Visual JForex

Version 1.6.39

USER GUIDE

Revision 1

October 2016

DUKASCOPY
Swiss Forex Bank & Marketplace

Table of contents

Disclaimer

The present user guide aims to describe and explain accurately Visual JForex platform. The user guide is updated as regularly as possible to cope with the high pace of development of the trading instances.

Although the user guide is updated on regular basis, users must be fully aware that it may not contain a description of all processes and features as the first evolve over time, and the latter are added, removed, or modified through the development process. By the same token, some descriptions, explanations and other notices may be outdated and not reflecting as a consequence, current processes and procedures.

Therefore, clients/users are strongly encouraged to explore this document while testing the different features with a demo account. The trading environment, features and functioning of the platform as well as various surrounding issues must have been reviewed by the clients/users, all possible remaining questions should have been asked to and answered by Dukascopy before trading live and managing real equity.

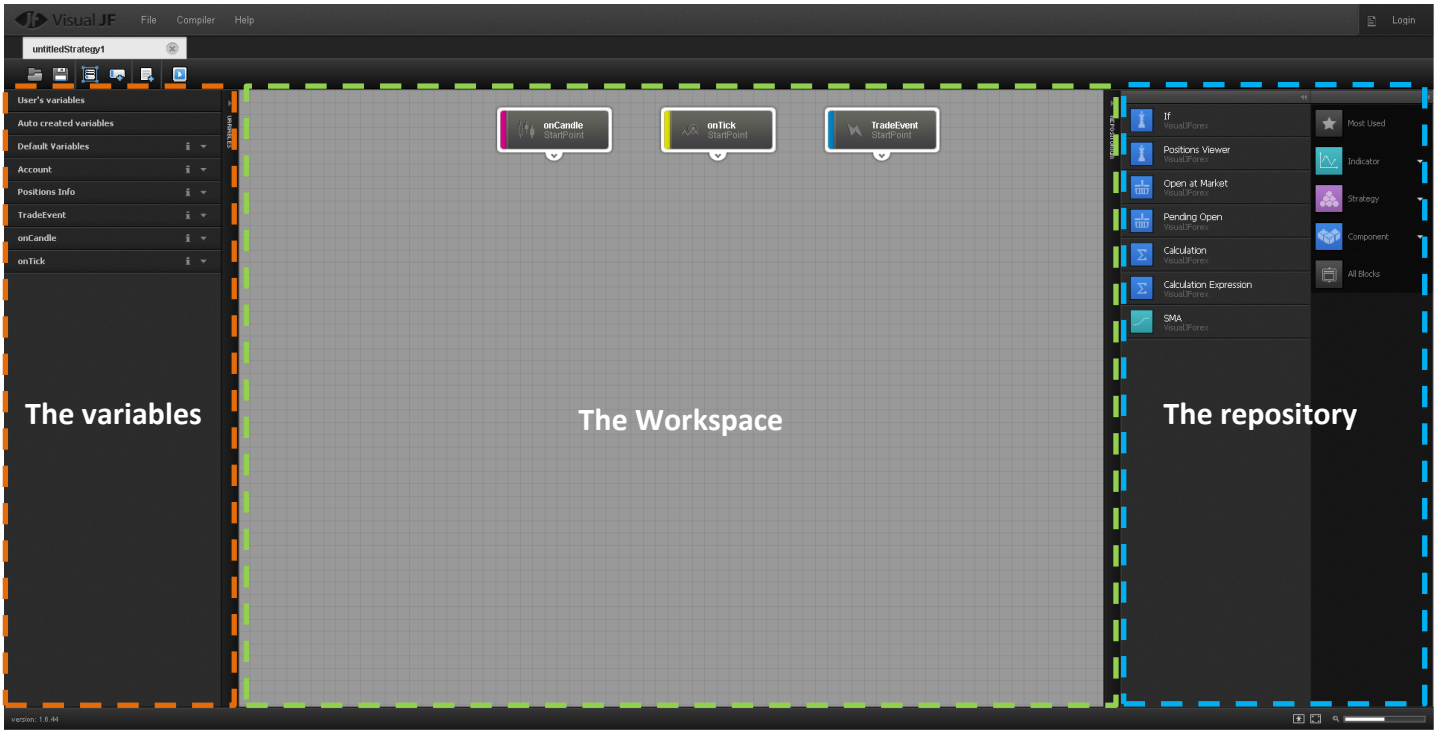
Dukascopy is neither responsible nor liable for any damage including but not limited to trading losses or loss of profit deriving from the use, possible incompleteness, inaccuracies and obsolescence of the user guide. The user guide shall not be construed as a trading advice, an investment advice or a recommendation to execute or to not execute specific strategies on Dukascopy's platforms.

The present user guide is not intended to cover any and all aspects of the business relationship between Dukascopy and its clients, prospects and/or other partners. For information about the frame in which Dukascopy operates with respect to the market and with respect to clients, prospects and other partners, the reader is encouraged to consult Dukascopy website, and/or to contact Dukascopy's representatives.

What is Visual JForex?

Visual JForex is a comprehensive solution to build, test and run strategies; it is based on a user-friendly interface using drag and drop feature. It has a unique way to develop fully automated strategies using blocks linked to each other to build a complete and ready-to-execute forex strategies. The platform is web based presenting a flash user interface with a Java backend.

Visual JForex platform

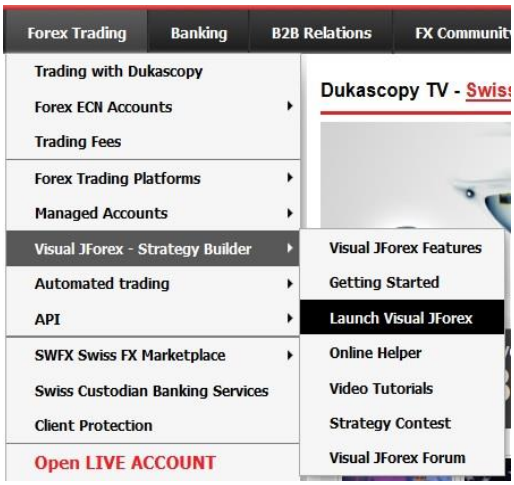


Requirements:

- Operating systems: Windows, Linux, Apple OS x
- Mozilla Firefox, Google Chrome, MS explorer or Safari.
- Java 1.7 and above
- Flash Player

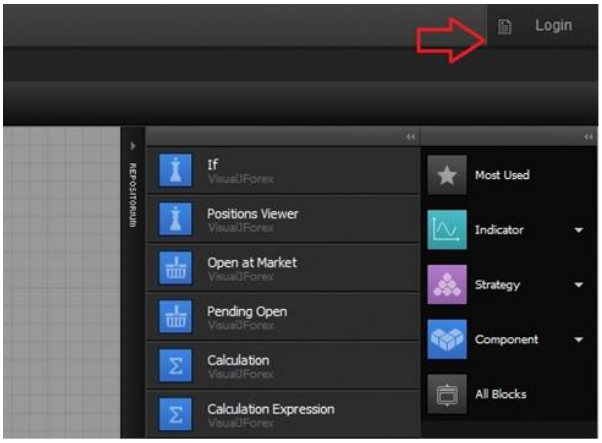
1. Starting Visual JForex platform

1.1. Logging on:



To start Visual JForex platform navigate to www.dukascopy.com and click on “Forex Trading” > Visual JForex –Strategy builder.

Most internet browsers support embedded flash plugins and we recommend the use of Mozilla Firefox, Google Chrome, MS explorer (including Edge) or Safari.



Once the platform launched the user has the choice to start with logging from the upper right corner using his community credentials although this is not mandatory to use the platform. A community account can be created from our website.

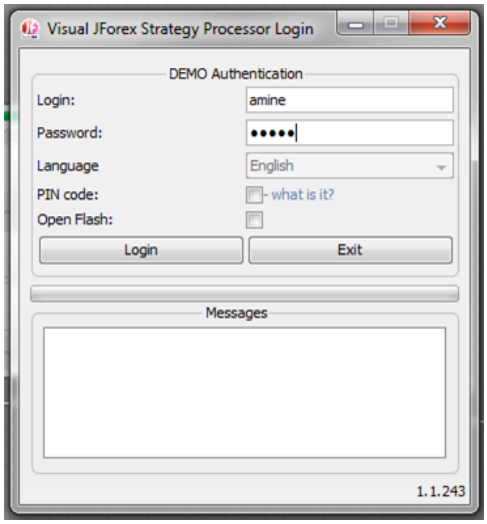
1.2. Starting and logging to strategy processor:

Visual JForex is connected to a trading account using a bridge that links the platform to the JForex account. This bridge is called VJF processor and consists of a java process that connects both platforms and transfers the necessary data to the historical testing.

➔ *The strategy processor cannot be launched if there is no strategy loaded or built.*

After building a strategy, click on “Compiler” menu and “Run” or press the button  to launch the strategy processor. At the first time the platform will attempt to connect to the strategy processor and it is normal to return an alert message notifying that the processor is not available.

Click on Download to start the strategy processor. A java file will be downloaded and executed using Java program (java webstart process) then a login window should appear:



Input your JForex DEMO credentials to establish the connection between Visual JForex and the JForex trading account.

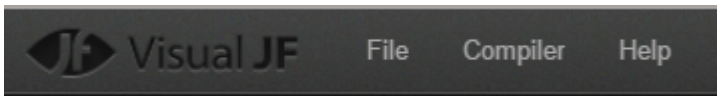
This connection allows the platform to run strategies in both real-time run and historical tester mode.

The connection process is made only once per session. The same procedure should be repeated if the strategy processor is closed. In case of a disconnection it will reconnect automatically, an alert message will notify the user.

Alternatively the processor can be downloaded directly from this link.

2. Menu, workspace and tool bar

2.1. The Menu



2.1.1.File

To create a new strategy click on File > New you'll be asked to give a strategy name to your new file so that it can be loaded afterwards. A new tab will be created with the name of the strategy.

To open a saved file, click File > open Draft. The user must be connected to be able to access the file location. Click Save Draft to save your work.

➔ *Free disk space is offered in our servers to save Visual JForex strategies*

Import and export functions are used to save/load strategies in a specific file format that can only be read with Visual JForex platform, this file has a ".vfs" extension. The user also has the option to export the file in Java format.

2.1.2.Compiler

The compiler menu allows running a strategy either in real time or back-testing. It is also possible to build the strategy which means to compile it to get an exhaustive Java code that could be used directly in the JForex platform. Viewing the source code on-the-fly is made with the option "View source". The "Contest" option is a shortcut to push a strategy to be run in Dukascopy Strategy Contest

2.1.3.Help

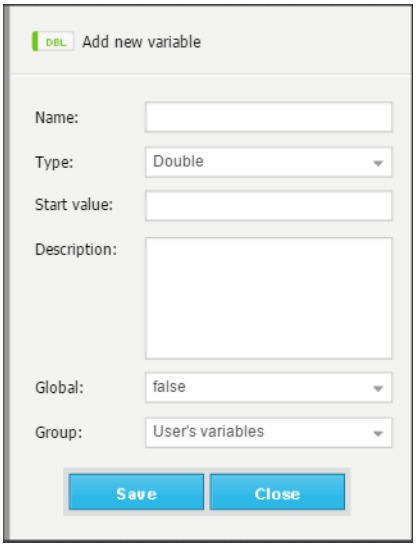
The Help menu will route the user to our online documentation or to request live chat support.

2.2. The Tool bar

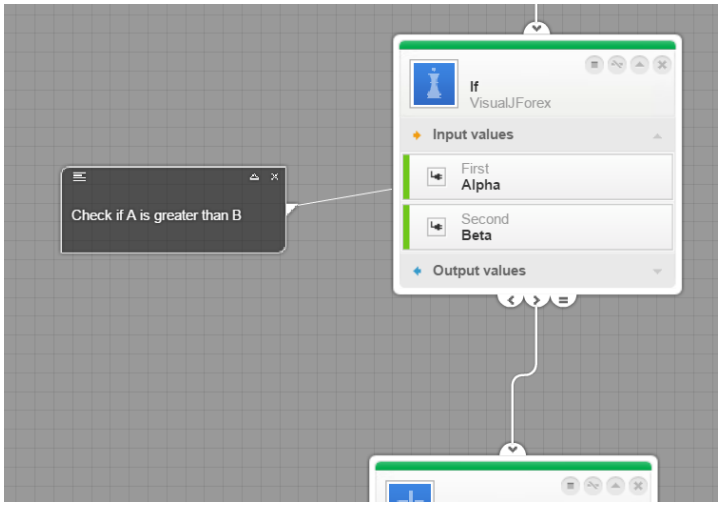
The tool bar offers some shortcuts to open, save files in one click and also to run a strategy. It has also three additional functions:

Create new group  Used to add a new group to organize custom variables. The new group will be displayed as a sub-section in the variable's left panel.

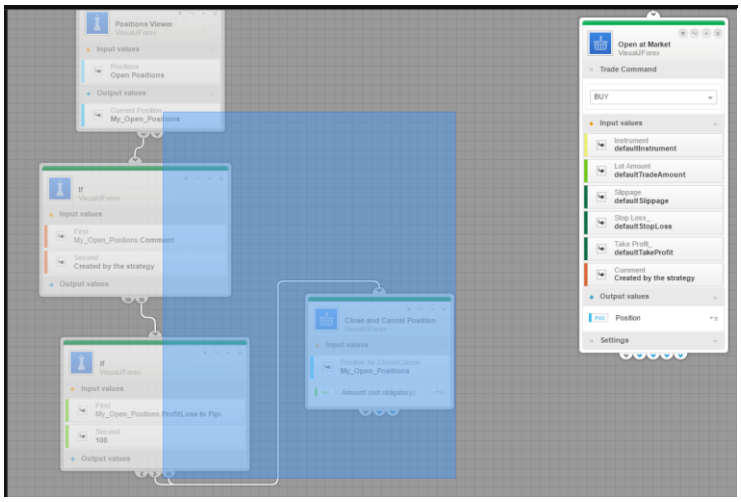
Create new variable  Press this button to add a new variable. A new window pop-up where the variable attributes should be defined (Section 6).



Insert a note  Click on this button to insert a note into the workspace; a note can be linked to a block or kept independent.

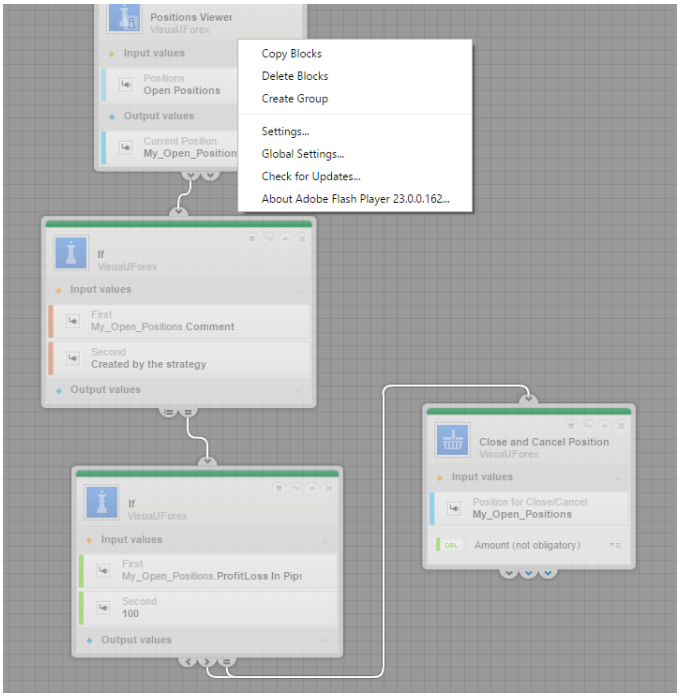


2.3. Miscellaneous options

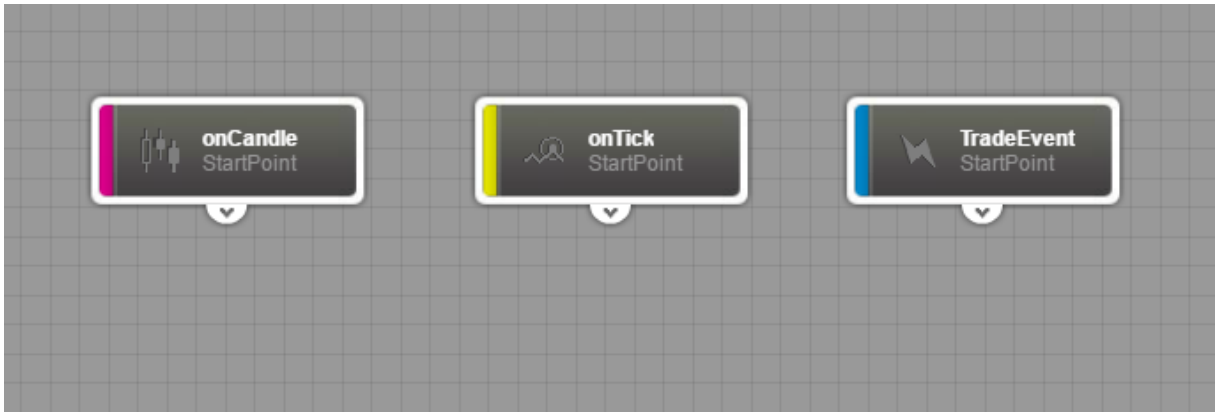


Hold Ctrl button and click to select multiple blocks. It is possible to move a set of selected blocks inside the workspace; deletion can be made as well when pressing “delete” from the keyboard.

Right click to display contextual menu to copy a group of blocks, delete them or create a group with the selected ones.



3. The start points



"Start points" are the 3 blocks that are displayed in the workspace when the platform is opened. They represent trading methods on which any strategy should start. Once connected from the exit pin they are automatically activated. A start point which is not connected remains inactive and thus its method will not be used. These blocks cannot be removed from the workspace.

