They are aimed at optimizing the mathematical expectancy of a trading system. The expectancy is the average win or loss for each transaction if many transactions are done. This implies being able to estimate the probability of a transaction being winning and the probable amount of profit or loss.

In order to implement a martingale, it can be very useful to have stop loss, take profit and inactivity orders coded directly in your trading system, so that they are fully customizable, and to have sub-strategies allowing us to dynamically manage a position's size.

Protection stops and profit targets

For more information about protection stops, trailing stops and profit targets, see **Stops and targets**

Exit the market after a set amount of time

The code below allows you to integrate a market order into your system to close a position after a set amount of bars in case it has not been closed by other conditions of your system before that. In the following example, positions are closed after 10 candlesticks.

```
ONCE Count = 10
// Choice of the number of bars after which the position will be automatically closed
IF ONMARKET AND (BARINDEX - TRADEINDEX + 1) > Count THEN
   IF LONGONMARKET THEN
      SELL AT MARKET
   ENDIF

   IF SHORTONMARKET THEN
      EXITSHORT AT MARKET
   ENDIF
ENDIF
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

Cumulate orders – Adding to an existing position with use of a position counter

An example of cumulating orders to increase position size is given in the following section.

To enable cumulating orders, enter the command **"DEFPARAM CumulateOrders=True"** at the beginning of the program.

This trading system uses the first condition based on the RSI to take the initial position only. Additional shares are added each bar where open is greater than previous close up to a maximum of 3.

"**Countofposition**" is used in this code to limit the maximum position size to three.

```
DEFPARAM CumulateOrders = True

// Buy 1 when RSI < 30 and there is no position already open
IF RSI[14](Close) < 30 AND NOT ONMARKET THEN
   BUY 1 SHARES AT MARKET
ENDIF

// If there is an open long position and open > previous close, each time we buy an additional quantity up to a maximum of three.
IF LONGONMARKET AND COUNTOFPOSITION < 3 AND Open > Close[1] THEN
   BUY 1 SHARES AT MARKET
ENDIF

// When price crosses under a simple moving average, close the position
IF Close Crosses Under Average[14](Close) THEN
   SELL AT MARKET
ENDIF
```

With these tools, we can now look at martingales. Here are some of the most popular. These techniques can be added to any trading system.

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

The classic martingale

The classic martingale doubles the position size when it loses in order to make up for the loss if the next trade is a winner. The disadvantage of a trading system like this is that successive losses make it more and more difficult (or impossible) to double your position. Starting with 1,000€ for example, if you lose 5 times in a row, you would need 1,000 x 32 = 32,000€ to continue with this trading system.

As a result, trading systems with the martingale may be more adapted to trading stocks than Futures or Forex because the initial capital required to trade may be much larger in these 2 types of markets.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE ExitIndex = -2
// Initial position size of 1.
//**********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex

//***********Code to insert at the end of the trading system**********//
IF BarIndex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      OrderSize = OrderSize * 2
      // Double OrderSize if the last position was a losing position.
   ELSIF PositionPerf(1) > 0 THEN
      OrderSize = 1
      // Reset position size to 1 if the last trade was a winning trade.
   ENDIF
ENDIF
//**********************//
// The position size must be determined depending on the variable OrderSize in the entire
code.
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

The great martingale

The great martingale is similar to the classic martingale, except that in addition to doubling the position size after each loss, we add one additional unit.

This is more risky than the classic martingale in case of successive losses but it allows significantly increasing gains otherwise.

This code must be integrated with your own entry and exit conditions

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE ExitIndex = -2
// Initial position size of 1.
//*********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex

//***********Code to insert at the end of the trading system**********//
IF BarIndex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      OrderSize = OrderSize * 2 + 1 // if the last trade was losing, double OrderSize and
      add 1.
   ELSIF PositionPerf(1) >= 0 THEN
      OrderSize = 1 // if the last trade was winning, set the OrderSize to 1.
   ENDIF
ENDIF
//*********************//
// The position size must be determined depending on the variable OrderSize in the entire
code.
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

The Piquemouche

The Piquemouche is another variant of the classic martingale. In case of loss, we increase the position size by 1 if there are less than 3 consecutive losses. If there are more than 3 consecutive losses, we double the position. A gain resets the position size to 1 unit.

This trading system is less risky than the 2 previous ones because the position size is not exponentially increased until 3 successive losses are attained.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE BadTrades = 0
ONCE ExitIndex = -2

// Initial position size of 1.
//*********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex

//***********Code to insert at the end of the trading system**********//
IF BarIndex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      BadTrades = BadTrades + 1
      IF BadTrades < 3 THEN
         // If the last trade was losing and there are less than 3 successive losses,
         // add one to OrderSize.
         OrderSize = OrderSize + 1
      ELSIF BadTrades MOD 3 = 0 THEN
         // If the last position was losing and there are more than 3 consecutive losses,
         // double OrderSize.
         OrderSize = OrderSize * 2
      ENDIF
   ELSIF PositionPerf(1) >= 0 THEN
      // If the previous trade was winning, reset OrderSize to 1.
      OrderSize = 1
      BadTrades = 0
   ENDIF
ENDIF
//*********************//
// The position size must be determined depending on the variable OrderSize in the entire
code.
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

The d'Alembert pyramid

This martingale was made famous by d'Alembert, a French 18th century mathematician. In case of loss, the position size is increased by 1 unit, in case of gain it is decreased by 1 unit.

This technique of position size management is relevant only if we suppose that successive gains reduce the probability of winning again and successive losses reduce the probability of losing again.

This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE ExitIndex = -2
// Initial position size of 1.
//*********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex

//***********Code to insert at the end of the trading system**********//
IF BarIndex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      OrderSize = OrderSize + 1
   ELSIF PositionPerf(1) >= 0 THEN
      OrderSize = MAX(OrderSize -1, 1)
   ENDIF
ENDIF
//*********************//
// The position size must be determined depending on the variable OrderSize in the entire code.
```

Warning: The example codes shown in this manual are for learning purposes only. You are free to determine all criteria for your own trading. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment. All of the information in this manual is "General" information and is not in any case personal or financial advice nor a solicitation to buy or sell any financial instrument. Past performance is not indicative of future results. Any trading system may expose you to a risk of loss greater than your initial investment.

The contre d'Alembert

This is a reciprocal trading system of the D'Alembert Pyramid. We decrease the position size in case of a loss and increase the position size in case of a gain.

This technique is relevant if we believe that a past loss increases the probability of a future loss and a past gain increases the probability of a future gain.

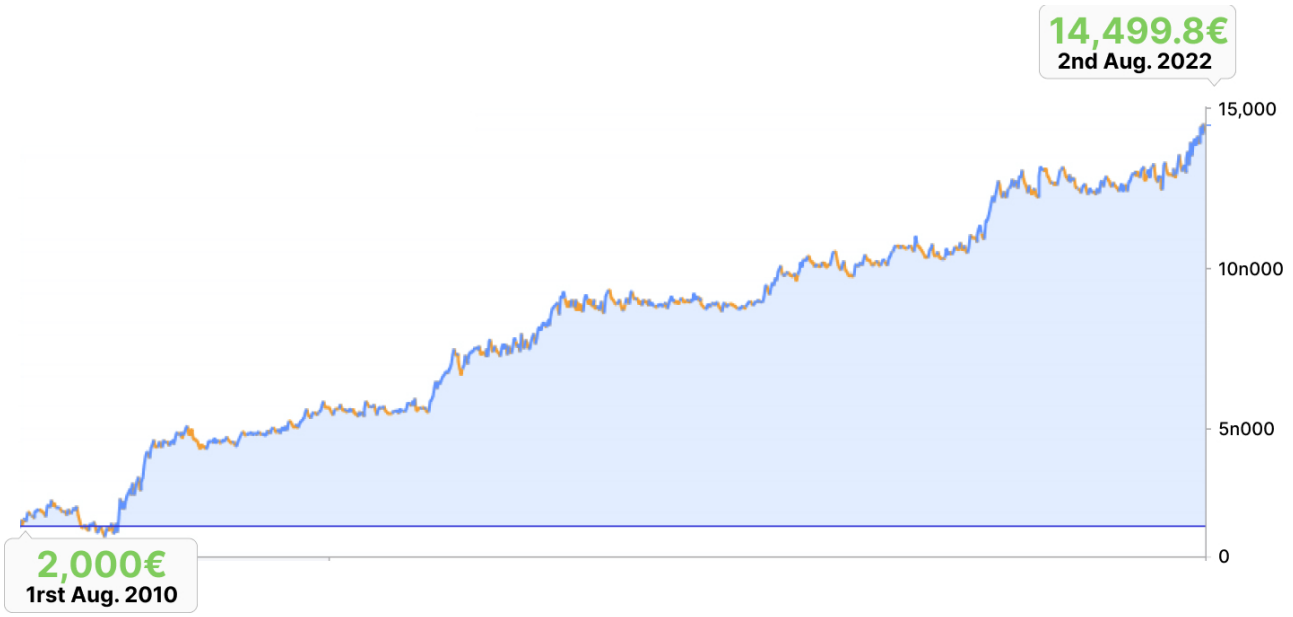
This code must be integrated with your own entry and exit conditions.

```
//***********Code to insert at the beginning of the trading system**********//
ONCE OrderSize = 1
ONCE ExitIndex = -2
// Initial position size of 1.
//*********************//
//***********Code to insert just after closing a position**********//
ExitIndex = BarIndex

//***********Code to insert at the end of the trading system**********//
IF BarIndex = ExitIndex + 1 THEN
   ExitIndex = 0
   IF PositionPerf(1) < 0 THEN
      OrderSize = MAX(OrderSize -1, 1)
   ELSIF PositionPerf(1) >= 0 THEN
      OrderSize = OrderSize + 1
   ENDIF
ENDIF

//*********************//
// The position size must be determined depending on the variable OrderSize in the entire
code.
```

Annex C: Example of a trading system

Warning: The example trading system described in this section is for information purposes only. Its purpose is to present the features of the ProOrder service. This presentation is only for learning purposes and is not a recommendation or suggestion by ProRealTime to use the strategy described in any manner. The reader should note that the numbers referenced in this example relate to past data and that past performance (as referenced in this document) is not a reliable indicator of future results.

This annex presents a detailed example of a "Breakout" trading system based on the 15-minute time frame and applied on the mini France 40 CFD contract (1€ per point) and analyzes its performance analysis over the past years[1] with the ProBacktest simulation*.



Raw performance[1] of the "Breakout ProOrder" automatic trading system simulated with ProBacktest, over 12 years.

Gross annualized performance[1] **16.46%**

Starting capital: **2,000.00 €** (August 2 2010)

End capital: **14,499.80 €** (August 2 2022)

Gross profit[1] over 12 years: **+12,499 .8 (+624.9 %)**

	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Breakout ProOrder Automatic trading system	+27%[2]	**+76%**	+7%	+12%	+33%	+21%	-0.5%	+0.9%	+9%	+6%	+21%	+3%	+10%[2]
CAC 40 Index[1,3]	-3.3%[2]	-17%	+15.2%	+18%	-0.5%	+8.5%	+4.9%	+9.2%	-10.9%	+26.3%	-7.4%	+29%	-14%

[1] The performances and gross gains are calculated excluding brokerage fees (commissions, royalties or other charges). These fees would reduce the performance of the system.

[2] 2010 and 2022: partial years.

[3] CAC 40 data provided by Euronext Paris.

Introduction to the "ProOrder Breakout" automatic trading system

A classic "Breakout" system checks the highest and lowest price levels at the end of a defined period (in our example, the first 30 minutes of trading after 9:00 a.m.), then places a buy order on the upper level and a sell order on the lower level.

The "ProOrder Breakout" system presented below is a modified version of the classic breakout strategy.

This example "ProOrder Breakout" system takes at most 2 positions per day (sometimes only one position and sometimes none at all) between 9:30 a.m. and 9:45 p.m. In any case, the system is no longer in position after 9:45 p.m. and it is possible to know the gain or loss of the day at that time.

To learn more:

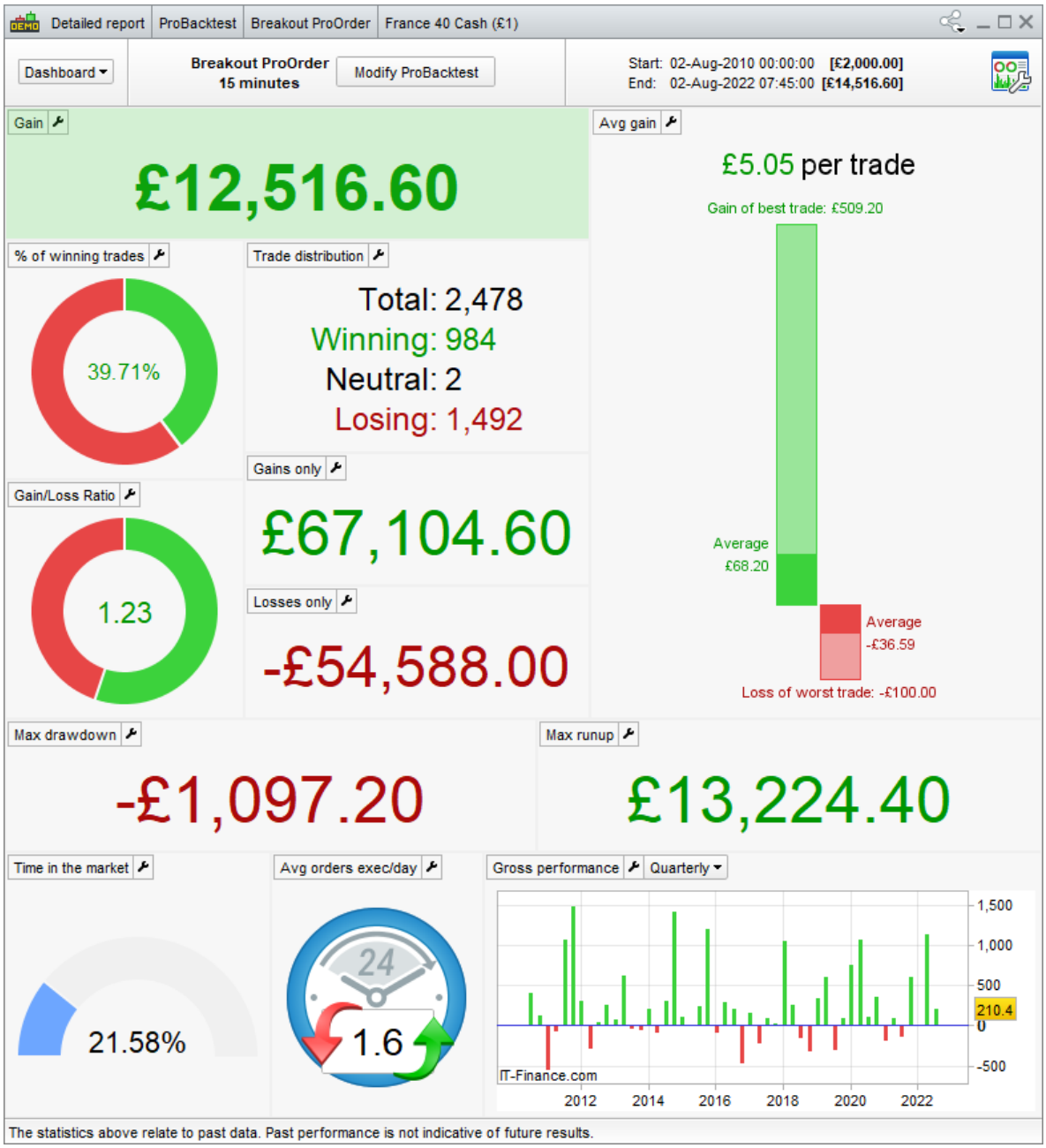
- **Results of the trading system (data from 2010 to 2022)**
- **Description of the ideas of the trading system**

- **Code of the trading system**
- **How to test a code/trading system**

Results of the trading system (from 2010 to 2022)

Warning concerning the results: The numbers presented relate to past data. Past performance is not a reliable indicator of future results and is not constant over time.