➔ *A strategy can use one, two or three start points depending on its logic and requirements.*

As illustrated, there are 3 possible start points: On Candle, On Tick and Trade Event.

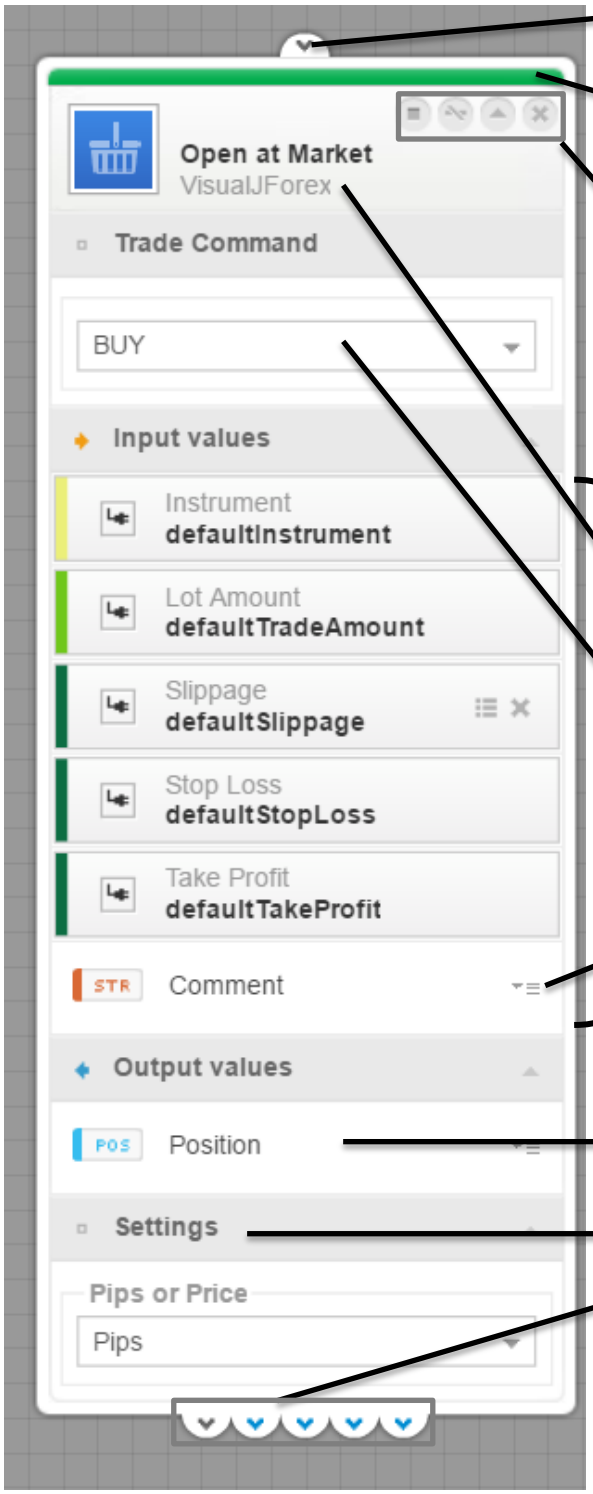
- **On Candle**: This method is based on candle periods and it automatically handles periods of 10 seconds, 1 minute, 5mn, 10mn, 15mn, 30mn, 1 hour, 4 hours, 1 day, 1 week and 1 month. Any of the listed periods can be selected from the list of choices in "Default Period" variable type (more information in 6.2).
  The candle start point is developed to handle the above listed periods dynamically and it is activated with a flashing red light at the end of every period. In other words, this start point is subscribed by default to all periods and it is then flashing every 10 seconds (minimum candle period available by default) when no period filtration is implemented by the user.
  The variables listed in the left section "On Candle" (section 6.1.7) belong to this start point therefore they get values dynamically when the start point flashes.

- **On Tick**: This method is based on tick prices received for a given instrument. By default, the on tick method is subscribed to EUR/USD pair but this can be changed in the Default variables section (section 6.1.3).
  Similarly to the On Candle, the On Tick variable section in the left panel handles data related to tick prices and it gets filled automatically when the On Tick start point is activated (flashes in red).

- **Trade Event**: This method is based on the messages received by the platform and gets activated every time a specific message is received.

4. The block

A block is a user-friendly box used to build a logic function or to insert ready-made indicators as well as trading commands. Each block is unique and has its own functionality such as executing a logical comparison, performing a trading command or implementing an indicator.



Entry: This is the entry point of the block where a connection is made

Status bar: When green the block is ready to be executed, when red an input or output variable should be completed or created.

The tool bar (form left to right):
Break point: This is an option that breaks the strategy when the system reaches this block. It is mainly used for troubleshooting to halt the strategy and verify functions and variable values. The strategy can be resumed afterwards.
Disconnect block: remove all links connected to the block in one click.
Minimize: Collapse the block to reduce its size for better workspace ergonomics.
Remove: remove permanently the block form the workspace.

Block name

Trade command: This is specific to such trading command blocks; it lists the available order types.

Input variables: Variables used as input to fulfill the block execution.

Quick access to variables: Shortcut to access the possible variables compatible with the related field.

Output variable(s): Field where the output variable is either automatically or manually created.

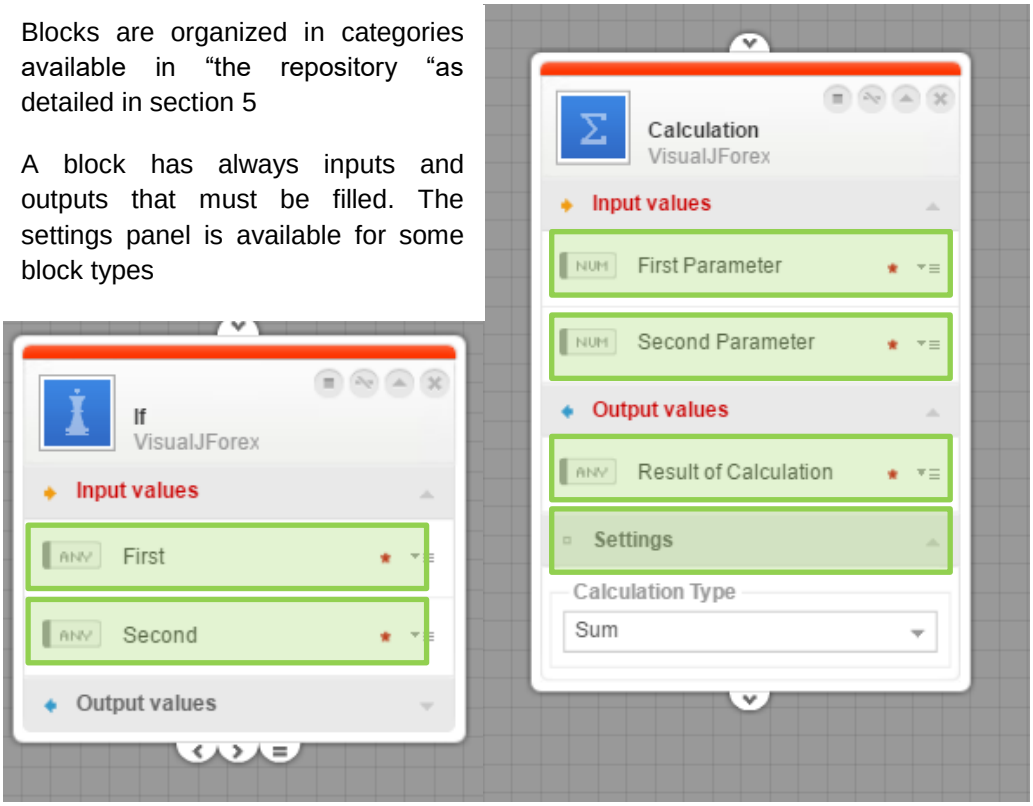
Settings: list of block parameters.

Exit(s): This is where the exit link is/are made. A block may have only one or multiple exit links; it must be connected to be fully functional.
Blue exits refer to order status messages that are also used in Trade Event start point. Gray exits are used within the strategy logic to link a block to another one thus they do not refer to any order status or acknowledgment message.

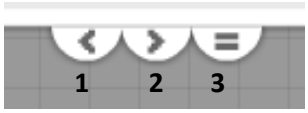
➔ *Any block must be connected at its entry and exit pins in order to be fully functional otherwise the block instruction will not be considered by the system.*

Blocks are organized in categories available in "the repository "as detailed in section 5

A block has always inputs and outputs that must be filled. The settings panel is available for some block types

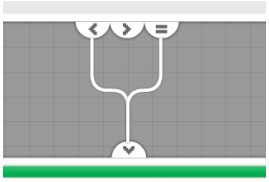


To execute the condition/operation called by the block the input parameters must be filled. The output is also a mandatory variable that is available by default or created manually.



When the block is connected out of the exit with another block, it becomes active and the system will then be able to go through it to execute the requested command.
Exit 1: First input less than second input
Exit 2: First input greater than the second input
Exit 3: First input equal to the second input
A tooltip is always available when placing the mouse on a given exit or area of the block

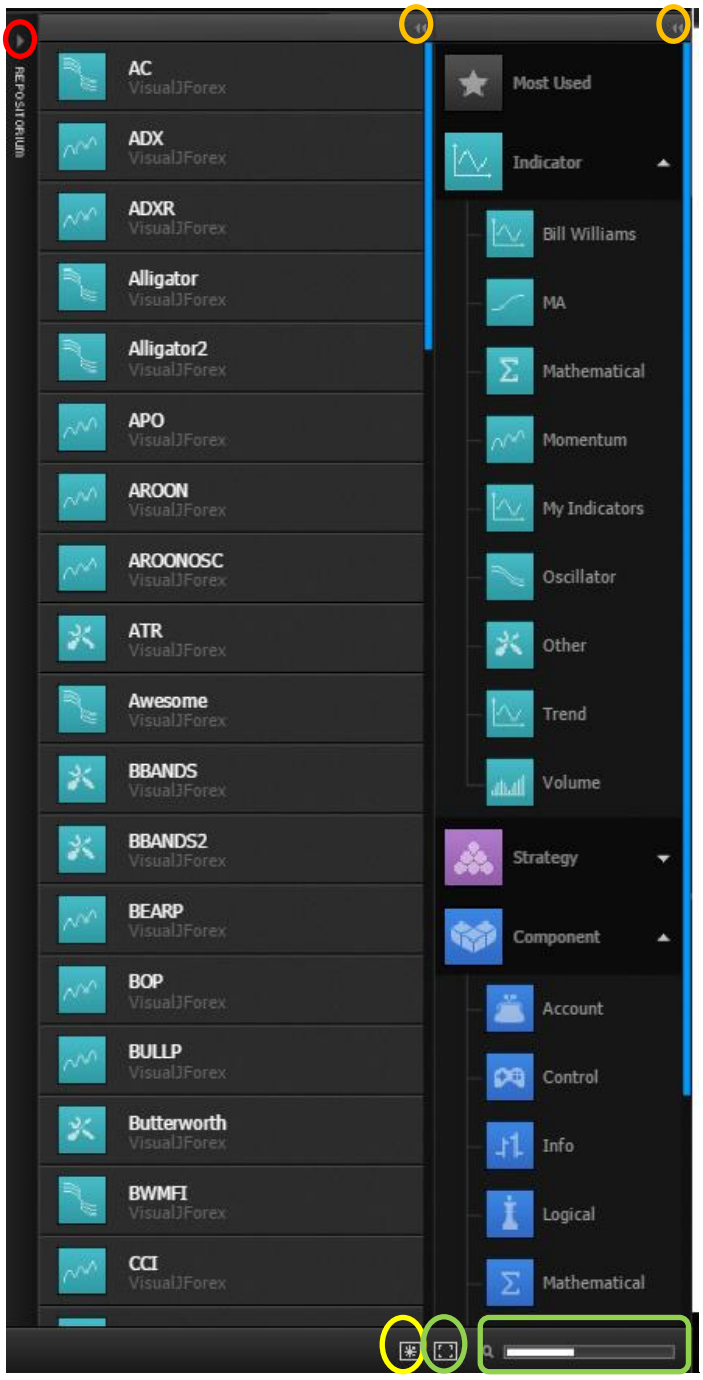
➔ *Multiple links are allowed out of the block exits*



5. The repository

The repository is the panel displayed at right side of the platform; it's a sliding panel dedicated to functions and components. These items are displayed as icons that can be dragged and dropped into the workspace in the form of block. The repository is organized into 3 main categories:

- list of the indicators by type
- Strategies including user's strategies in "My strategies"
- Components where the entire list of the logical blocks is stored and arranged by category

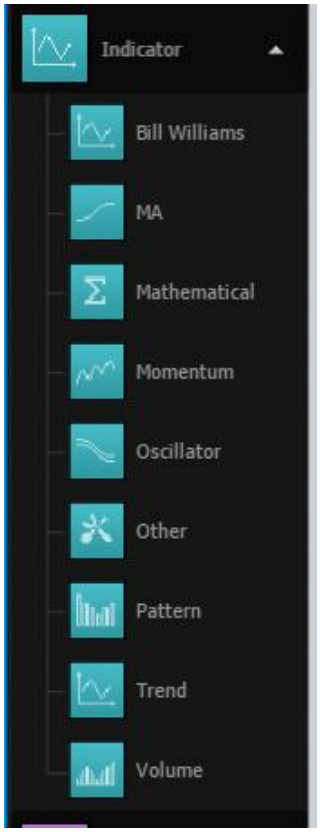


The entire section can be collapsed when clicking on the red circled button. It can be also reduced to optimize the workspace surface when clicking on the orange circled buttons.

The indicators section contains ready-made indicators to be used simply by drag & drop into the workspace.

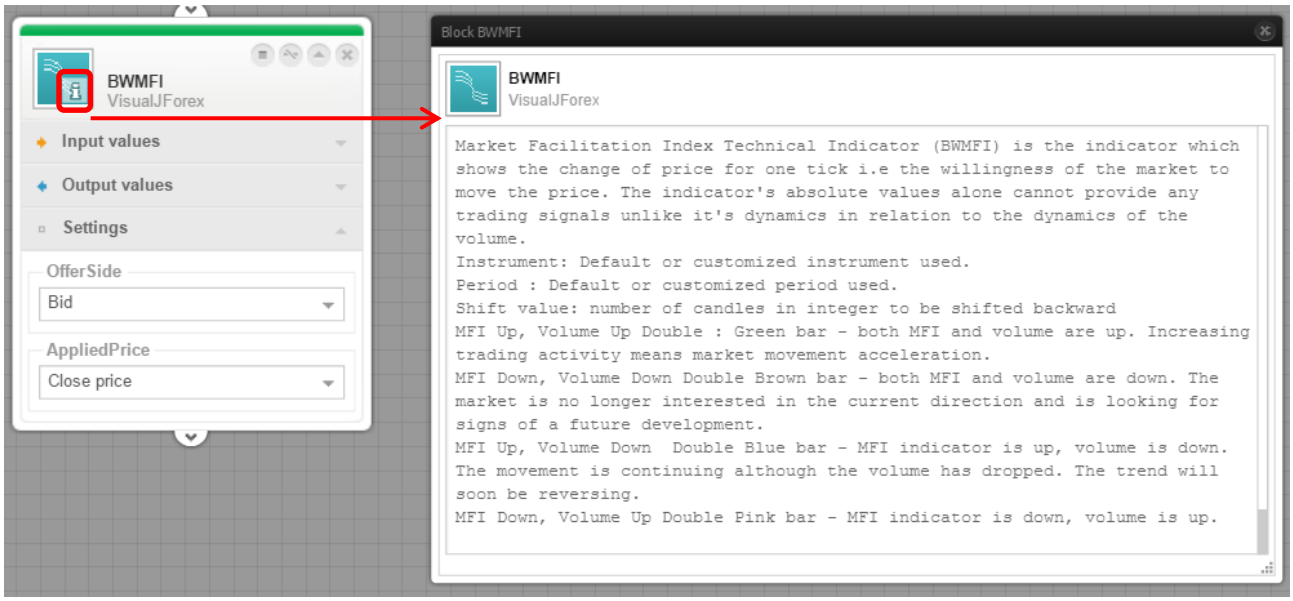
Click on the Navigation icon (Yellow circled) to display the workspace preview which is a nice tool to locate blocks and strategy sections especially for complex strategies that require large amount of blocks. The Zoom button and scale (green) helps setting back the zoom to 100%. Zooming in and out is made using the scroll wheel of the mouse but also by dragging on the scale.

5.1. The indicators:



Visual JForex offers a large list of indicators which are sorted in 9 categories: “Bill Williams”, “Moving Average (MA)”, “Mathematical”, “Momentum”, “Oscillator”, “Pattern”, “Trend”, “Volume” and “Other”.

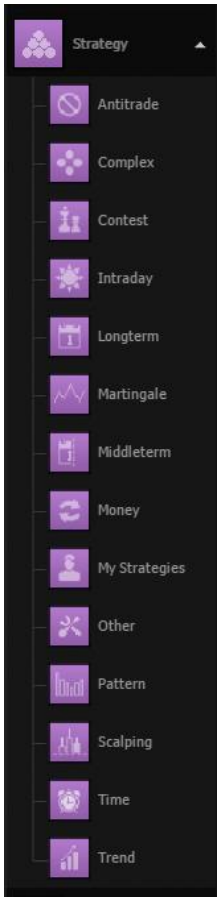
Every class of indicator has its set of indicators which are displayed with a short name. The complete name of the indicator is seen when clicking on the tool tip in the upper left corner of the block. If this description is not available, the indicator can be alternatively identified from JForex platform library window.