The image below shows the results of the system applied to the instrument CFD France 40 on past data, from August 2nd, 2010 to August 2nd, 2022 , which is about 3084 trading days simulated with our ProBacktest module.



As shown above, out of 2,478 positions taken, the average gain in case of a winning position is 68.20 £ over the period (biggest gain of 509.20 £) while the average loss in case of a losing position is 36.59 £ over the same period (biggest loss of 100 £). In our example, the number of winning positions (990) is lower than the number of losing positions (1,502) and the total amount of gains is higher than the total amount of losses.

Estimating the spread fee on the France 40 CFD contract at €1.25 on average per order, the total fees would be 6235 £ over the entire period, which would result in:

- an annualized performance of 8.21% net of fees with an initial capital of 2,000£.

Notes:

1) In this example, we chose an initial capital of 2,000 £, which is almost twice the maximum consecutive loss over the entire period since the beginning of the simulation in case the strategy were started at the most unfavorable moment (maximum consecutive historical loss of 1,097.2 £). A simulation with a lower initial amount (for example 800 £) would have been possible since the margin for the mini France 40 CFD (1£) is 305 £. If the strategy initially loses about 400 £ (as in the case of the simulation) then regularly gains additional capital.

2) In the example, the position size does not change, even though theoretically it would be possible to increase the position size every 2,000 £ of capital gained by adding 2 additional contracts to each new position and exponentially increasing both the potential gain and the potential risk. In fact, beginning with 2,000 £ of capital and position size of 2 contracts (even if the margin required by the broker is only 42 £ of capital), has the same risk in terms of percent loss as starting with 4,000 £ of capital and positions of 4 contracts for example.

3) A CFD is a contract for the difference between the price of an asset at the time an investor opens a position and when he closes it. CFDs are leveraged products. That means an investor only deposits part of the total of their exposure to the market. CFDs can significantly increase the return of an investment, but the losses can also be greater than deposits. CFDs are intended for experienced clients who are able to understand the risks involved and who have sufficient financial means to bear such risks.

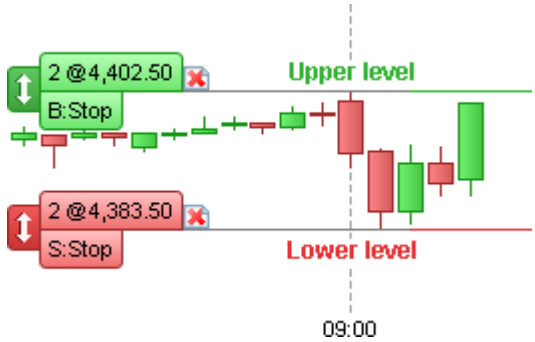
Description of the ideas of the trading system

Initial Idea: 30-minute breakout

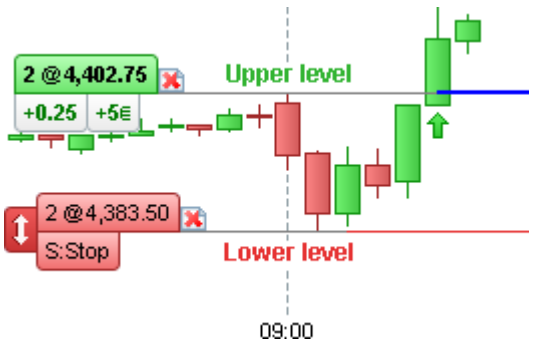
The classic 30-minute breakout strategy identifies the highest and lowest price of the mini France 40 CFD contract during the first 30 minutes of significant trading of the day (between 9:00 a.m. and 9:30 a.m. This period is represented by two 15-minute candlesticks in each of the images below and defines these levels as the upper level and lower level.

Then, a buy stop order is placed on the upper level and a sell stop order is placed on the lower level, as shown in the image on the left below:

- Once the orders have been placed, if the upper level is touched first by the price, a buying position will be opened as shown in the image on the right.
- If the lower level is touched first by the price, a selling position will be opened.



Display of the buy stop order in green and the sell stop order in red, once the high and lower levels have been defined.



Display of a buying position in an example where the price first touched the upper level, which opened the buying position.

Note:

The trading system does not use a profit target order to close a position. However, any position still open at 9:45 p.m. is closed to avoid the risk of holding a position overnight.

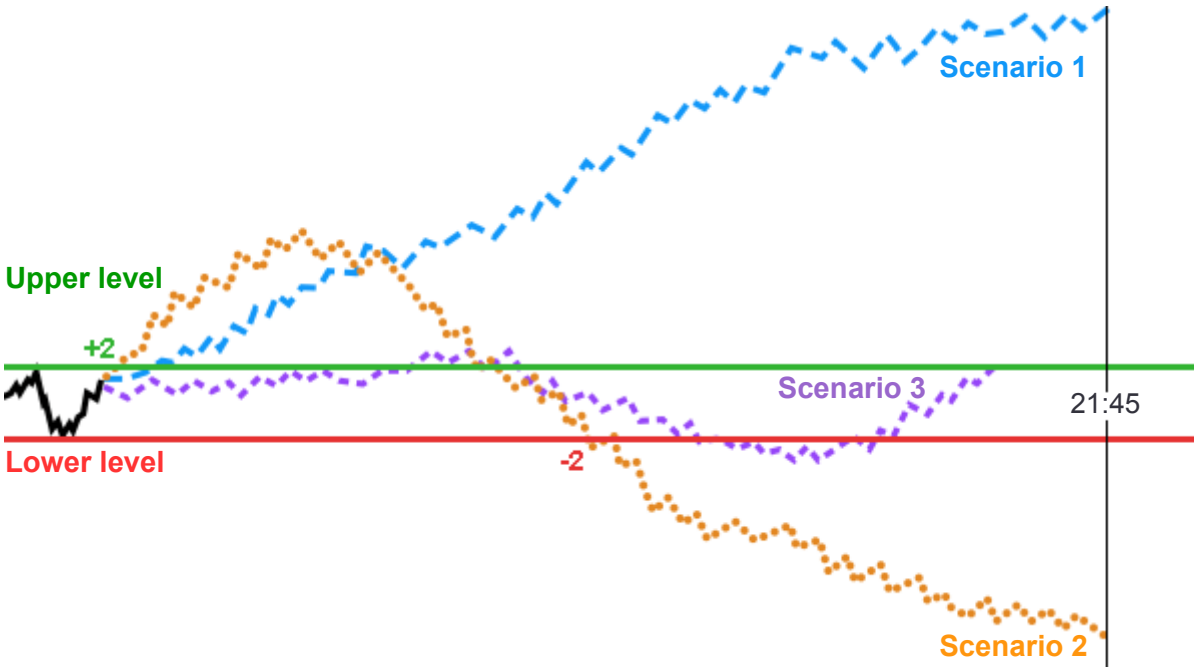
2nd idea: two positions will be taken per day at the most

In our example, we take at most two positions per day (one buying position and one selling position). When one of the two levels is touched first, a position is opened and the opposing level becomes a protection stop which inverts the position if it is touched after that.

Let's look at 3 scenarios which could happen if a buying position is opened first:

Scenario 1:

1. A buying position of **+2** is opened.
2. The price remains above the upper level all day.
3. The position is closed at 9:45 p.m. with a gain.



Scenario 2:

1. A buying position of **+2** is opened.
2. The price decreases to the lower level.
3. A protection stop closes the position with a loss and opens a new selling position of **-2**. The initial position is reversed as a result.
4. The price continues to decrease and the position is closed at 9:45 p.m. with a gain.

Scenario 3:

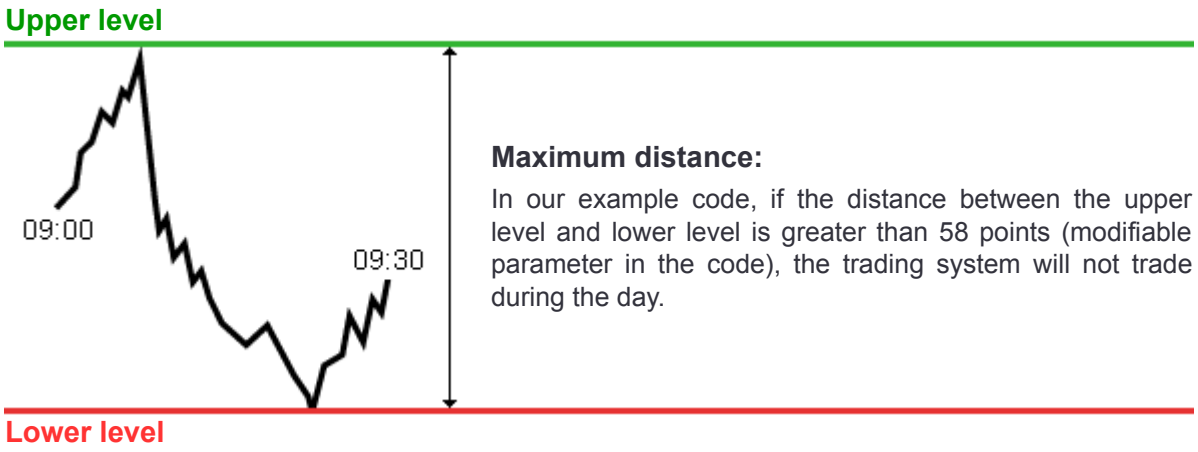
1. A buying position of **+2** is opened.
2. Price decreases to the lower level.
3. A protection stop closes the position with a loss and opens a new selling position of **-2**. The initial position is reversed as a result.
4. Price increases again to the upper level. The second selling position is closed by a protection stop with a loss (without reversing the position this time).

All of the scenarios described above are also possible in the opposite direction when a selling position is opened first.

3rd idea: limit risk by defining a maximum and minimum distance between the two levels

In our example code, if there is a distance of more than 58 points between the upper level and lower level, the system will not take a position during the day.

This condition was introduced to attempt to directly limit the risk, because as we saw in the previous examples, the trading system can theoretically lose twice the distance between the upper and lower level.



Maximum distance:

In our example code, if the distance between the upper level and lower level is greater than 58 points (modifiable parameter in the code), the trading system will not trade during the day.

The maximum distance is a modifiable parameter in the code that can be adjusted depending on the amount of risk acceptable to each investor.

Notes:

1) The theoretical maximum loss indicated above is based on execution of orders at the stop prices calculated by the system. In some cases, real execution price may be different from the price of a requested stop order.

2) If the upper and lower levels between 9:00 a.m. and 9:30 a.m. were crossed several times per day between 9:30 a.m. and 9:45 p.m. (example of scenario 3 which is the most unfavorable), every day starting from the beginning date of the trading system, the results would be negative day after day during this period and all of the initial capital would eventually be lost.

3) The trading system could place a 3rd position in the same day if the buy order level and sell order level were both crossed during the 15-minute period after defining these two levels.

4) Our ProOrder module lets you easily simulate a trading system with several different values for the maximum distance. This variable optimization can show that the performance of the trading system could be even better with a larger maximum distance, but that we would have higher risk of loss per trade in this case because the protection stop orders would be farther from the orders to enter positions.

4th idea: increase the chance of favorable execution

In our example, we start with the theory that the upper and lower levels are important zones of support and resistance on which many investors may have placed orders. When price breaks out of such levels, this can cause an acceleration of price in the direction of the breakout with varying speeds in a few seconds.

To take advantage of this amplification of a trend just after a breakout, the system has been configured to place stop orders 4 points underneath the upper level and 4 points above the lower level as shown below. This 4 parameter called "OrderDistance" in the code of the system can be modified depending on the preferences of each investor.

Please note:

1) This condition reduces the distance between the buy and sell order by 2 x 4 points and as a result also decreases the maximum risk, because in our system, 50 points at most separate the two orders. The new maximum theoretical daily loss of the system becomes 200 € (2 X amplitude X 50 € X 2 positions).

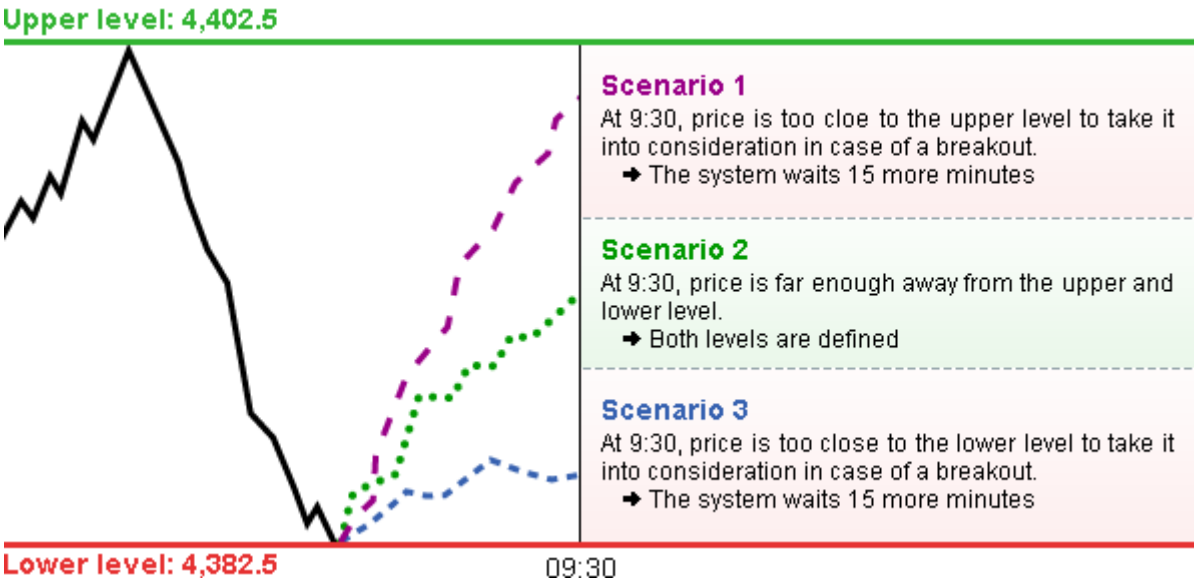
2) If many other investors were to place orders at the same level and in the same direction as the strategy, it would be possible to increase "OrderDistance" (to 4.5, or 5, or 5.5 or more) to take advantage of acceleration after the execution of your own order.

5th idea: only trade on clear breakouts to avoid false signals

Case 1: price too close to the upper or lower level at 9:30 a.m.

In this example, we decided that the system would not place an order if the price of the financial instrument of reference is too close to the upper or lower level at the end of the second 15-minute period.

In this case, the system waits until the end of an additional 15-minute period.



If at the end of the new 15-minute period (9:45 a.m.), the price is still too close to the levels, the system waits 15 more minutes and so on until the levels can be defined. In any case, for a given day, if the maximum distance of 58 points is passed, the system will not trade on that day.

This parameter measuring distance from the upper and lower level is is called "MinPercent" in the example code of the system and in this case is set to 30%, but can be modified depending on the choice of each investor.

In the image on the left, price closes too close to the lower level until the 10:45 a.m. candlestick.

The lower level was lowered each time as a result until the close of the 10:45 a.m. - 11:00 a.m. candle.

However, at that time, the distance between the upper and lower level was more than 58 points and in our example, the system will not take a position that day.