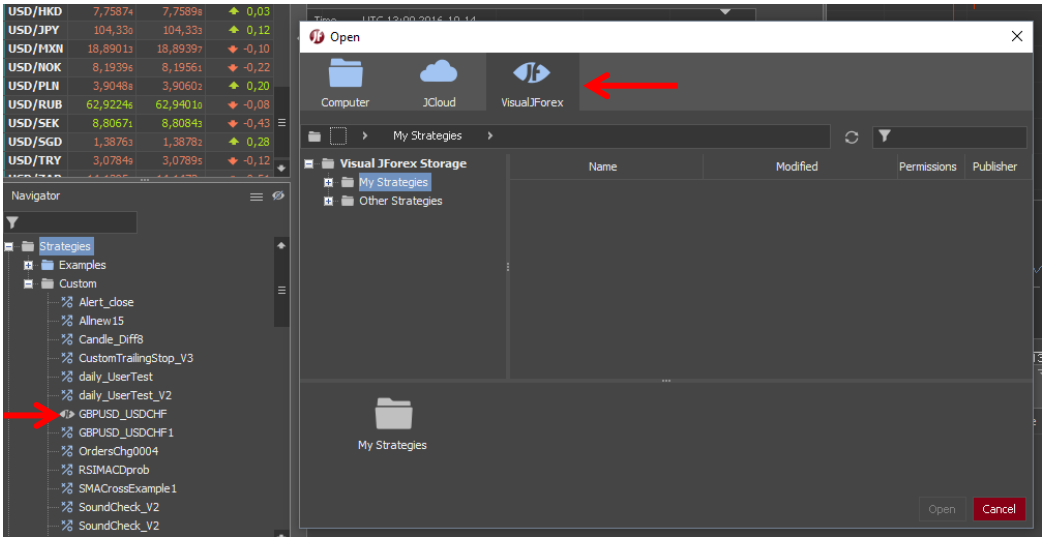
Almost all indicators display a settings section with two parameters: The offer side where the user selects which side the calculation is based on, the bid or the ask. The second parameter is the calculation method of the indicator shown as “Applied price”. Several methods are available offering simple and aggregated price inputs.

5.2. Strategy:

The strategy section displayed in purple displays strategy groups sorted by theme.



This section encloses several subsections sorted by trading idea. All of them are public, only the section “My strategies” is private and can be accessed only when the user is logged on.
The most used item of this list is the section “My strategies” where the compiled strategies are saved and can be loaded by right-clicking on the strategy icon and select “Load VFS” to open the strategy in the workspace or “View source” to display the java code in a pop-up window. A strategy can also be deleted from the same contextual menu.
When a strategy is compiled it will be saved in “My Strategies” section and can be loaded from JForex platform as well as the mobile application SWFX Trader.

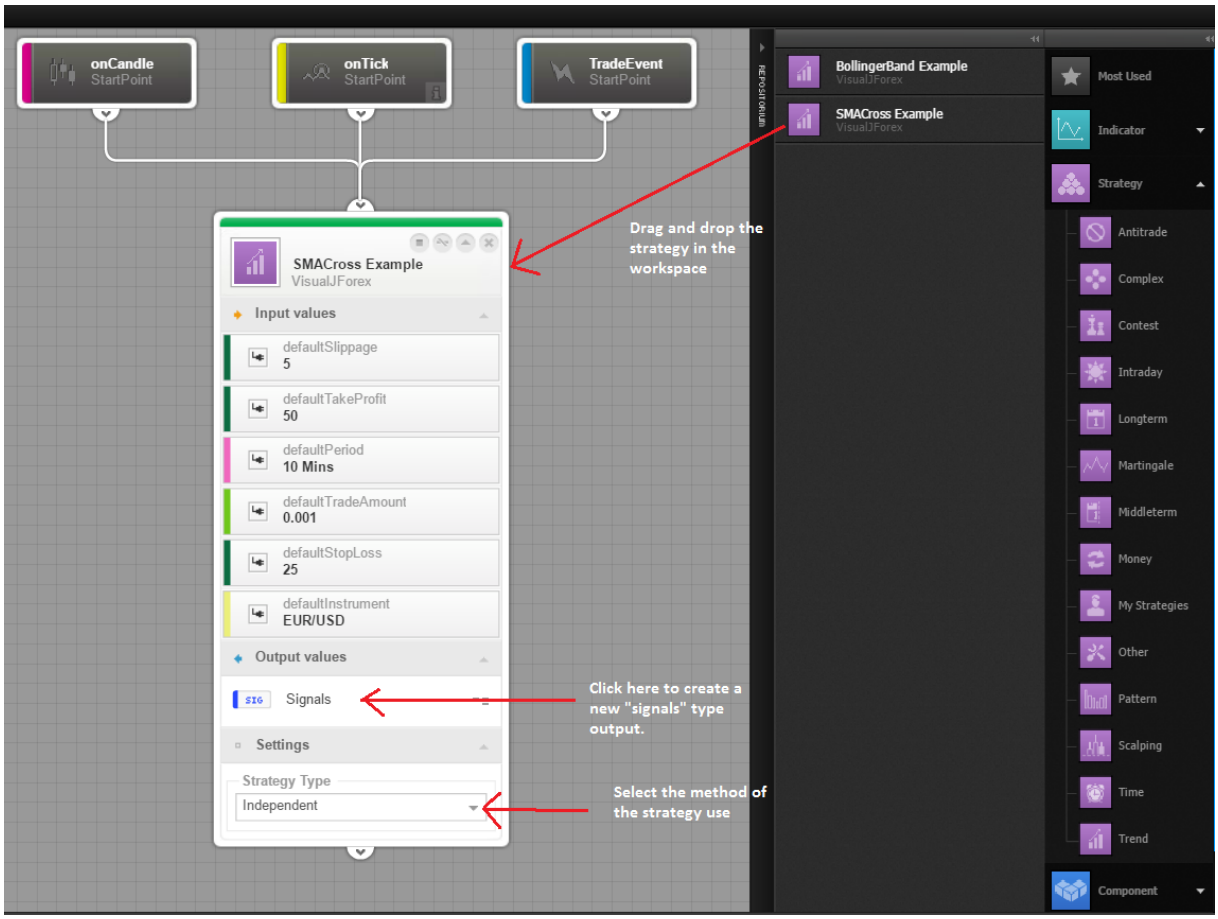


Strategies are available under the tab ‘Visual JForex’ and when loaded it can be identified with a specific icon in the list of strategies (example: GBPUSD

What is a strategy used as a block?

This is a simplified way to optimize the workspace and display the entire strategy as a single block to generate signals or even to be run normally. A signal is the capture of the trading instructions initiated by the strategy in addition to information related to the positions and the orders created.
Such use is also practical when it comes to the utilization of ready-made strategies to benefit from the generated trading instructions and add additional conditions and enhancements. Changing the strategy parameters in one click is also made easier as they are available as input variables in the strategy block.

To use a strategy as a block and generate signals, simply drag and drop the strategy in the workspace and create a new “Signals” output then select “generate signals” parameter in the block settings.

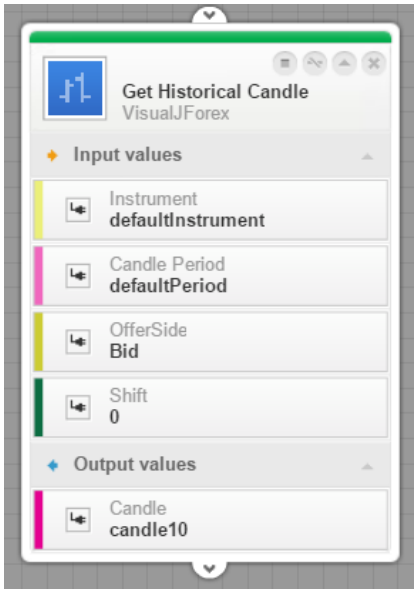


It is mandatory to connect the strategy used as a block to the three start points (On Candle, On Tick and Trade Event) in order for it to work properly.
The output variable is a “Signal” type (section 6.2.2) which is an array of data that saves the information related to these signals. Such variable is “embedded” and the data breakdown is obtained with the help of “Loop Viewer” block. To do so, connect a loop viewer block and use the ‘signals’ output newly created as first input. Create a new output in the loop Viewer block to breakdown and save your Signals (Loop cycle detailed in section 7.6).
When running a strategy in such way the user can either select the “Independent” mode or the “Generate signals” mode; the first mode is similar to running the strategy when opened normally (full blocks) and the second mode generate only trading signals without effective trading command. Signals are a backend notification of a trading command used by the strategy; the user can then use signals as a trigger for further conditions.

5.3. Components

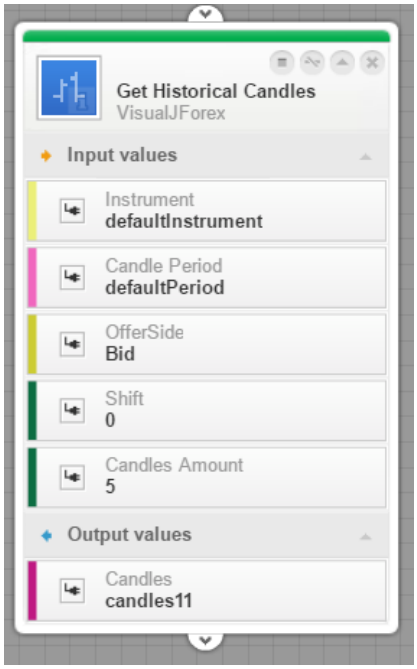
The last blue sub-section of the repository is dedicated to the components; they are sorted by category as well. Components are the functional and technical blocks used to build logical conditions and statements as well as mathematical and trading commands and operations.

5.3.1.Info:



Get Historical Candle: To retrieve information related to the current candle still in process or a previous candle already completed. The choice is made with the help of "Shift" parameter which is set by default to 0 in reference to the current candle. The user can select the instrument, the candle period, the side; the output is saved in an array variable created automatically having the type "Candle" (6.2.2)

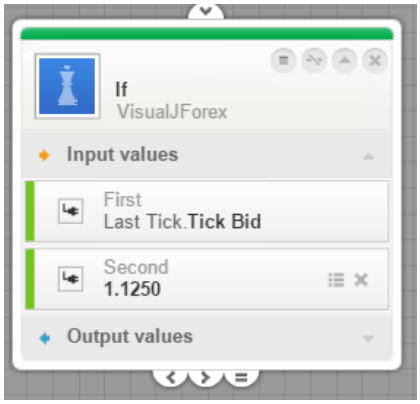
The output will be available under "Auto created variables" and encloses the Open, High, Low, Close prices as well as the volume, the candle period and the candle start time.



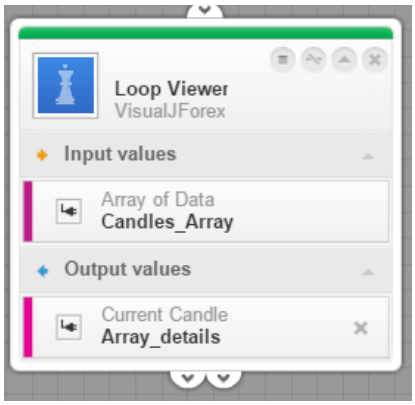
Get Historical Candles: In line with the previous block which is handling single candle information, this block offers the ability to retrieve multiple candles data with the help of the input "Candle Amount" that specifies the number of candles to be called and saved. The output is indeed an array that stores this information in an embedded variable to be used in "Loop Viewer" block to get a new variable breakdown.

Such block is used when a large number of candles are needed and which can be difficult to be done with many individual "Get Historical Candle" blocks.

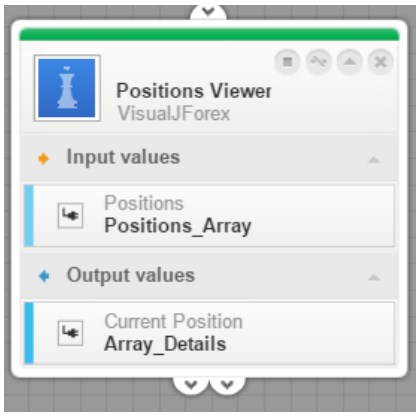
5.3.2.Logical



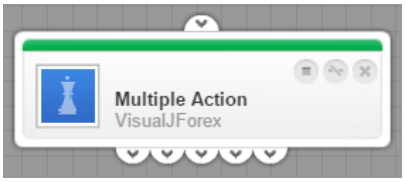
IF: Widely used in programming to perform logical conditions; it handles two inputs at a time and supports many types of variables. The block warns the user if an inconsistency is detected with regards to the variable type and suggests changing it. If an instrument variable type is used, the block will automatically change the exit pins to two (Equal / Not equal) and the second input will display a drop down list of all the tradable instruments. This is the case for variables of the following type: Boolean, Instrument, String, Offer Side, State, and Command. (More details in section 6.2)



Loop Viewer: A loop is a fundamental programming idea that is commonly used in writing programs**.** This block performs a loop or a sequence of instructions that is continually repeated until a certain condition is met. It is mostly used to breakdown array variables having the type “Candles”, “Positions” or “Signals”. In association with the block “Get historical Candles” output, it retrieves the details related to the collection of candles requested. When linked to a strategy used as block (5.5.2), the output “Signals” is dragged as an input and the full breakdown will be saved in a new output variable. When using “Positions” variable as an input it does exactly the same job as “Position Viewer” Block. The cycle check is made when using the exit at the left side and the end of the cycle (loop) is then obtained from the right exit.

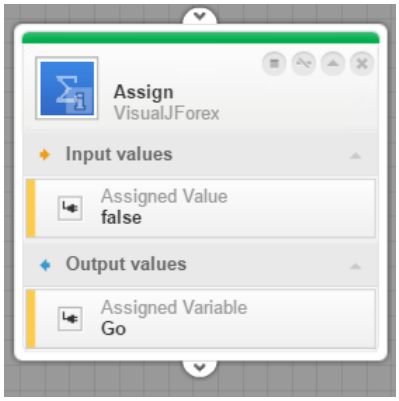


Position Viewer: it allows the user to do a sequence of instructions that is continually repeated until a given condition is reached. This will do the same job as the Loop Viewer used with positions array of data. The difference is that the Position viewer block is dedicated to Positions variables. Commonly, the input is taken from "Position Info" section in the left panel, this offers the user the option to work on a given positions status (Opened, pending or both). In order to breakdown the used input, the user should create an output variable; the result of the iteration will be saved into this new variable that will be saved in "User’s variable" section.

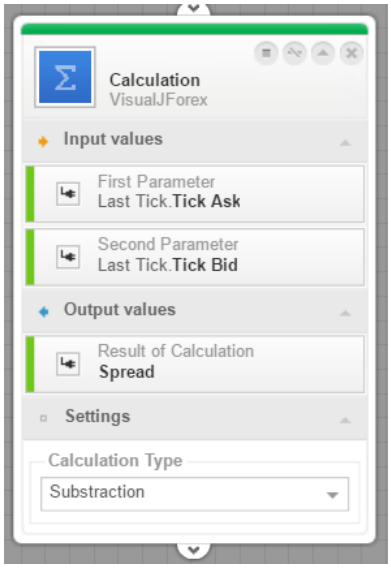


Multiple actions: This is a simple block that helps splitting the main stream into multiple flows (up to five exits); it can be repeated to multiply the distribution even further.

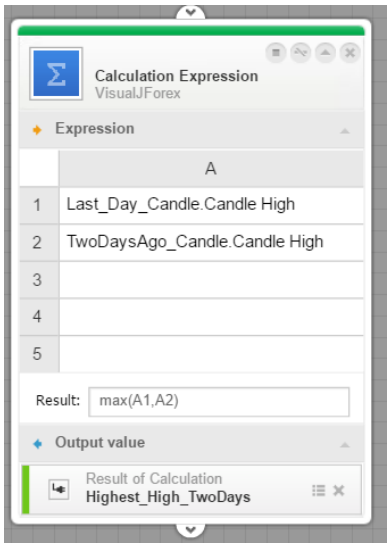
5.3.3. Mathematical



Assign: To attribute a given value to a variable with regards to its type. Once the first input is filled, the output will automatically take the appropriate variable type.



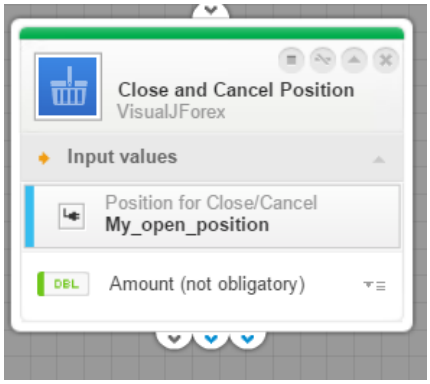
Calculation: This is for basic mathematical calculations such as addition, subtraction, multiplication and division where the result of the calculation is needed for further use. The supported variable types are Double, Integer, Date & Time (6.2.1)



Calculation Expression: Used for complex calculations and mathematical functions, this block offers a display that is similar to excel spreadsheet. The user can add a row or a column by right-clicking on the row/column header. The supported math operations are:

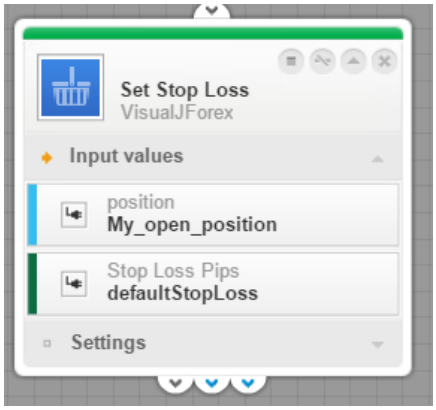
"-" Subtraction
"+" Addition
"/" Division
"*" Multiplication
"Power(X, Y)" Returns the value of X to the power of Y
"%" Modulus
"Min(X, Y)" Returns the minimum between X and Y.
"Max(X, Y)" Returns the maximum between X and Y.
"Sqrt(X)" Returns the square root of X.
"Sin(X)" Returns the sine of X, where X is assumed to be in radians
"Cos(X)" Returns the cosine of X, where X is in radians
"Tan(X)" Returns the tangent of X, where X is assumed to be in radian
"Ln(X)" Returns the logarithm base e of X
"Atan(X)' Returns the angle in radians between -pi/2 and pi/2 whose tangent is X.
"acos(X)" Returns the angle in radians between 0 and pi whose cosine is X.
"asin(X)" Returns the angle in radians between -pi/2 and pi/2 whose sine is X.
"exp(X)" Returns Euler's number raised to the power of X value.