Case 2: distance between upper and lower level too small

In our example, the system also avoids situations in which the distance between the upper level and the lower level is too small (any breakout could be considered not significant in this case). To look at this, the system measures the distance between the buy order and the sell order. If this distance is less than 11 points, (parameter called "AmplitudeMin" and modifiable in the code), the system will not open any positions during the day.

6th idea: Do not take a position if there is not enough time left in the trading day

Because in our example code, we can not have an open position after 9:45 p.m., we have configured the strategy to not open any new position after 5:15 p.m.

Also, if a trading day is shortened due to a holiday (ex: Christmas or December 31st), the system will not open a position on that day.

These parameters are also customizable in the code of the strategy to define the time after which signals are ignored by the system.

The example trading system described in this section is for information purposes only. Its purpose is to present the features of the ProOrder service. This presentation is only for learning purposes and is not a recommendation or suggestion by ProRealTime to use the strategy described in any manner. The reader should note that the numbers referenced in this example relate to past data and that past performance (as referenced in this document) is not a reliable indicator of future results.

You can get free programming help **via our dedicated form**.

Code of the "ProOrder Breakout" trading system

```
// We don't load data before the start of the system.
// As a result, if the system is started in the afternoon,
// it will wait until the next day before placing any orders.
DEFPARAM PreLoadBars = 0
// The position is closed at 9:45 p.m., local market time (France).
DEFPARAM FlatAfter   = 214500
// No new position is taken after the candlestick that closes 5:15 p.m.
LimitEntryTime       = 171500
// The market analysis starts at the 15-minute candlestick which closes at 9:30 a.m.
StartTime            = 091500
// Some holidays such as the 24th and 31st of December are excluded
IF (Month = 5 AND Day = 1) OR (Month = 12 AND (Day = 24 OR Day = 25 OR Day = 26 OR Day = 30 OR Day = 31)) THEN
   TradingDay = 0
ELSE
   TradingDay = 1
ENDIF

// Variables which can be adapted based on your preferences
PositionSize  = 2
AmplitudeMax  = 58
AmplitudeMin  = 11
OrderDistance = 4
MinPercent    = 30
```

```
// We initialize this variable once at the beginning of the trading system.
ONCE StartTradingDay = -1
// The variables which can change during the day are initialized
// at the beginning of each new trading day.
IF (Time <= StartTime AND StartTradingDay <> 0) OR IntradayBarIndex = 0  THEN
   BuyLevel          = 0
   SellLevel         = 0
   BuyPosition       = 0
   SellPosition      = 0
   StartTradingDay   = 0
ELSIF Time >= StartTime AND StartTradingDay = 0 AND TradingDay = 1 THEN
   // We store the index of the first bar of the trading day
   IndexStartDay   = IntradayBarIndex
   StartTradingDay = 1

ELSIF StartTradingDay = 1 AND Time <= LimitEntryTime THEN
   // For each trading day, the highest and lowest price of the instrument
   // are recorded every 15 minutes since StartTime
   // until the buy and sell levels can be defined
   IF BuyLevel = 0 OR SellLevel = 0 THEN
      UpperLevel = Highest[IntradayBarIndex - IndexStartDay + 1](High)
      LowerLevel = Lowest [IntradayBarIndex - IndexStartDay + 1](Low)
      // Calculation of the difference between the highest
      // and lowest price of the instrument since StartTime
      DayDistance = UpperLevel - LowerLevel
      // Calculation of the minimum distance between the upper level and lower level
      // to consider a breakout of the upper or lower level to be significant
      MinDistance  = DayDistance * MinPercent / 100
      // Calculation of the buy and sell levels for the day if the conditions are met
      IF DayDistance <= AmplitudeMax THEN
         IF SellLevel = 0 AND (Close - LowerLevel) >= MinDistance THEN
            SellLevel = LowerLevel + OrderDistance
         ENDIF
         IF BuyLevel = 0 AND (UpperLevel - Close) >= MinDistance THEN
            BuyLevel = UpperLevel - OrderDistance
         ENDIF
      ENDIF
   ENDIF
```

```
   // Creation of buy and sell short orders for the day if the conditions are met
   IF SellLevel > 0 AND BuyLevel > 0 AND (BuyLevel - SellLevel) >= AmplitudeMin THEN
      IF BuyPosition = 0 THEN
         IF LongOnMarket THEN
            BuyPosition = 1
         ELSE
            BUY PositionSize CONTRACT AT BuyLevel STOP
         ENDIF
      ENDIF
      IF SellPosition = 0 THEN
         IF ShortOnMarket THEN
            SellPosition = 1
         ELSE
            SELLSHORT PositionSize CONTRACT AT SellLevel STOP
         ENDIF
      ENDIF
   ENDIF
ENDIF
// Definition of the conditions to exit the market when a
// buying or selling position is open
IF LongOnMarket AND ((Time <= LimitEntryTime AND SellPosition = 1) OR Time >
LimitEntryTime) THEN
   SELL AT SellLevel STOP
ELSIF ShortOnMarket AND ((Time <= LimitEntryTime AND BuyPosition = 1) OR Time >
LimitEntryTime) THEN
   EXITSHORT AT BuyLevel STOP
ENDIF
// Definition of the maximum amount to risk per position in case
// price goes in an unfavorable direction
SET STOP PLOSS AmplitudeMax
```

How to test a trading system / code

With a virtual portfolio (PaperTrading mode)

If you have an account on **ProRealTime**, you can execute trading systems on a virtual PaperTrading portfolio.

PaperTrading mode lets you try your trading system day after day in real market conditions, without risking real money.

It will let you see positions opened in real time and also test your own reactions to automatic trading. Note that you can reset the value of your PaperTrading portfolio as many times as you want in order to start a new simulation.

Real trading mode

ProOrder AutoTrading is also available in real trading mode with **ProRealTime via Interactive Brokers** or an **IG account sponsored by ProRealTime**.

Warning: If you use a trading system via the ProOrder service, in real trading mode, this service will send automatic signals, according to the parameters you have set, in order to execute orders without validation of each individual order from you being required. Your system will be executed automatically, even if your computer is off. It is your responsibility to ensure that your trading system does not lead to losses above an amount acceptable to you.

In any case, ProRealTime will not be liable for any losses incurred following your execution of a trading system.

We remind you that due to leverage, CFD trading may expose you to risk of loss greater than your deposits. These financial instruments are only suitable for experienced clients who can understand the risks and have sufficient financial means to bear such risks.

Glossary

A

CODE	SYNTAX	FUNCTION
ABS	ABS(a)	Mathematical function "Absolute Value" of a.
AccumDistr	AccumDistr(price)	Classical Accumulation/Distribution indicator.
ACOS	ACOS(a)	Mathematical function "Arc cosine (returns an angle in degrees).
AdaptiveAverage	AdaptiveAverage[x,y,z](price)	Adaptive Average Indicator.
ADX	ADX[N]	Indicator Average Directional Index or "ADX" of n periods.
ADXR	ADXR[N]	Indicator Average Directional Index Rate or "ADXR" of n periods.
AND	a AND b	Logical AND Operator.
ArrayMax	ArrayMax($MyArray)	Returns the highest value of the array.
ArrayMin	ArrayMin($MyArray)	Returns the lowest value of the array.
ArraySort	ArraySort($MyArray, ASCEND)	Sort the table in ascending (ASCEND) or descending (DESCEND) order.
AroonDown	AroonDown[P]	Aroon Down indicator.
AroonUp	AroonUp[P]	Aroon Up indicator.
ATAN	ATAN(a)	Mathematical function "Arc tangent" (returns an angle in degrees).
AS	GRAPH x AS "ResultName"	Instruction used to name a line or indicator displayed on chart. Used with "GRAPH".
ASIN	ASIN(a)	Mathematical function "Arc sine" (returns an angle in degrees).
Average	Average[N](price)	Simple Moving Average of n periods.
AverageTrueRange	AverageTrueRange[N](price)	"Average True Range" - True Range smoothed with the Wilder method.