5.3.4. Trading



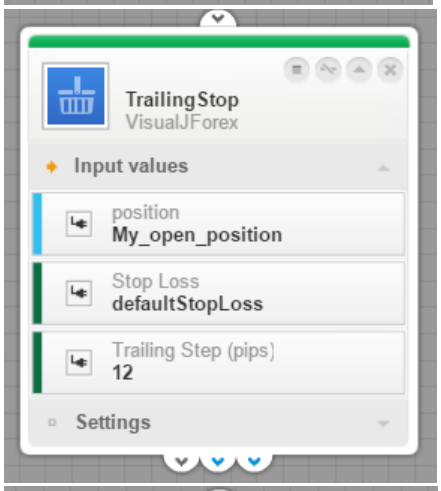
Close and Cancel Position: Depending on the position variable used as input, this block either closes open positions or cancels a placed order. To do a partial close, the second field "Amount" is filled with the position amount to be closed.
The block exits are:

- Simple exit regardless of the order acknowledgment
- Position closed or order cancelled (status OK)
- Order cancellation rejected or position closure rejected



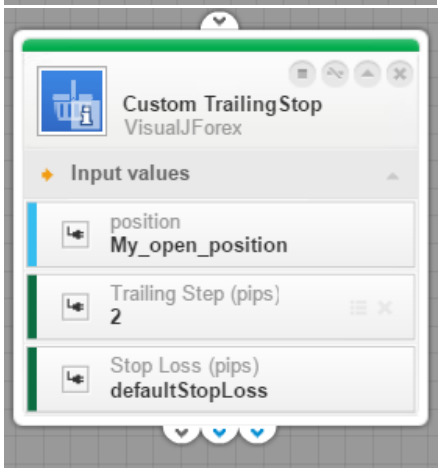
Set Stop Loss: To add a stop loss order linked to a given position which is identified with the variable "Position" as first input.
The SL order can be set in pips (integer) or as a price (Double); the choice is made from the settings menu. The trigger side can also be chosen (trigger on the BID/ASK)
The block exits are:

- Simple exit regardless of the order acknowledgment
- Order submission confirmed (Order placed)
- Order submission rejected



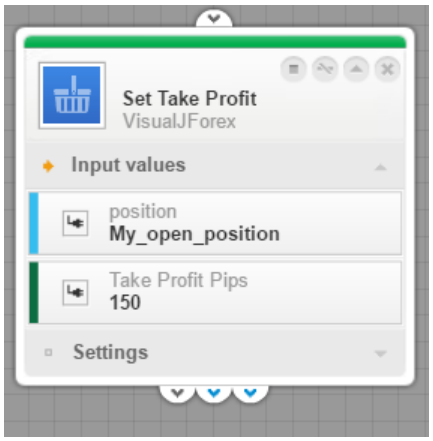
Trailing Stop: To add a trailing stop loss order linked to a given position. The initial SL order is specified in the field "Stop Loss"; the trailing step should be above 10 pips. Same as the previous SL block, the user has the choice of setting the price in pips 'integer) or as a calculated price (Double)
The block exits are:

- Simple exit regardless of the order acknowledgment
- Order submission confirmed (Order placed)
- Order submission rejected



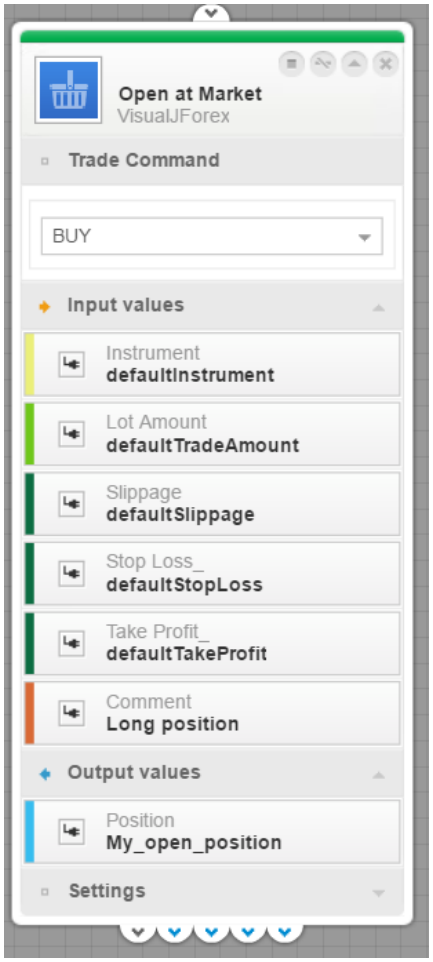
Custom Trailing Stop: In order to implement a specific trailing stop loss order having less than 10 pips as trailing step, the user can use this block in association with the position and its initial stop loss order. The block exits are identical to previous trading blocks.

*Stop loss and trailing stop blocks should be implemented in a way to be fully managed by the strategy; **the trailing process is made by the strategy and not by the server.** The strategy should be able to execute these blocks once the price condition is met.*



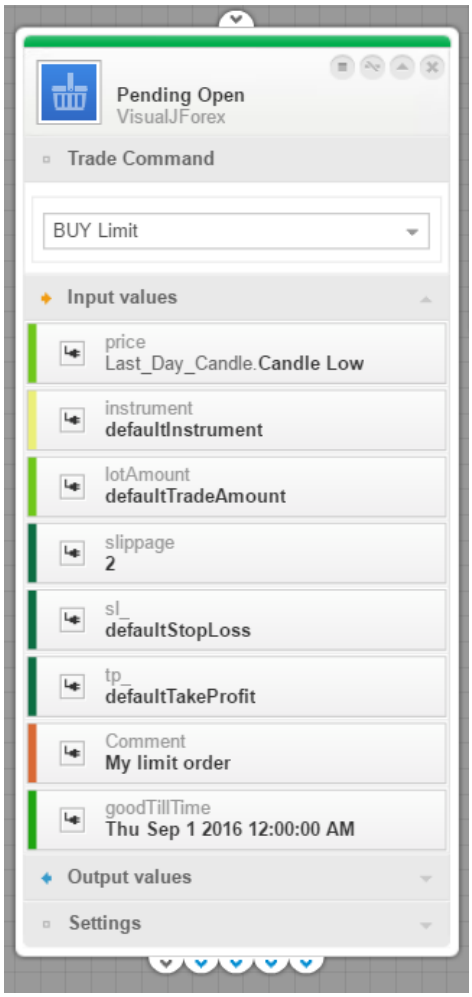
Set Take Profit: To add a Take Profit order linked to a given position which is identified with the variable “Position” as first input.
The TP order can be set in pips (integer) or as a price (Double) and the choice is made from the settings menu. The trigger side can be also chosen (trigger on the BID/ASK)
The block exits are:

- Simple exit regardless of the order acknowledgment
- Order submission confirmed (Order placed)
- Order submission rejected



Open at Market: This block is used to submit market orders. It displays the default variables at a first use but it is indeed possible to add new variables in inputs and output fields.
The order side can be selected from the trade command drop-down list. The settings section offers the possibility to set the price of the SL and TP in pips or prices. When SL/TP orders are not required the user can simply delete the variable by clicking on the “cross” button in the right corner of the variable; this is also the case for the “Slippage” variable
By default the traded amount is defined in millions (1 for one million, 0.001 for one thousand, etc.)
The comment field is not defined when the block is freshly dropped in the workspace, as this is not a mandatory input; the user can leave it empty or fill it. The “comment” will be associated to the position thus a technique to identify it.
The block exits are:

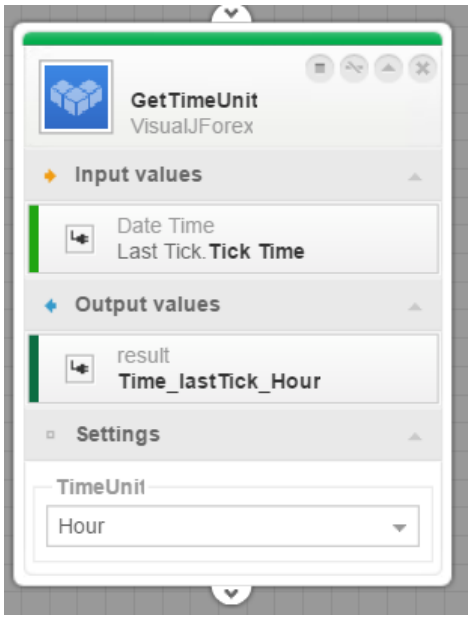
- Simple exit regardless of the order acknowledgment
- Order submission confirmed (Order placed :A very quick status for a market order as the it is instantly sent to the market)
- Order submission rejected
- Order filled
- Order execution cancelled (rare cases when an execution can be cancelled)



Pending Open: Use this block to place entry orders listed in the trade command drop down-list: Limit orders, Stop orders (with trigger side option), Market If Touched orders, Place bid and Place offer. In addition to the common input variables this block needs the price input indeed but it also has a "Time In Force" field which refer to the order validity (last input variable in the list). The block exits are:

- Simple exit regardless of the order acknowledgment
- Order submission confirmed (Order placed)
- Order submission rejected
- Order filled: when the order price is reached and the order is executed. This is detected with a "FILL" message received.
- Order execution cancelled (rare cases when an execution can be cancelled

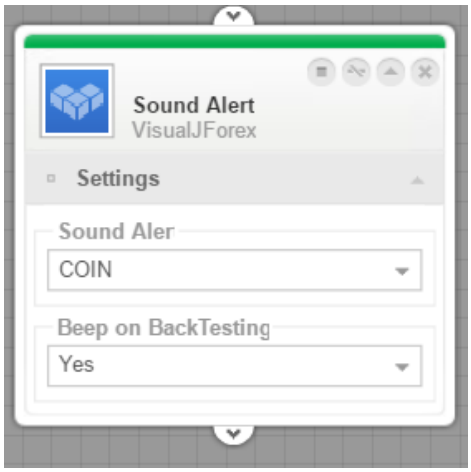
5.3.5. Miscellaneous



Get Time unit: Used to catch the time stamp either form the candle time (candle opening time) or from the tick time available in the left panel. This block works in a way to breakdown the date and time input into separate variables: The hour, the minutes, the seconds, the day of the week and the day of the month. Variables are saved in an integer type. To get the full time format, multiple blocks can be used. (7.10)

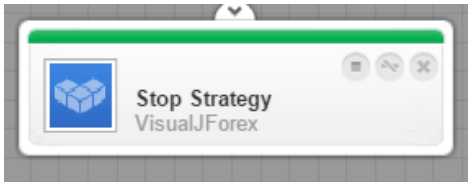
Days of the week reference is defined as 1 for Sunday, 2 for Monday, 3 for Tuesday, 4 for Wednesday, 5 for Thursday, 6 for Friday and 7 for Saturday.

➔ *Date and time information corresponds to the candle or the tick events, it is possible to get the time of a tick price or a candle is opening but not the current time in GMT.*



Sound Alert: To add an audio notification insert this block and choose the alert to be played. By default the setting “Beep on back-testing” is set to No, switch it to Yes in order to hear the alert.

All alerts are supported by JForex platform so when the strategy code is exported to Java and used in JForex, the alert(s) is played as well.

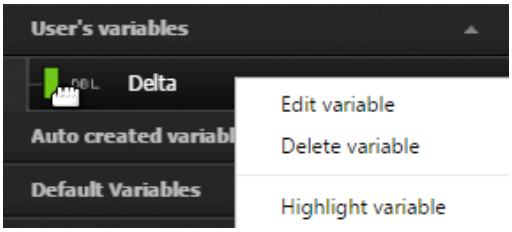


Stop Strategy: A block used to stop the strategy permanently.

6. The variables

The list of variables are available in the left section of the platform and grouped by category

The variables panel

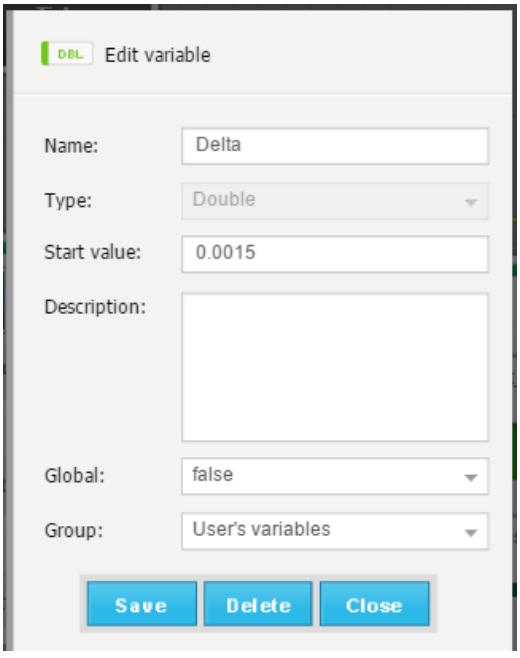


Variable contextual menu and options

→**Edit variable:** To edit the variable name, start value, description, Global setting or Group (see below for more info).

→**Delete variable**: Permanently delete the variable

→**Highlight variable**: This is a way to highlight the selected variable flashing in order to locate it in the workspace. Once activated the identification color of the variable will be flashing in the left section but also in the appropriate block to catch the user's attention.



Variable's attributes can be edited from the above window. The type of the variable cannot be changed; in case the user needs that a new variable should be created.

The "Global" setting allows displaying the variable with the default variables, thus it can quickly be modified from the pop-up window that appears before launching the strategy.

6.1. The Variables panel

Variables section is displayed at the left side of the workspace; it displays the variables that are used to fill the blocks. Variables are sorted out by category as following:

6.1.1. User's variables:

This section is dedicated to custom variables created by the user. Any variable created by the user is automatically saved in this section. Once the variable created, it can be edited or deleted by right-clicking and choose the action from the contextual menu.

6.1.2. Auto created variables:

Some blocks automatically create their outputs in the form of variables and such type will be automatically saved in this section. When using an indicator for instance, the output is created automatically and is often a variable with the type "Double" named randomly. The user can rename this variable which will be then stored in the user's variable section.

6.1.3. Default variables:

Variables considered by default are available in this section:

- Default Instrument:
- Default Traded Amount
- Default Slippage
- Default Stop Loss
- Default Take Profit
- Default Period

6.1.4. Account:

This section is dedicated to variables that belong to the trading account:

- Account ID: The trading account identifier
- Account Currency: The base currency of the trading account
- Equity: the equity available in the account. When the strategy is launched in real time run, it retrieves the real time equity. When back-testing the strategy the equity is defined by the user. By default the equity is set to 50.000 USD
- Leverage: The leverage settings of the account
- Margin cut level: this is the level of use of leverage where the account will get a margin cut. Usually set to 1:200
- Over the Weekend leverage: this is the weekend leverage of the account. Usually set at 1:30 but may be changed upon user request
- Use of leverage: The real time use of leverage of the account
- Global account: this variable tells whether the account is set to "hedged" or "non-hedged" (Global) mode. By default, the account is set to hedged mode thus "non-global"

6.1.5. Position's info:

The positions information is a section dynamically handling the status of the positions created. Such variables when used with a specific type of blocks such as "Position Viewer" can retrieve the status of positions created before the strategy is launched. By status we mean the order/position status as well as the number of positions obtained with the use of "Position amount" variable.

- All positions: This variable is often used with the block “Position viewer” and the help to identify all the positions of the account whatever the status of the positions or order
- Open positions: This variable is often used with the block “Position viewer” and returns detailed data about the open position(s).
- Pending Positions: This variable is often used with the block “Position viewer” and returns information related to pending orders: Stop, Limit, MIT

6.1.6.Trade Event:

Trade Event is handling trading messages received by the platform. It is possible to manually save past data in custom variables; this section is very practical when it comes to working with information related to previous trades and orders. Trade Event variables are created to be used with the start point “Trade Event”. This start point is particular as it is only activated when a trade message is received.

Trade Event variables are sorted in multilevel way with “Last Trade Event” as a first level with the most recent message in first position.

➔ *Previous trading messages cannot be retrieved or used if the strategy was not in running status.*

6.1.7.On Candle:

Variables that belong to current candles are saved into this sub-section. Both BID and ASK sides are considered and each side includes the open, high, low, close prices in addition to the candle period and correspondent volume**,** traded instrument and the candle open and closing time.

➔ *Last candle data corresponds to the current candle still fluctuating. The only known price for an ongoing candle is the **open price**. The close, high and Low prices cannot be determined definitely till the candle is finished.*

6.1.8.On Tick:

Same as the candle data, the tick variables are: The BID and ASK prices, volume traded, instrument and the tick time.

➔ *The tick time is not the GMT time but the timestamp when a price quotes (GMT)*

6.2. The variable types:

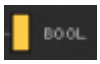
The description of the below variables is made from the user perspective and its use in Visual JForex.

6.2.1. Common variables:

There are five standard variables used in the platform as following:

Double: Used for prices, traded amounts and indicators outputs. The price of currency pair is saved as a Double type; a distance of 5.2 pips must be saved as 0.00052 in a Double variable type.