B

CODE	SYNTAX	FUNCTION
BarsSince	BarsSince(condition,n)	Return the number of candle since the nth occurrence of the specified condition (n=0 means last occurrence and is the default, n=1 means second last occurrence).
BarIndex	BarIndex	Number of bars since the beginning of data loaded (in a chart in the case of a ProBuilder indicator or for a trading system in the case of ProBacktest or ProOrder).
BollingerBandWidth	BollingerBandWidth[N](price)	Bollinger Bandwidth indicator.
BollingerDown	BollingerDown[N](price)	Lower Bollinger band.
BollingerUp	BollingerUp[N](price)	Upper Bollinger band.
BREAK	(FOR...DO...BREAK...NEXT) or (WHILE...DO...BREAK...WEND)	Instruction forcing the exit of FOR or WHILE loop.
BUY	BUY (n) SHARES	Long position opening instruction.
BREAKEVEN	SET STOP BREAKEVEN	Allows you to set a protective stop at the entry price of the current position.

C

CODE	SYNTAX	FUNCTION
CALL	myResult = CALL myFunction	Calls a user indicator to be used in the program you are coding.
CASH	BUY x CASH	Designates the amount to be used.
CCI	CCI[N](price)	Commodity Channel Index indicator.
CEIL	CEIL(N,M)	Returns the smallest number greater than N applied to the decimal m.
ChaikinOsc	ChaikinOsc[Ch1, Ch2](price)	Chaikin oscillator.

CODE	SYNTAX	FUNCTION
Chandle	Chandle[N](price)	Chande Momentum Oscillator.
ChandeKrollStopUp	ChandeKrollStopUp[Pp, Qq, X]	Chande and Kroll Protection Stop on long positions.
ChandeKrollStopDown	ChandeKrollStopDown[Pp, Qq, X]	Chande and Kroll Protection Stop on short positions.
Close	Close[N]	Closing price of the current bar or of the n-th last bar.
COLOURED	GRAPH x COLOURED(R,G,B)	Colors a curve with the color you defined using the RGB convention.
COS	COS(a)	Cosine function ('a' argument in degrees).
COUNTOFLONGSHARES	COUNTOFLONGSHARES	Counts the number of securities in long position.
COUNTOFPOSITION	COUNTOFPOSITION	Counts the number of securities in position (either long or short).
COUNTOFSHORTSHARES	COUNTOFSHORTSHARES	Counts the number of securities in short position.
CONTRACT	BUY 1 CONTRACT	Refers to the number of contracts to be purchased. Equivalent of "Shares".
CROSSES OVER	a CROSSES OVER b	Boolean Operator checking whether a curve has crossed over another one.
CROSSES UNDER	a CROSSES UNDER b	Boolean Operator checking whether a curve has crossed under another one.
cumsum	cumsum(price)	Sums a certain price on the whole data loaded.
CurrentDayOfWeek	CurrentDayOfWeek	Represents the current day of the week.
CurrentHour	CurrentHour	Represents the current hour.
CurrentMinute	CurrentMinute	Represents the current minute.
CurrentMonth	CurrentMonth	Represents the current month.
CurrentSecond	CurrentSecond	Represents the current second.
CurrentTime	CurrentTime	Represents the current time (HHMMSS).
CurrentYear	CurrentYear	Represents the current year.
CustomClose	CustomClose[N]	Constant which is customizable in the settings window of the chart (default: Close).
Cycle	Cycle(price)	Cycle Indicator.

D

CODE	SYNTAX	FUNCTION
Date	Date[N]	Reports the date of each bar loaded on the chart.
Day	Day[N]	Day number at the end of the current candle.
Days	Days[N]	Counter of days since 1900.
Days	TIMEFRAME(X Days)	Set the period to "X Days" for further calculations of the code.
DayOfWeek	DayOfWeek[N]	Day of the week of each bar.
DClose	DClose(N)	Close of the n-th day before the current one.
DEFPARAM	DEFPARAM	Allows you to define parameters.
DEMA	DEMA[N](price)	Double Exponential Moving Average.
DHigh	DHigh(N)	High of the n-th day before the current bar.
DI	DI[N](price)	Refers to the Demand Index.
DIminus	DIminus[N](price)	Represents the DI- indicator.
DIplus	DIplus[N](price)	Represents the DI+ indicator.
DivergenceCCI	DivergenceCCI[Div1,Div2,Div3,Div4]	Indicator for detecting discrepancies between price and CCI.
DivergenceMACD	DivergenceMACD[Div1,Div2,Div3,Div4](close)	Indicator for detecting divergences between the price and the MACD.
DLow	DLow(N)	Low of the n-th day before the current one.
DO	See FOR and WHILE	Optional instruction in FOR loop and WHILE loop to define the loop action.
DonchianChannelCenter	DonchianChannelCenter[N]	Middle channel of the Donchian indicator for N periods.
DonchianChannelDown	DonchianChannelDown[N]	Lower channel of the Donchian indicator for N periods.

CODE	SYNTAX	FUNCTION
DonchianChannelUP	DonchianChannelUp[N]	Upper channel of the Donchian indicator for N periods.
DOpen	DOpen(N)	Open of the n-th day before the current one.
DOWNTO	See FOR	Instruction used in FOR loop to process the loop with a descending order.
DPO	DPO[N](price)	Detrended Price Oscillator.
DynamicZoneRSIDown	DynamicZoneRSIDown[rsiN, N]	Lower band of the Dynamic Zone RSI indicator.
DynamicZoneRSIUp	DynamicZoneRSIUp[rsiN, N]	Upper band of the Dynamic Zone RSI indicator.
DynamicZoneStochasticDown	DynamicZoneStochasticDown[N]	Lower band of the Dynamic Zone Stochastic indicator.
DynamicZoneStochasticUp	DynamicZoneStochasticUp[N]	Upper band of the Dynamic Zone Stochastic indicator.

E

CODE	SYNTAX	FUNCTION
EaseOfMovement	EaseOfMovement[I]	Ease of Movement indicator.
ElderrayBearPower	ElderrayBearPower[N](close)	Elder Ray Bear Power indicator.
ElderrayBullPower	ElderrayBullPower[N](close)	Elder Ray Bull Power indicator.
ELSE	See IF/THEN/ELSE/ENDIF	Instruction used to call the second condition of If-conditional statements.
ELSIF	See IF/THEN/ELSE/ENDIF	Stands for Else If (to be used inside of conditional loop).
EMV	EMV[N]	Ease of Movement Value indicator.
ENDIF	See IF/THEN/ELSE/ENDIF	Ending Instruction of IF-conditional statement.
EndPointAverage	EndPointAverage[N](price)	End Point Moving Average.
EXITSHORT	EXITSHORT x SHARES	To close a short position.
EXP	EXP(a)	Mathematical Function "Exponential".
ExponentialAverage	ExponentialAverage[N](price)	Exponential Moving Average.

F – G – H

CODE	SYNTAX	FUNCTION
FractalDimensionIndex	FractalDimensionIndex[N](close)	Fractal Dimension Index indicator.
FOR/TO/NEXT	FOR i =a TO b DO a NEXT	FOR loop (processes all the values with an ascending (TO) or a descending order (DOWNTO)).
FLATAFTER	DefParam FlatAfter = HHMMSS	Cancels any pending order, closes any open position and prevents additional orders from being added after the time set (in hours, minutes and seconds) in the user time zone.
FLATBEFORE	Defparam FlatBefore = HHMMSS	Cancels any pending order, closes any open position and prevents additional orders from being added before the time set (in hours, minutes and seconds) in the user time zone.
ForceIndex	ForceIndex(price)	Force Index indicator determines who controls the market (buyer or seller).
FLOOR	FLOOR(N,M)	Returns the largest number less than N with a precision of M digits after the decimal point.
GetTimeFrame	GetTimeFrame	Returns the current period in seconds.
GRAPH	GRAPH myvariable AS "myvariable"	Backtest instruction that allows you to visualize the values of the variables on the historical data.
GRAPHONPRICE	GRAPHONPRICE myvariable AS "myvariable"	Backtest instruction that allows you to visualize the values of the variables on the price graph.
High	High[N]	High of the current bar or of the n-th last bar.
Highest	highest[N](price)	Highest price over a number of bars to be defined.
HistoricVolatility	HistoricVolatility[N](price)	Historic Volatility (or statistic volatility).
Hour	Hour[N]	Represents the hour of each bar loaded in the chart.
Hours	TIMEFRAME(X Hours)	Defines the "X hours" period for further calculations in the code.
HullAverage	HullAverage[N](close)	Designates the Hull Average indicator.