Integer: Stop Loss and take profit expressed in pips are set as integer as a matter of example. Such variable types do not support decimals. IF the user needs a SL of 5.2 pips for instance, he should convert this into Double type and use the SL field as a price and not in pips in his trading command block (Block settings 4).



Boolean: Logical variable that takes only 2 values (True/False). It is widely used to implement logical triggers defined by the user. Example: A position can be either long or short, thus the variable “position is long” can be either true or false.

String: supports text input. A comment attached to a position is defined as string. Also, the position ID which could be a combination of text and numbers accepts such type.

Date and Time: The date and time variable is displayed in European/American date &time format at the level of the user interface but expressed in Linux epoch format at the level of the back end. The Epoch time format is simply the time expressed in milliseconds: 1 hour and half is equivalent to 90’*60’’*1000= 5400000

6.2.2. Custom variables:

The following variables are used in JForex API which is a library used in programing dedicated to Dukascopy JForex.

➔ *An **array** is a range of data that is meant to describe a collection of elements. Tick, Candle, Position and Signals variables defined below are arrays thus they include a collection of related data variables.*

Tick: it is an array that includes prices of Tick BID/ASK sides, respective traded volume, instrument, date & time of the tick event.

Candle: Candle array contains OHLC prices, candle period, traded instrument, date & time.

Position: The position array saves the information related to a given position which includes the position ID, the traded amount, the close price and time, the creation and fill time, the open price and more. The entire list can be obtained once a “Position” variable type is created. Such variable is mostly used with the block ‘Position viewer” to get information related to previous/current positions.

Instrument: This variable refers to the traded instrument. When creating a new variable with this type the user will be asked to choose the instrument from a drop down list.

State: The status of an order or a position is retrieved by this variable; it is typically available under “Position” array and receives dynamically the current status of the order. The possible values of an order or position status are:
Created: when the order is just submitted and still at the level of the front-end
Opened: When the order reaches the server and was recorded
Filled: The order gets filled thus it becomes an open position
Closed: when a position is closed
Cancelled: when an order is cancelled either by the server or by the user

Period: This variable belongs to the candle array; it uses the candle standard periods which are: 10 seconds, 1 mn, 5mn, 10mn, 15mn, 30mn, 1H, 4H, 1day, 1 week and one month. Any candle period different from the previous list is not supported; custom periods require workarounds.

Command: This variable handles the order types which are: Buy at market, Sell at market, Buy limit, Sell limit, Buy Stop, Sell Stop, Buy limit trigger on the BID, Sell limit trigger on the ASK, Buy Stop trigger on the BID, Sell Stop trigger on the ASK, Place bid and Place offer.

Message: it refers to the messages received by the platform which are classified with the help of the variable “Message Type”. This is an array used in the section of “Trade Event”.

Message Type: Is an array that contains the classes of trading messages received by the platform and also some additional message types that are not yet fully implemented. This variable’s type can be found under Trade Event section.

- The implemented message classes are:

Position Rejected: When an order gets rejected
Position submitted: When an order is received by the server
Position Filled: When a “FILL” message is received by the platform
Position Fill rejected: When an execution is cancelled
Position Close rejected: When a close request is rejected by the server or the market
Position closed: When a position is closed successfully
Position change: When an order is edited successfully
Position change rejected: When an order edit request is rejected by the server

- Currently, the non-supported message classes are:

Position Merged: When a merge of 2 positions (or more) is made successfully
Position Merge rejected: When the merge request is rejected by the server
Mail: handling email requests configured in a strategy for instance, (when an email request is sent)
News: Market News received by the platform
Calendar: Economic calendar event received
Notification: Platform notifications received
Instrument status: Status of the instrument (if changed)
Connection status: Current connection status of the account (if changed)

Offer side: Related to the indicator’s calculation method which takes either ASK or BID side.

Signal: A signal is an array of variables specifically used when the entire strategy is connected as single block (More about “strategy used as a block” in the section How to build a strategy). Such variable triggers trading signals without effectively sending trading commands. It is created for the specific use of signal trading or any similar way of trading. It also simplifies visual JForex when it comes to building complex ideas.
The signal array contains the “Signal Type” and the information related to the position in question represented in the variable “Signal Position” which lists the same variable types as “Position”.

Signals: In most cases this variable is created when the strategy is used as a block to save its trading signals. This array is embedded and the detailed breakdown is obtained with the help of the “Loop Viewer” Block. The output is then listed in a “Signal” detailed variable (described above)

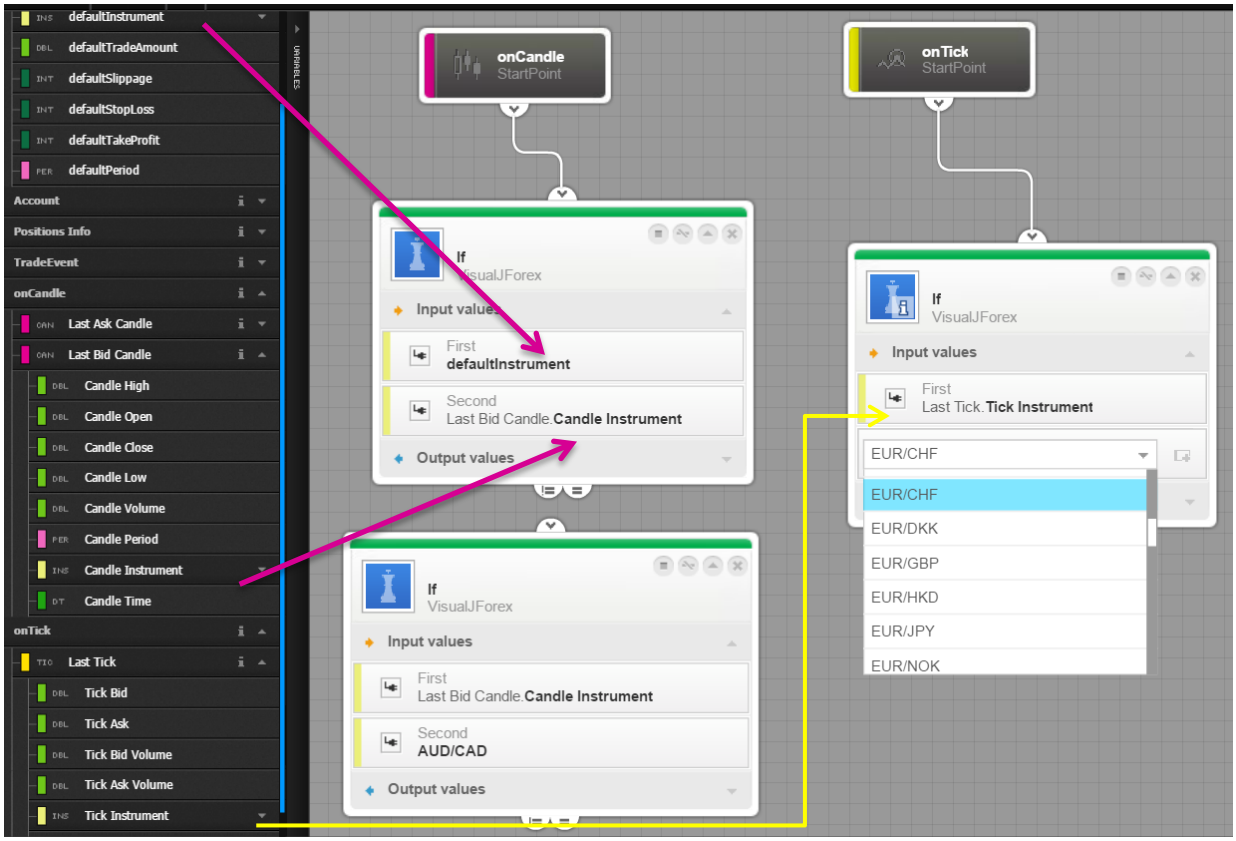
Signal Type: Part of the Signal array, the Signal Type refers to the trading commands created by a given strategy. The Signal types are: Position Buy, Sell, Merge, Close and order change.

Moving average type: This is a specific variable related to MA types; it takes a given value as listed in its drop down list (12 Moving average types are listed).

Applied price: Same as the previous variable, the Applied Price is also used in association to a given indicator and refers to the price type used in the calculation formula of the indicator (Close price, open price, Median price, weighted close price …). This variable helps changing the indicator calculation method dynamically.

7. How to

7.1. How to do a strategy subscription



7.1.1.Instrument subscription:

By default Visual JForex Candle method is subscribed to EURUSD but it is recommended to subscribe to a specific instrument especially if the strategy is planned to be run in JForex platform. If the strategy is not subscribed it will trade randomly on any instruments when used in JForex. The technical reason behind this is related to the way “On Candle” and “On Tick” methods are used in the API and the default subscription to all instruments.

Subscribing to an instrument consists then of filtering out the non-used instruments and keeping only the needed one. In the above screenshot, the first top left block is filtering the instrument using the “default instrument” variable and the last Bid Candle instrument (or Ask Candle). The filtration is then made by linking the IF condition from the “Equal” exit. The second IF condition underneath is also a way to filter out the unused instruments but without the “Default Instrument” variable, the candle instrument is simply dropped as a first variable and when clicking on the second input field, a drop-down list will be displayed. This method is used in some particular cases where a strategy is implementing more than a single instrument.

When the strategy starts from the On Tick method, the same method is applied but instead of Candle instrument variable the user should use Tick instrument as displayed in the top right block.

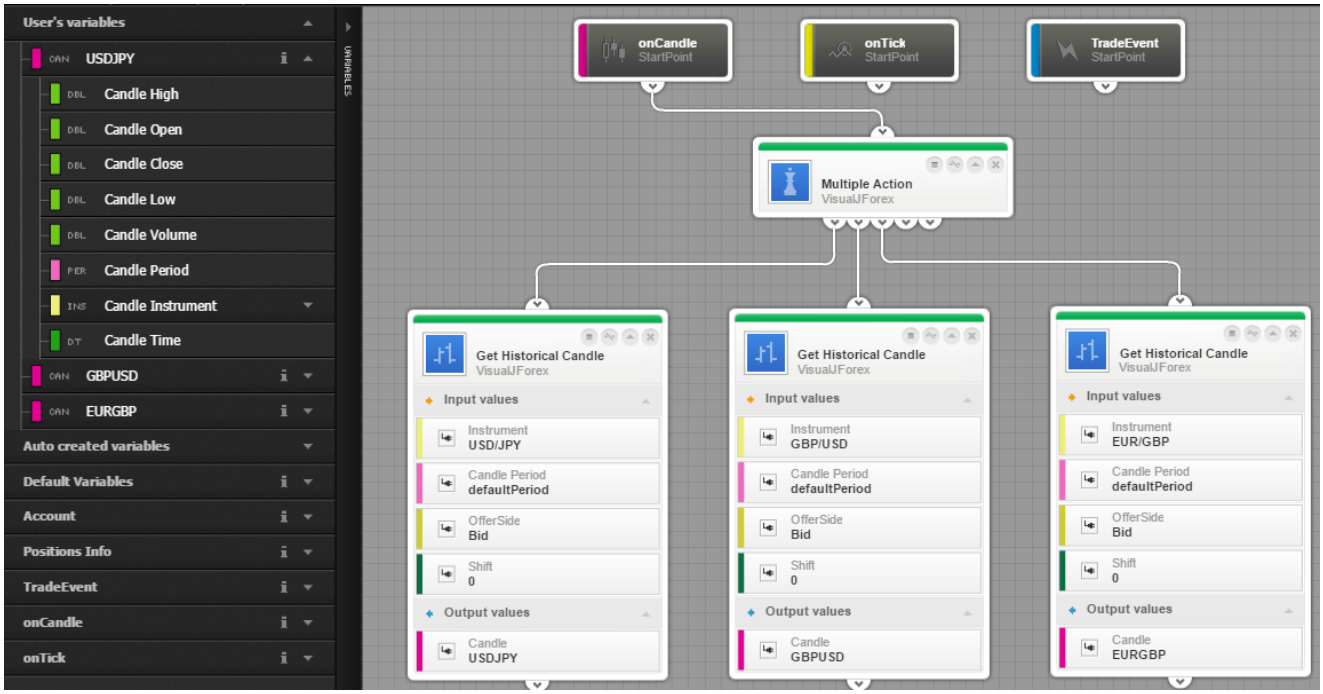
7.1.2. Period subscription:

The period subscription is similar to the instrument subscription described above but using “Last Candle period” and “default period variables” to filter out the non-needed candle periods. The reason is that the “On Candle” method automatically retrieves all the trading periods and therefore is activated every 10 seconds as it is the shortest period available by default. Custom period subscription requires workaround.

7.2. How to use multiple instruments

Developing a strategy with multiple instruments is possible but testing it is more complicated as it requires exporting the strategy to JForex platform in order to run the test simultaneously on several instruments. Technically, building such strategy is made with the help of “Get historical candle” block that calls the previous candle data for each instrument.

Use as many “Get Historical Candle” blocks as needed for the instruments and delete the default variable then chose the needed instrument from the drop-down list. Instead of using the auto created outputs, a dedicated one is used in the following example; this will ease the identification of variables per instrument afterwards.



It is possible to add a period filter before the above blocks.

In the screenshot at the left, the variables were dropped on the workspace to debug their values while the strategy is launched. As shown, the OHLC prices are displayed with regards to each instrument and can be used for further conditions afterwards.

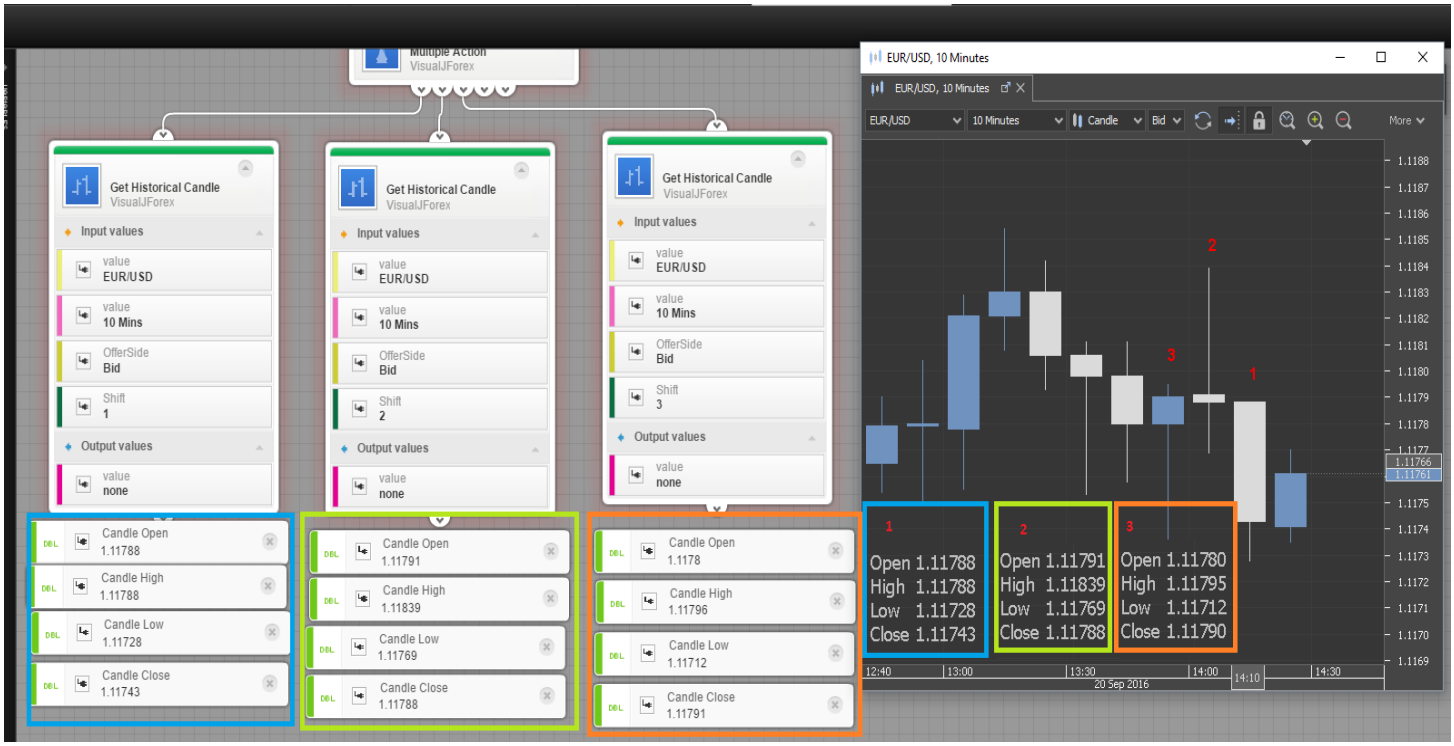
The previous way of retrieving candle data is frequently used when the strategy idea implements cross-instrument logic; in other words when condition A is verified on EURUSD then trade USDJPY for example. So to trade a given instrument, the entry condition must be verified on another instrument. For strategies that implement the same logic applied to several instruments, it is easier to create an additional strategy file and change the instrument at the level of Visual JForex. Alternatively run or test the strategy in JForex and select the needed instruments.

Testing multi-instrument strategies cannot be done in Visual JForex as it implements a single default instrument. Such strategy is tested only in JForex.