I - J - K

CODE	SYNTAX	FUNCTION
IF/THEN/ENDIF	IF a THEN b ENDIF	Group of conditional instructions without second instruction.
IF/THEN/ELSE/ENDIF	IF a THEN b ELSE c ENDIF	Group of conditional instructions.
IntradayBarIndex	IntradayBarIndex[N]	Counts how many bars are displayed in one day on the whole data loaded.
IsSet	IsSet($MyArray[index])	Returns 1 or 0 if the value at the index of the array is defined or not.
KeltnerBandCenter	KeltnerBandCenter[N]	Central band of the Keltner indicator of N periods.
KeltnerBandDown	KeltnerBandDown[N]	Lower band of the Keltner indicator of N periods.
KeltnerBandUp	KeltnerBandUp[N]	Upper band of the Keltner indicator of N periods.
KijunSen	KijunSen[T,K,S]	Returns the KijunSen value of the Ichimoku indicator.

L

CODE	SYNTAX	FUNCTION
LastSet	LastSet($MyArray)	Returns the highest defined index from the given Array.
LIMIT	BUY AT x LIMIT	Instruction that introduces a Limit order.
LinearRegression	LinearRegression[N](price)	Linear Regression indicator.
LinearRegressionSlope	LinearRegressionSlope[N](price)	Slope of the Linear Regression indicator.
LOG	LOG(a)	Mathematical Function "Neperian logarithm" of a.
LONGONMARKET	LONGONMARKET	Indicates whether you have a long position in the market.
LONGTRIGGERED	LONGTRIGGERED[N]	Indicates whether a buy order has been executed on the Nth previous candle.
Low	Low[N]	Low of the current bar or of the n-th last bar.
lowest	lowest[N](price)	Lowest price over a number of bars to be defined.
LOSS	SET STOP LOSS x	Allows you to place a stop loss at x units from the entry price.
%LOSS	SET STOP %LOSS x	Defines a loss at x% from the position entry price.
$LOSS	SET STOP $LOSS x	Defines a loss of x €,$ (currency of the instrument).
PLOSS	SET STOP PLOSS x	Defines a loss at x points from the position entry price.
LOT	BUY 1 LOT	Designates the number of lots to be purchased (equivalent to "SHARE").

M – N

CODE	SYNTAX	FUNCTION
MACD	MACD[S,L,Si](price)	Moving Average Convergence Divergence (MACD) histogram.
MACDline	MACDLine[S,L,Si](price)	MACD line indicator.
MACDSignal	MACDSignal[S,L,Si](price)	MACD Signal line indicator.
MARKET	BUY AT MARKET	Designates at-market order. It will be executed at the opening of the following bar.
MassIndex	MassIndex[N]	Mass Index Indicator applied over N bars.
MAX	MAX(a,b)	Mathematical Function "Maximum".
MedianPrice	MedianPrice	Average of the high and the low.
MIN	MIN(a,b)	Mathematical Function "Minimum".
Minute	Minute	Represents the minute of each bar loaded in the chart.
Minutes	TIMEFRAME(X Minutes)	Sets the period to "X Minutes" for the following code calculations.
MOD	a MOD b	Mathematical Function "remainder of the division".
Momentum	Momentum[I]	Momentum indicator (close – close of the n-th last bar).
MoneyFlow	MoneyFlow[N](price)	MoneyFlow indicator (result between -1 and 1).

CODE	SYNTAX	FUNCTION
MoneyFlowIndex	MoneyFlowIndex[N]	MoneyFlow Index indicator.
Month	Month[N]	Represents the month of each bar loaded in the chart.
Months	TIMEFRAME(X Months)	Sets the period to "X Months" for the following code calculations.
NegativeVolumeIndex	NegativeVolumeIndex[N]	Negative Volume Index indicator.
NEXT	See FOR/TO/NEXT	Ending Instruction of FOR loop.
NOCASHUPDATE	DEFPARAM NOCASHUPDATE=true/false	Allows backtests to not update their initial capital with gains and losses (instruction for backtests only).
NOT	Not A	Logical Operator NOT.

O

CODE	SYNTAX	FUNCTION
OBV	OBV(price)	On-Balance-Volume indicator.
ONCE	ONCE Variable = Value	Introduces a definition statement which will be processed only once.
ONMARKET	ONMARKET	Indicates if you are in a position.
Open	Open[N]	Open of the current candle or of the n-th previous candle.
OpenDay	OpenDay[N]	Opening day of the current candle or the nth previous candle.
OpenDayOfWeek	OpenDay[N]	Day of the week of the opening of the current candle or the nth previous candle.
OpenHour	OpenHour[N]	Opening time of the current candle or the nth previous candle.
OpenMinute	OpenMInute[N]	Opening minute of the current candle or the nth previous candle.
OpenMonth	OpenMonth[N]	Opening month of the current candle or the nth previous candle.
OpenSecond	OpenSecond[N]	Opening second of the current candle or the nth previous candle.
OpenTime	OpenTime[N]	Time (HHMMSS) of the opening of the current candle or the nth previous candle.
OpenTimestamp	OpenTimestamp[N]	UNIX opening timestamp of the current candle or the nth previous candle.
OpenWeek	OpenWeek[N]	Opening week of the current candle or the nth previous candle.
OpenYear	OpenYear[N]	Opening year of the current candle or the nth previous candle.
OR	a OR b	Logical OR Operator.

P - Q

CODE	SYNTAX	FUNCTION
PERPOINT	PERPOINT	Designates the number of points to be purchased (equivalent of SHARES).
PIPSIZE	Pipsize	Size of a pip (forex) : PIPSIZE = POINTSIZE.
PIPVALUE	PipValue	Value in €/$ of a pip (or point), PipValue=Pointvalue.
POINTSIZE	PointSize	Size of a pip (or point): PIPSIZE = POINTSIZE.
POINTVALUE	PointValue	Value in €/$ of a pip (or point), PipValue=Pointvalue.
POSITIONPERF	PositionPerf(n)	Indicates the percentage gain or loss of the nth closed position.
POSITIONPRICE	PositionPrice	Indicates the average price of the current position.
POW	POW(N,P)	Returns the value of N at power P.
PRELOADBARS	DEFPARAM PRELOADBARS = 200	Indicates the maximum amount of preloaded bars for the calculation of indicators used in a trading system.
PRICE	SET TARGET PRICE x	Allows to set a profit target (or stop loss) at a given price x.
PRINT	PRINT x	Displays the variable in its own window, useful for debugging.
PROFIT	SET TARGET PROFIT x	Allows to set a profit target at x units from the position entry price.
%PROFIT	SET TARGET %PROFIT x	Defines a gain at x% from the position entry price.
$PROFIT	SET TARGET $PROFIT x	Defines a gain of x €,$ (currency of the instrument).
PPROFIT	SET TARGET PPROFIT x	Defines a gain at x point of the entry price in position.

CODE	SYNTAX	FUNCTION
PositiveVolumeIndex	PositiveVolumeIndex(price)	Positive Volume Index indicator.
PriceOscillator	PriceOscillator[S,L](price)	Percentage Price oscillator.
PRTBANDSUP	PRTBANDSUP	Gives the value of the upper band of PRTBands.
PRTBANDSDOWN	PRTBANDSUP	Gives the value of the lower band of PRTBands.
PRTBANDSHORTTERM	PRTBANDSSHORTTERM	Gives the value of the short term band of PRTBands.
PRTBANDMEDIUMTERM	PRTBANDSMEDIUMTERM	Gives the value of the long term band of PRTBands.
PVT	PVT(price)	Price Volume Trend indicator.
QUIT	QUIT	Instruction to stop a trading system.