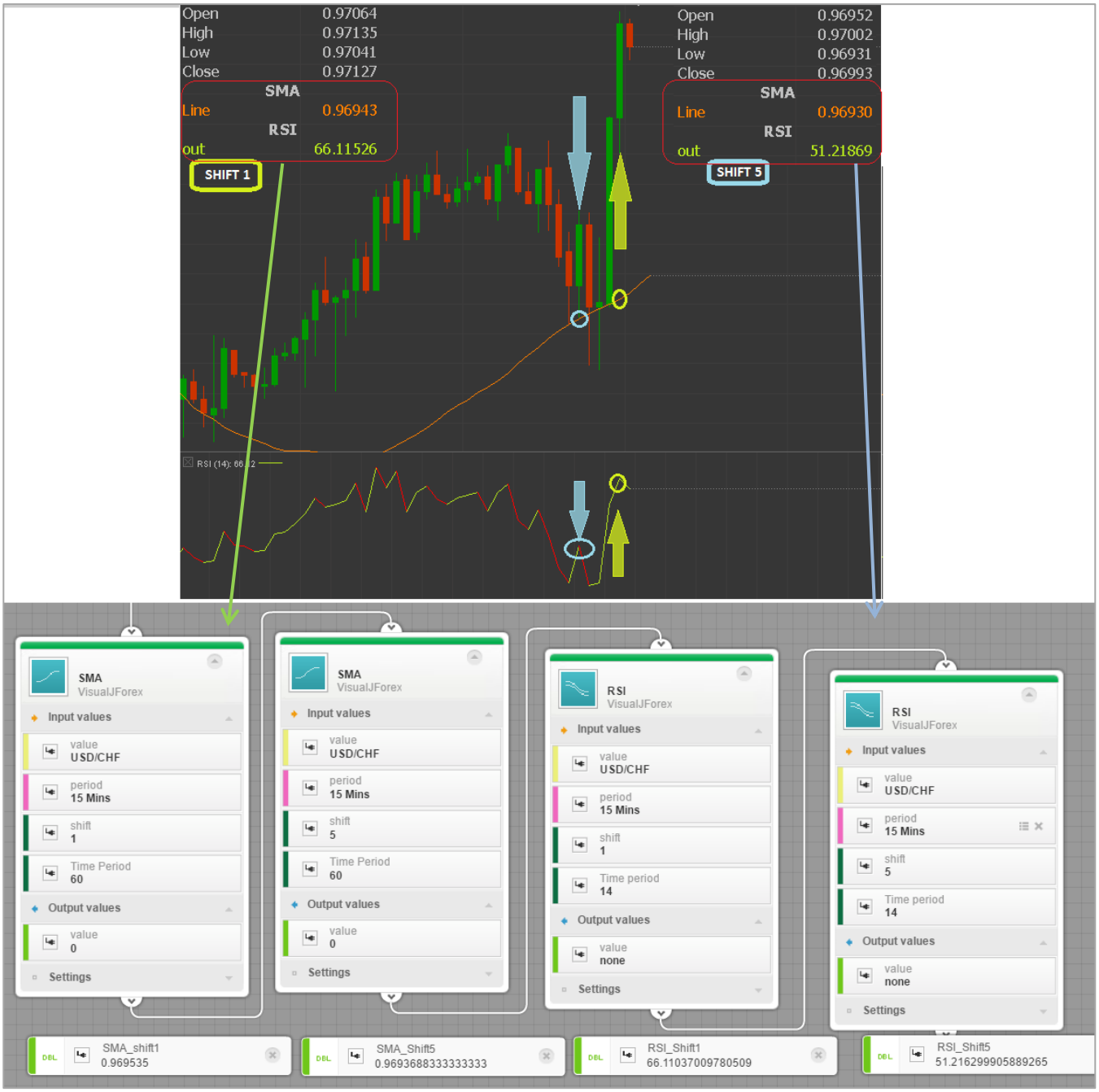
7.3. How to work with shift values

Shift parameter is available for almost all indicators as well as in “Get Historical Candle(s)” blocks. This parameter is particularly useful when it comes to calling previous data that belongs to specific candle(s) in the past.

Shift 0 always stands for the current candle still fluctuating therefore the only known data is the open price, the remaining information will be definitely assigned when the candle is formed and expired and thus this candle becomes the previous candle alongside with a new one that will start. Shift 1 is then used to call the information related to this previous candle, shift 2 is related to two candles ago and so forth.



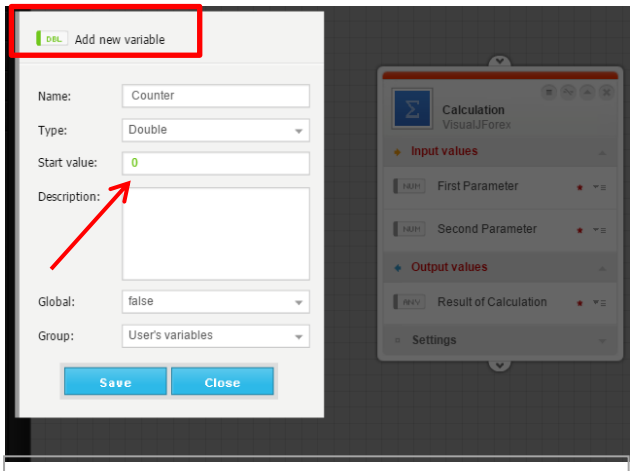
Shift values are frequently used in indicators especially when the user requires detecting the trend of a linear indicator; such pattern is identified when comparing previous value of an indicator with a recent one. The comparison can be used to check if the trend is bullish, bearish or even flat for example.



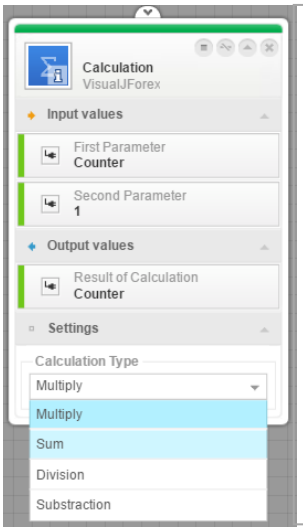
In the above screenshot SMA and RSI indicators are used with shift 1 and shift 5, this requires 4 blocks in total; when the strategy is launched in real-time run the output variables get their real-time values and it matches JForex indicators outputs. Sometimes a difference is noticed between Visual JForex outputs and JForex outputs is explained by the rounding applied in JForex indicator's outputs

7.4. How to implement a counter

A counter is a simple tool that helps in the implementation of an instruction sequence. The basic logic of the counter is to increment a variable while a situation A is happening till a situation B occurs then reinitialize the counter back 0. In most of the cases it is implemented using the calculation block as following:



Create a new variable (Double or integer type) and set it to 0. In this example, the new variable name is "Counter"

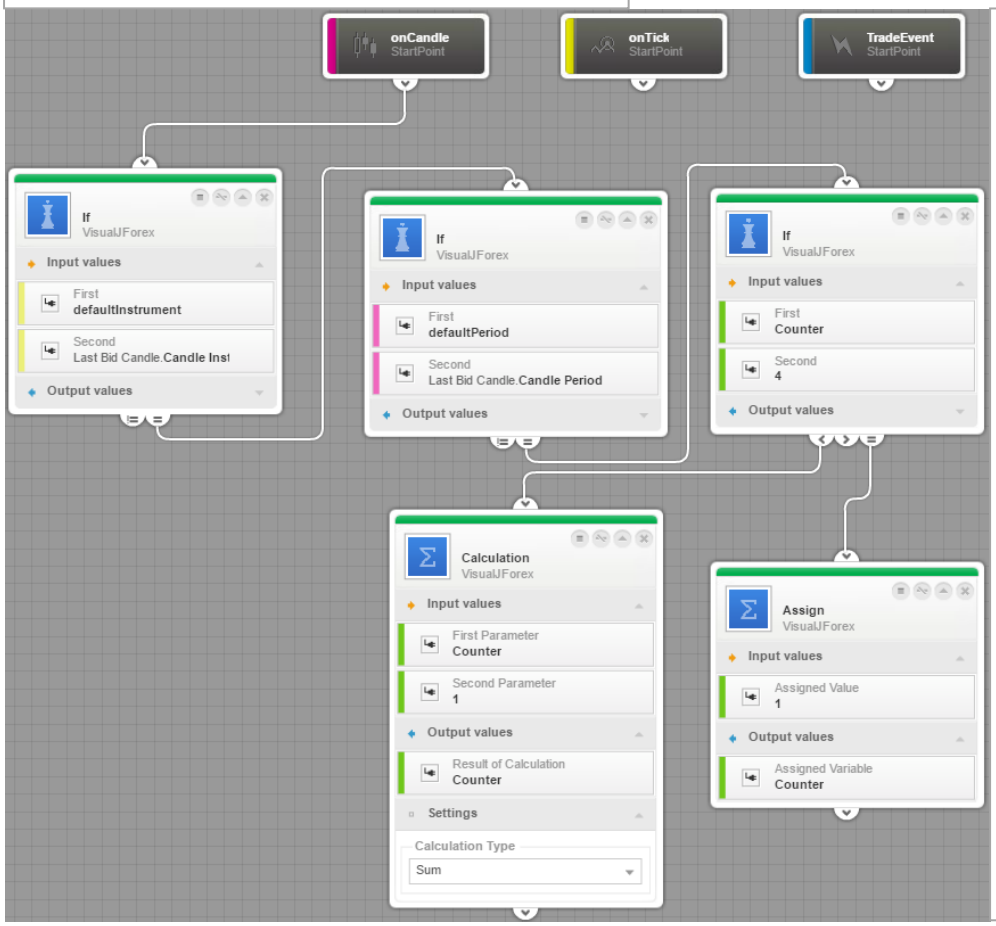


Drop the new variable "Counter" in a calculation block and create another variable and set it to 1.

Use the same "Counter" variable in the result field (output) and choose "Sum" as operation.

➔ The counter is ready to be used

The way this block will be connected will define the logic of the counter.



Example 1:

The idea: Say you need to build candles with a period of 40 seconds:

1/ Define the trading period to be equal to 10 seconds.

2/ Build the counter as detailed above.

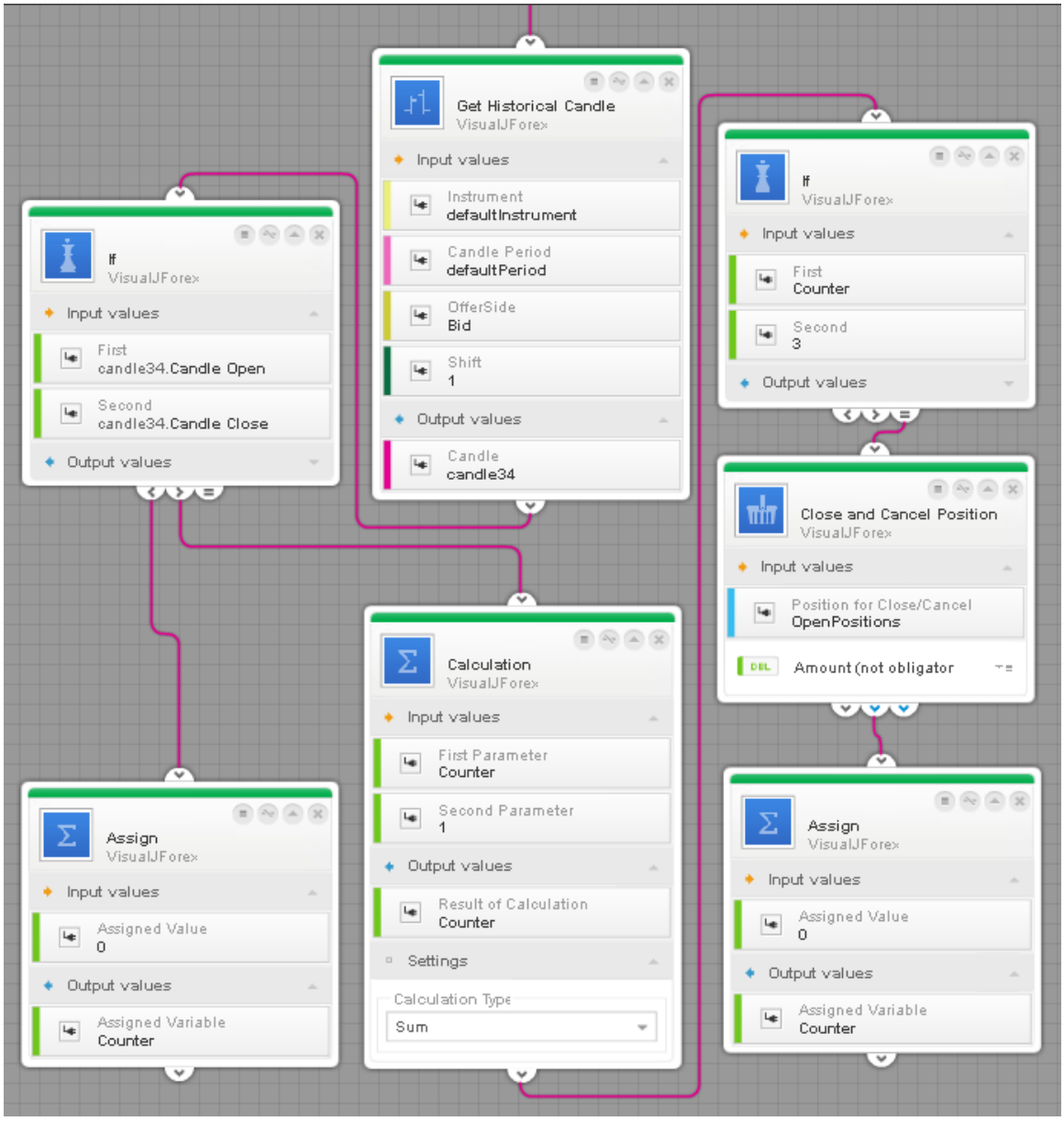
3/ 40 seconds is 4 times 10 sec, therefore the counter should be set to increment till 4 then reset to 1 afterwards.

The IF condition is **essential** because it checks the value of the counter whether it has reached the level 4 or not. Once reached the assign block will set it to 1 so that the iteration restarts.

Example 2:

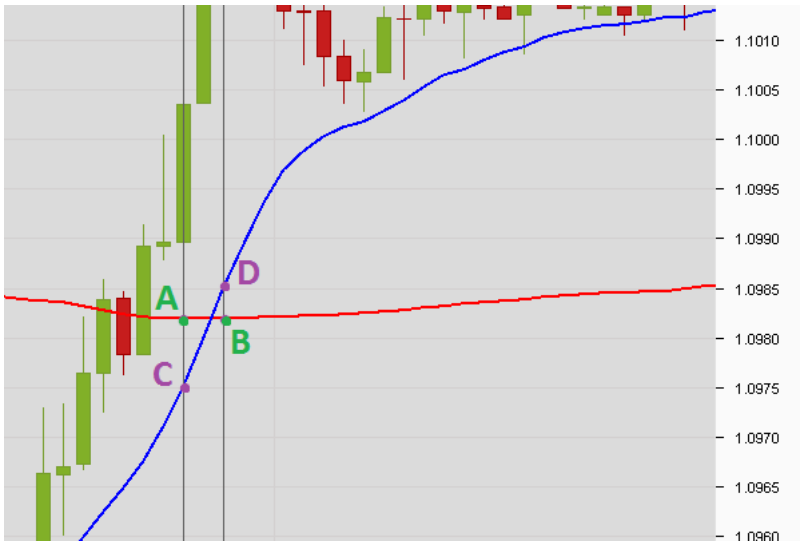
Close a long position if 3 consecutive red candles are made starting from the moment the position is opened.

1- Check the status of the position which should be in “Filled” status
2- Get the previous candle data and check whether it is red or green
3- Implement a counter to make sure that there are three red candles in a row.



The above blocks are handling the strategy part that is in charge of identifying 3 red candles in a row. The “Get Historical Candle” block is plugged after the IF condition checking if the position in question is opened and having the status “Filled”. Once the previous candle is retrieved and identified as red (Open>Close) the counter increments till reaching the value 3 therefore this is a forward count after the position is filled. The frequency of the counter calculation is made according the candle period filtration plugged at the beginning of the strategy. The counter increment only if the previous candle is red, in case it is green the counter is reset to 0 because the condition is breached. After reaching the value 3 it is then confirmed that the previous 3 candles are bearish therefore the position is closed and the counter is reset back to 0.

7.5. How to build a cross between 2 indicators

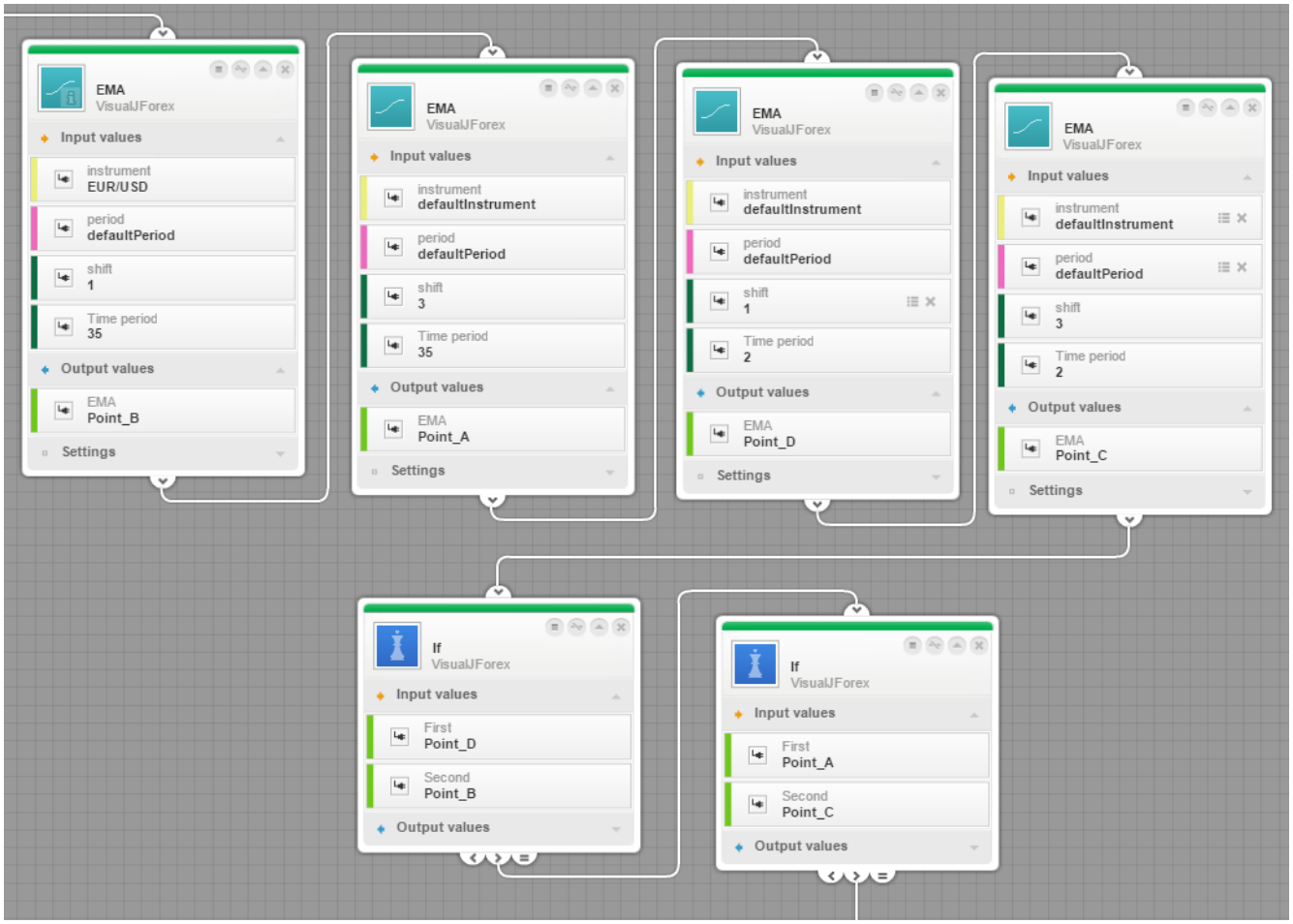


An intersection between two lines is defined with the help of four points, in this example A, B, C and D. The idea is to be able to call these points and verify their values to conclude whether a cross happened or not. This is made with the help of "Shift" values in the indicator block.

A cross is defined by: A>C and D>B → this is valid for the case where the fast indicator (blue) crossed the slow indicator (Red) in an upward movement. TO determine the trend of each line an additional verification can be added: A<B & C<D for an ascending cross or A>B & C<D for a dead cross, etc.

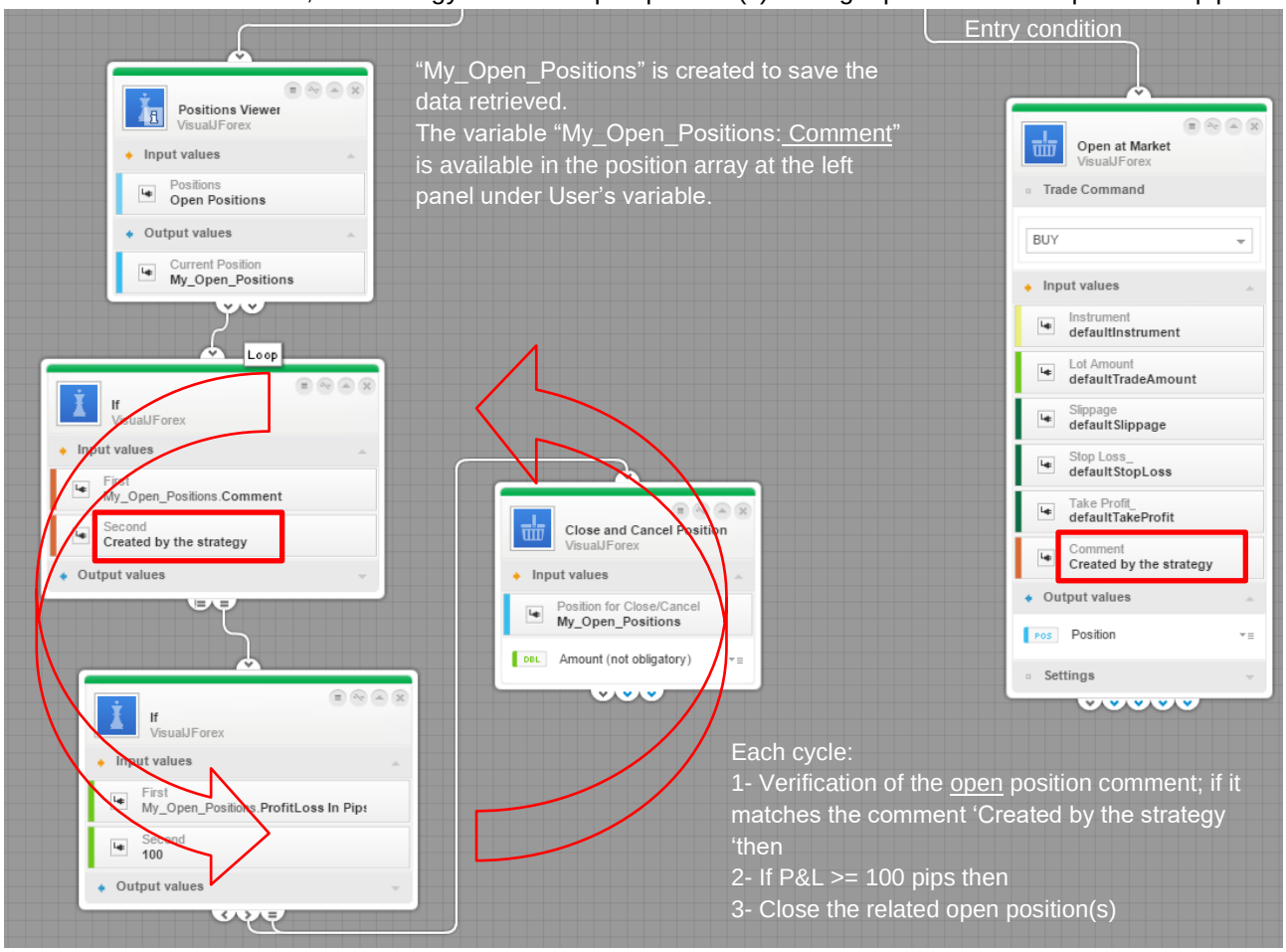
The first step is to add the indicator with the appropriate shift values, it is important to mention that the detection of this pattern depends on how shift values are chosen: Shift 1 Vs Shift 2 is more sensitive than Shift1 Vs Shift 3: there is more chances to catch the cross event when the values checked have longer distance in-between.

The comparison is made with an IF block as following:



7.6. How to identify and work with positions

Positions created by strategy or manually can be identified using “Position Viewer”; this block works as a sensor of any filled order using a loop function. Once the cycle in place it is easy to retrieve it is easy to pull out open positions per side, placed orders, current open P&L, etc.
The following example is aimed to manage open positions and close those created by the strategy with a profit of 100 pips.
The position viewer block can be linked to a start point directly. If there is conditions in-between, the user should make sure that this (these) condition(s) will not stop the position Viewer block form functioning. The cycle frequency is made according to the trading period. The Position Viewer checks if any open position is created by the strategy with the help of the “comment” variable that must be used in every trading command as shown below (Comment: “Created by the strategy”). If the comment matches, the strategy will close open position(s) having a profit more or equal to 100pips.



Moreover the user can work with positions using another technique: To create a dedicated position variable that is associated to each block “Open at market” as an output; the verification is made separately position by position. This method is used for basic strategies however it is more efficient to work with position viewer block.

The logic of “Position Viewer” is exactly the same as the “Loop Viewer” block it terms of method and the way a cycle is performed and managed. The only difference is that Loop viewer supports more types of arrays as input.

7.7. How to build tick bars (with a counter)

Tick bars are a type of chart based on a predetermined number of ticks that build a candle. This is easily made at the level of JForex platform via preferences menu but it another method is applied when programming this in Visual JForex. As the only built-in methods are candles (10sec to 1 month) ticks candles are built with the help of a counter that increments at each received price and stops when the target tick number is reached. We should then add another set of conditions to use the Open / High / Low and Close prices retrieved.

The workaround is made as following:

Step 2 and 3 in next pages

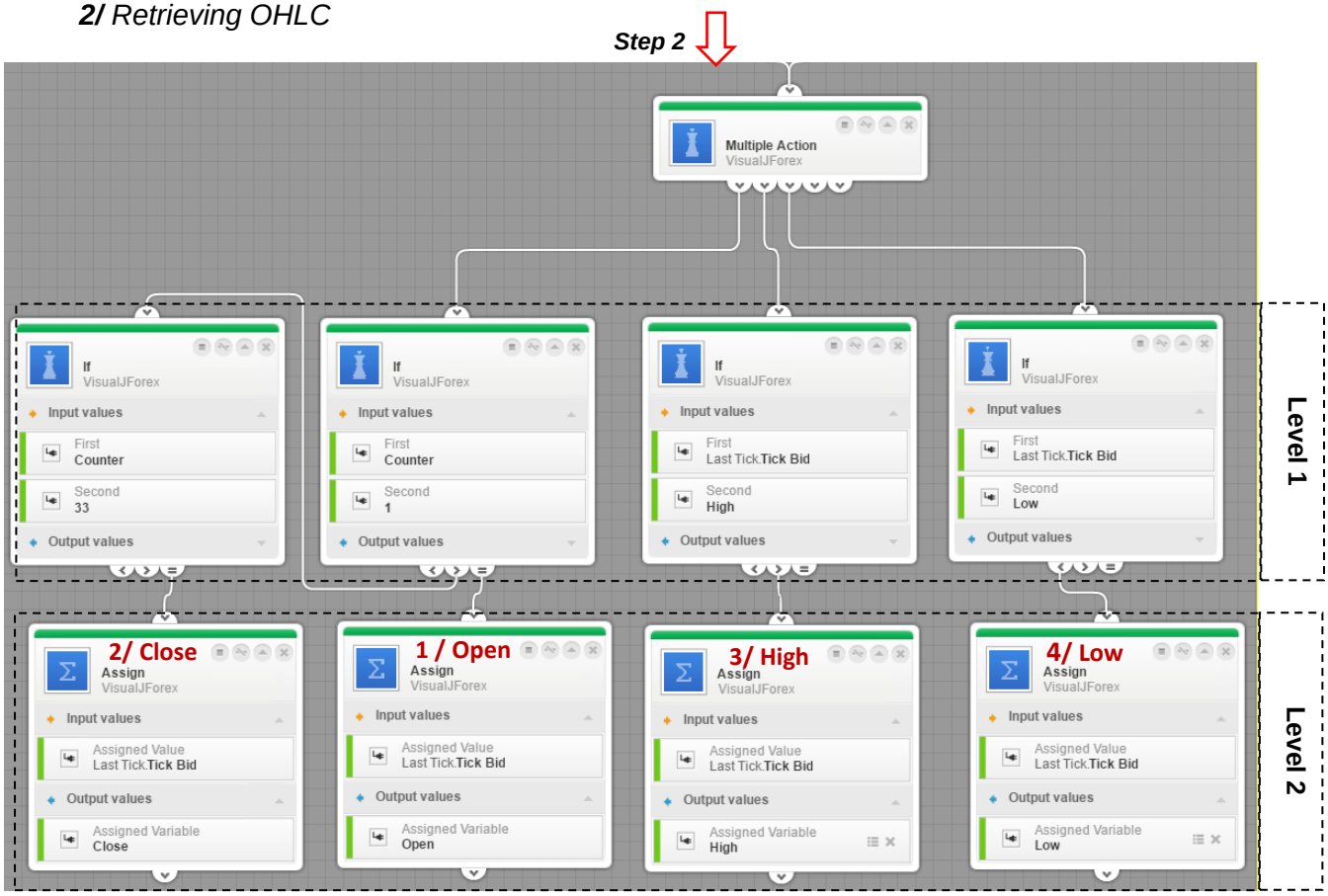
***1/** implementing and setting up the counter:*

In this example tick bars are based on 33 ticks.

A calculation block is used to set up the counter; note that the same variable “Counter” is used as a first input and as a result.

The “IF” condition is very important as it distributes the flow according to the counter value. As the counter’s value increases to 33 the strategy continues to step 2, once it is above step 3 is executed.

***2/** Retrieving OHLC*



Step 2 is handling the way the Open, High, Low and Close prices are registered. Four variables are used to save the data accordingly.

- Level 1: The new variables Open, High, Low and Close have no values assigned yet, they are used in every IF block to determine which value they should take:

- Level 2: Once the appropriate condition is met, each of the OHLC variables gets a price as following:

  If the counter is equal to 1 → Open price is determined

  If the counter is equal to 33 → Close price is determined

  If the current tick is greater than the "High" variable → Assign this tick price to the "High"

  If the current tick price is less than the "Low" variable → Assign this tick price to the "Low"

  ➔ High and Low variable are updated along with every new tick received until the sequence of 33 ticks is reached.

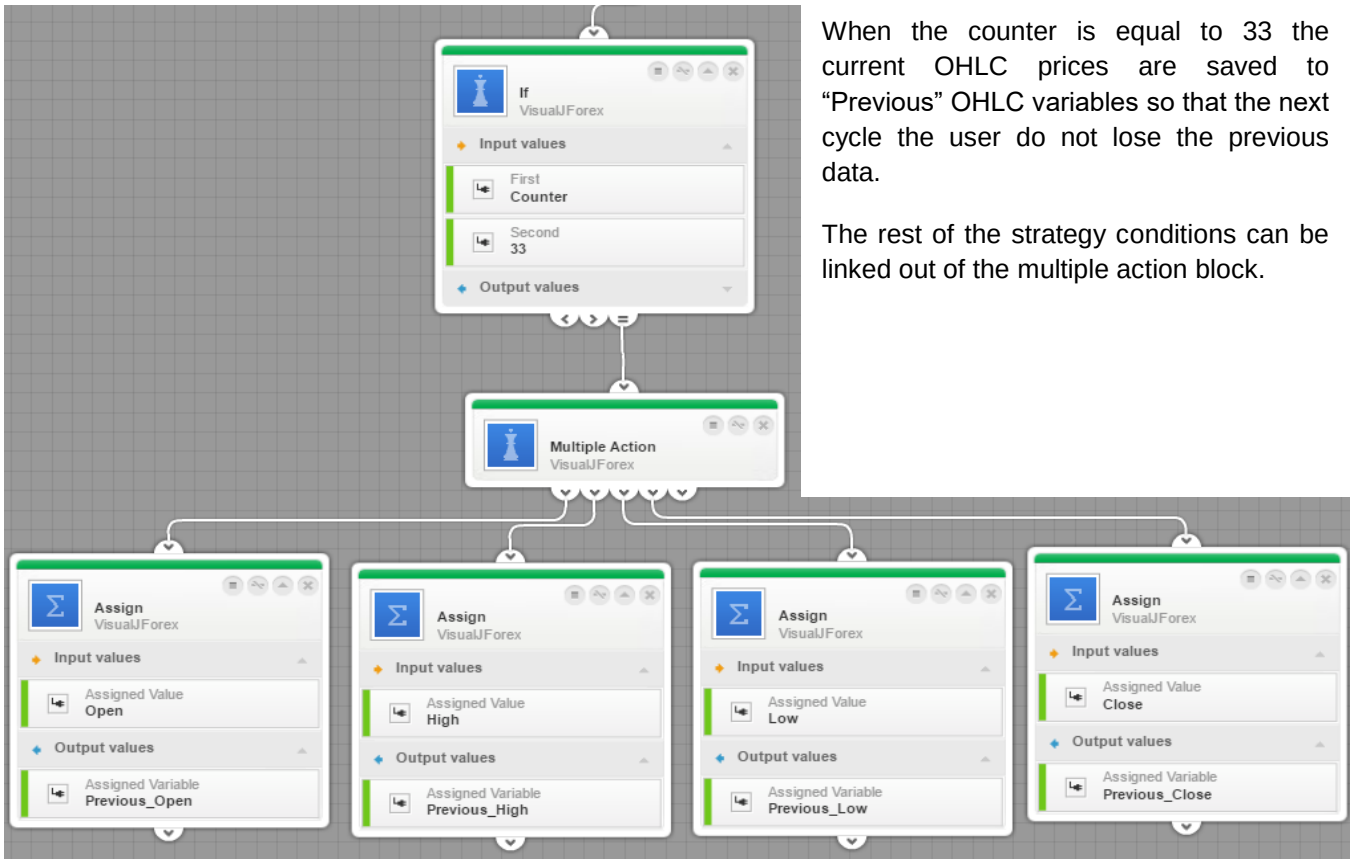
In this example once the first sequence of 33 ticks is completed, the strategy will over-write the previous OHLC but the user can save the previous OHLC by creating new variables.