R

CODE	SYNTAX	FUNCTION
R2	R2[N](price)	R-Squared indicator (error rate of the linear regression on price).
RANDOM	RANDOM(Min, Max)	Generates a random integer between Min and Max bounds (included).
Range	Range[N]	Returns the range (High – Low) of the current candle.
Repulse	Repulse[N](price)	Repulse indicator (measure the buyers and sellers force for each candle).
RepulseMM	RepulseMM[N,Period,factor](price)	Moving Average line of the Repulse indicator.
ROC	ROC[N](price)	Price Rate of Change indicator.
RocnRoll	RocnRoll(price)	Designates the RocnRoll indicator based on the ROC indicator.
ROUND	ROUND(a)	Mathematical Function "Round a to the nearest whole number".
RSI	RSI[N](price)	Relative Strength Index indicator.

S

CODE	SYNTAX	FUNCTION
SAR	SAR[At,St,Lim]	Parabolic SAR indicator.
SARatdmf	SARatdmf[At,St,Lim](price)	Smoothed Parabolic SAR indicator.
SELL	SELL (n) SHARES	To close a long position.
SELLSHORT	SELLSHORT (n) SHARES	To open a short position.
SET	SET	Allows you to set a stop or a limit.
SHARES	BUY (n) SHARES	Designates the number of shares to be purchased.
SHORTONMARKET	SHORTONMARKET	Indicates whether you have short positions in the market.
SHORTTRIGGERED	SHORTTRIGGERED[N]	Indicates whether a short position opening has taken place on the Nth previous candle.
Second	Second[n]	Returns the second of the bar n periods before the current bar.
Seconds	TIMEFRAME(X Seconds)	Sets the period to "X Seconds" for further code calculations.
SenkouSpanA	SenkouSpanA[T,K,S]	Returns the SenkouSpanA value of the Ichimoku indicator.
SenkouSpanB	SenkouSpanB[T,K,S]	Returns the SenkouSpanB value of the Ichimoku indicator.
SIN	SIN(a)	Mathematical Function "Sine" ('a' argument in degrees).
SGN	SGN(a)	Mathematical Function "Sign of" a (it is positive or negative).
SMI	SMI[N,SS,DS](price)	Stochastic Momentum Index indicator.
SmoothedRepulse	SmoothedRepulse[N](price)	Smoothed Repulse indicator.
SmoothedStochastic	SmoothedStochastic[N,K](price)	Smoothed Stochastic indicator.
SQUARE	SQUARE(a)	Mathematical Function "a Squared".
SQRT	SQRT(a)	Mathematical Function "Squared Root" of a.
STD	STD[N](price)	Statistical Function "Standard Deviation".

CODE	SYNTAX	FUNCTION
STE	STE[N](price)	Statistical Function "Standard Error".
Stochastic	Stochastic[N,K](price)	%K line of the Stochastic indicator.
Stochasticd	Stochasticd[N,K,D](price)	%D line of the Stochastic indicator.
STOP	SET STOP LOSS	Allows you to place a stop order (see LOSS glossary entry).
summation	summation[N](price)	Sums a certain price over the N last candles.
Supertrend	Supertrend[STF,N]	Super Trend indicator.

T

CODE	SYNTAX	FUNCTION
TAN	TAN(a)	Mathematical Function "Tangent" of a ('a' argument in degrees).
TARGET	SET TARGET PROFIT x	Allows you to set a profit target order at x units from price.
TEMA	TEMA[N](price)	Triple Exponential Moving Average.
TenkanSen	TenkanSen[T,K,S]	Returns the TenkanSen value of the Ichimoku indicator.
THEN	See IF/THEN/ELSE/ENDIF	Instruction following the first condition of "IF".
Ticks	TIMEFRAME(X Ticks)	Sets the period to "X Ticks" for further code calculations.
Ticksize	Ticksize	Minimum price variation of the instrument in the chart.
Time	Time[N]	Represents the time (HHMMSS) of each bar loaded in the chart.
TimeSeriesAverage	TimeSeriesAverage[N](price)	Temporal series moving average.
Timestamp	Timestamp[N]	UNIX date of the close of the Nth previous candle.
TO	See FOR/TO/NEXT	Directional Instruction in the "FOR" loop.
Today	Today	Today's date (YYYYMMDD format).
TotalPrice	TotalPrice[N]	(Close + Open + High + Low) / 4
TR	TR(price)	True Range indicator.
TRADEINDEX	TRADEINDEX(n)	Indicates the index of the bar on which the nth last order was executed.
TRADEPRICE	TRADEPRICE(n)	Indicates the price level of the nth last order was executed.
TRAILING	SET STOP TRAILING x	Place a trailing stop at x units from the price.
%TRAILING	SET STOP %TRAILING x	Place a trailing stop at x% from the price.
$TRAILING	SET STOP $TRAILING x	Place a trailing stop such that the loss is x €,$ (currency of the instrument).
TriangularAverage	TriangularAverage[N](price)	Triangular Moving Average.
TRIX	TRIX[N](price)	Triple Smoothed Exponential Moving Average.
TypicalPrice	TypicalPrice[N]	Represents the Typical Price (Average of the High, Low and Close).

U - V - W

CODE	SYNTAX	FUNCTION
Undefined	a = Undefined	Sets the value of a variable to undefined.
UnSet	UnSet($MyArray)	Resets the data in the table.
Variation	Variation(price)	Difference between the close of the last bar and the close of the current bar in %.
ViMinus	ViMinus[N]	Bottom band of the Vortex indicator.
ViPLus	ViPlus[N]	Top band of the Vortex indicator.
Volatility	Volatility[S, L]	Chaikin volatility.
Volume	Volume[N]	Volume indicator.
VolumeAdjustedAverage	VolumeAdjustedAverage[N](Price)	Volume Adjusted Moving Average.
VolumeOscillator	VolumeOscillator[S,L]	Volume Oscillator.
VolumeROC	VolumeROC[N]	Volume of the Price Rate Of Change.
Weeks	TIMEFRAME(X Weeks)	Sets the period to "X Weeks" for further code calculations.

CODE	SYNTAX	FUNCTION
WeightedAverage	WeightedAverage[N](price)	Refers to the Weighted Moving Average.
WeightedClose	WeightedClose[N]	Average of (2 * Close), (1 * High) and (1 * Low).
WEND	See WHILE/DO/WEND	Ending Instruction of WHILE loop.
WHILE/DO/WEND	WHILE (condition) DO (action) WEND	WHILE loop.
WilderAverage	WilderAverage[N](price)	Represents Wilder Moving Average.
Williams	Williams[N](close)	Returns the %R Williams indicator.
WilliamsAccumDistr	WilliamsAccumDistr(price)	Accumulation/Distribution of Williams Indicator.

X - Y - Z

CODE	SYNTAX	FUNCTION
XOR	a XOR b	Logical Operator exclusive OR.
Year	Year[N]	Returns the year of the bar n periods before the current bar.
Years	TIMEFRAME(X Years)	Set the timeframe to "X Year(s)" for further code calculations.
Yesterday	Yesterday[N]	Date of the day preceding the bar n periods before the current bar.
ZigZag	ZigZag[Zr](price)	Zig-Zag based on Elliott wave theory.
ZigZagPoint	ZigZagPoint[Zp](price)	Zig Zag calculated at Zp points.

Other

CODE	FUNCTION	CODE	FUNCTION
+	Addition Operator	<>	Difference Operator
-	Subtraction Operator	<	Strict Inferiority Operator
*	Multiplication Operator	>	Strict Superiority Operator
/	Division Operator	<=	Inferiority Operator
=	Equality Operator	>=	Superiority Operator


ProRealTime