***3/** End of the cycle and Reset of the OHLC variables*

Step 3

At the 33rd tick the Open, High and Close prices are reset to 0 while the Low gets a value of 2000 because of the condition [Last tick < Low] : If the Low price was reset to 0 the [Lasttick<Low] is a condition that doesn't make sense thus it will be impossible to determine the Low price

***4/** Using OHLC in the strategy*

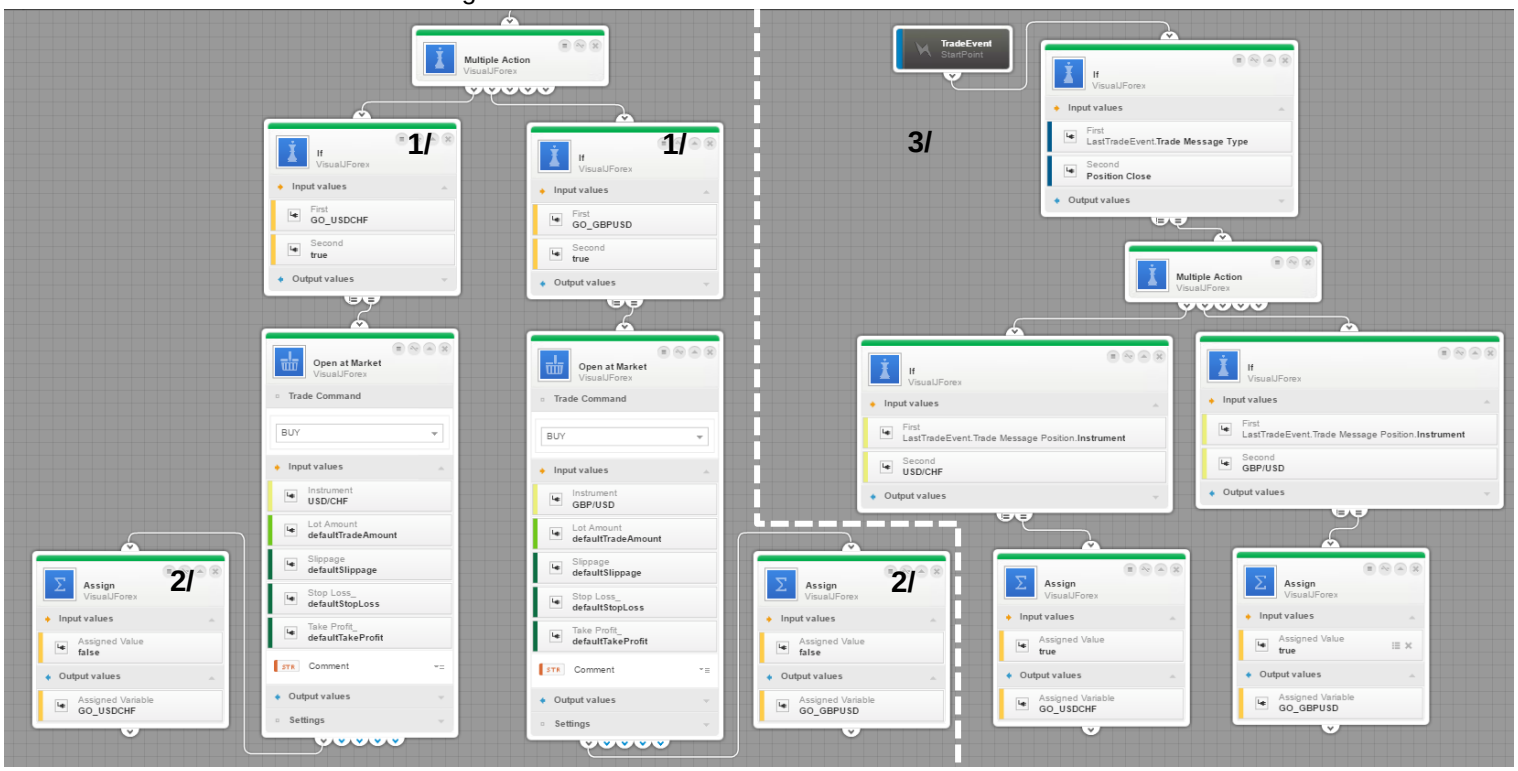


When the counter is equal to 33 the current OHLC prices are saved to "Previous" OHLC variables so that the next cycle the user do not lose the previous data.

The rest of the strategy conditions can be linked out of the multiple action block.

7.8. How to work with logical triggers

A logical trigger is a variable created by the user that takes a given value when a condition A occurs and switches to another value when condition B happens. Let's assume the following scenario:
A strategy trades 2 instruments and the user needs to manage the number of open positions per instrument: One way to achieve this is to create 2 variables that triggers/ block the position opening command.
1/ *the start value of such variable should let the strategy open a position*
2/ *once the position is opened the variable changes its value immediately to block the next process of position opening*
3/ *once the position is close, the same variable should reset to the initial value.*
This workflow implements a change of the value according to **2 conditions**, thus we can choose a Boolean variable that gets either True or False value.



On the left: The variables "GO_USDCHF" and "GO_GBPUSD" have the value 'True" at start; at the very first flow the first IF conditions will be passed and one position per instrument will be opened after which the variables "GO_USDCHF" and "GO_GBPUSD" get switched to "False". When the strategy executes the next flow the IF conditions (**1/**) will block further position opening. The result: The maximum number of open positions for each currency pair will not pass 1.

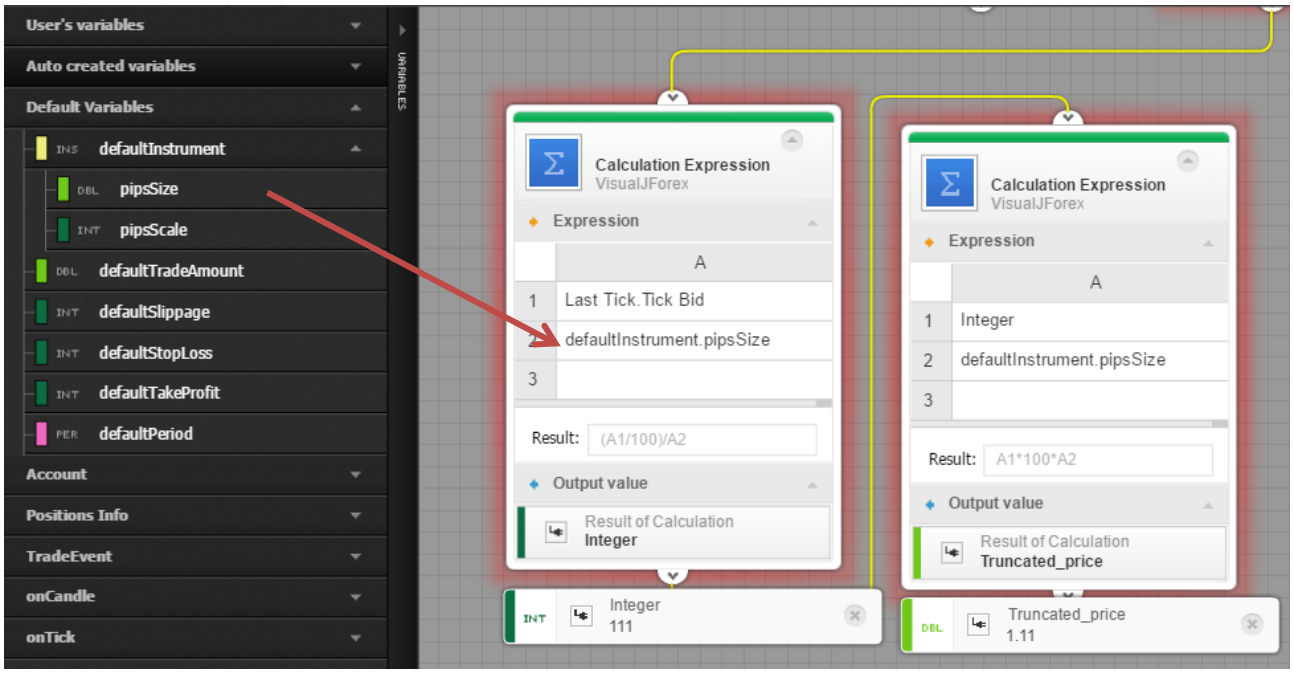
The right side: In this example 'Trade event' is used as starting point upon receiving the message "Position Closed" the strategy will check the currency pair and assign the value "True" to the trigger so that a new position can be opened the next flow.

It is practical to implement a logical trigger when the requirement is based on Start / Hold logic. The user should accurately define when the trigger switches and when it is reset to the initial value. More complex processes with 3 or 4 sequences can be implemented with a variable having the type "Double" or "Integer".

7.9. How to truncate numbers

Truncating number is the action of cutting number decimals in order to comply with the trading platform pricing requirement or to simplify a calculation result. Numbers truncation is not rounding.

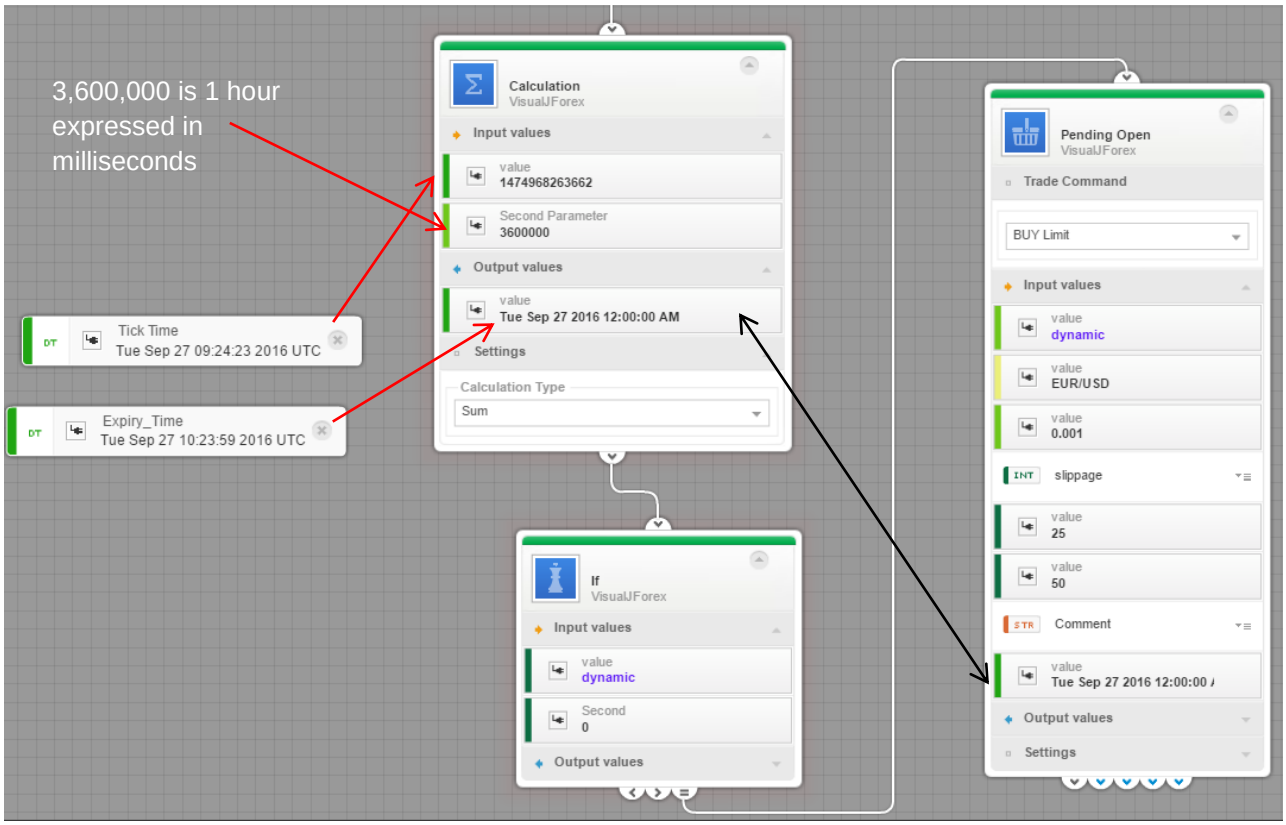
Visual JForex doesn't implement such feature in a dedicated block so the following way can be used.



In this example the last tick price will be truncated to 2 decimals; the first step is to get the tick price multiplied by 100 and saved in a new variable that is defined as "integer". The choice of integer type is made on purpose in order to eliminate decimals. The second step is to divide the result by 100 so that the result gets 2 decimals. "Pip size" variable is used so that this solution can be applied for any instrument (Example: EUR/USD=0.0001 / USD/JPY= 0.01 / stocks and indices =0.01)

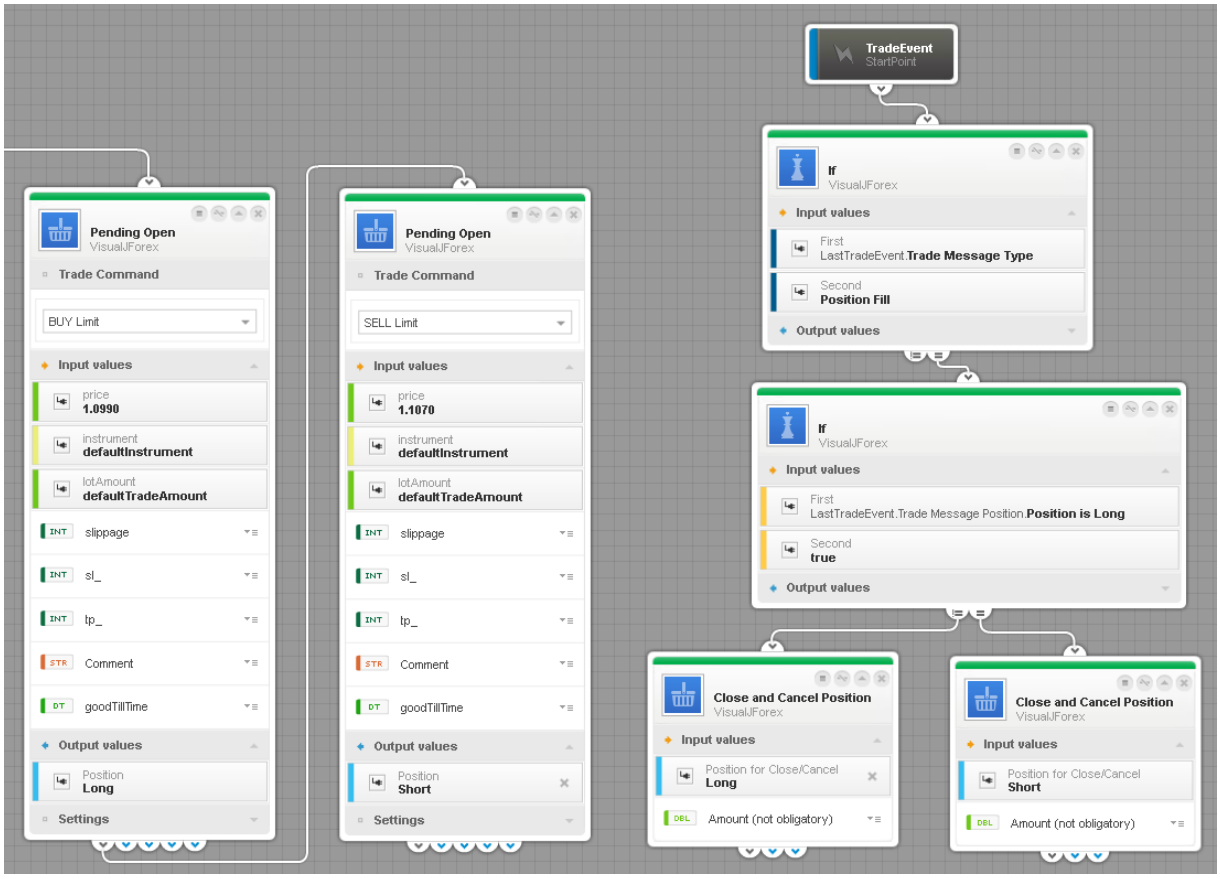
7.10. How to use date and time

The following example shows how to add an expiry time to a Limit order just before it is placed. Once the entry conditions are met the calculation block will capture the tick time and determine the order expiry time just before the order is placed. Tick time is displayed in epoch format inside the calculation block while the strategy is running, once the variable is placed on the workspace it is converted to regular Date and Time format.



7.11. How to cancel placed order(s) if one is filled

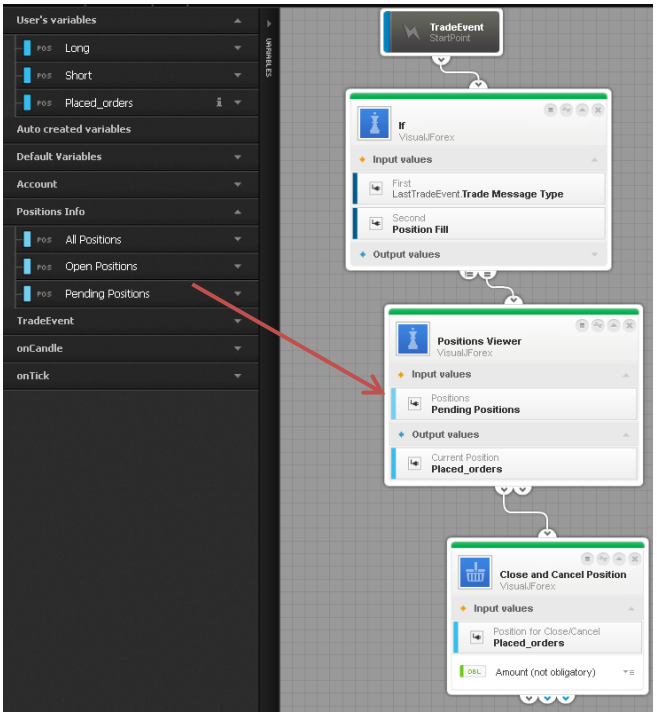
When the strategy works with pending orders and the user needs to cancel any placed order once one get filled then one of the best solutions is to work with trade event start point in order to catch the “FILL” message and issue a cancel request for the remaining orders still in pending status.
When the strategy is handling limited number of placed orders you can identify the order in question by creating a new output then if one of the placed orders is triggered the strategy proceed to the cancellation of the other one as following:



From “Trade Event” start point: When the trade message type “Order filled” is received the strategy verifies the position side and proceed to cancelling the opposite order. In this example IF the last trade is “Long” then we close the SELL order and vice versa.

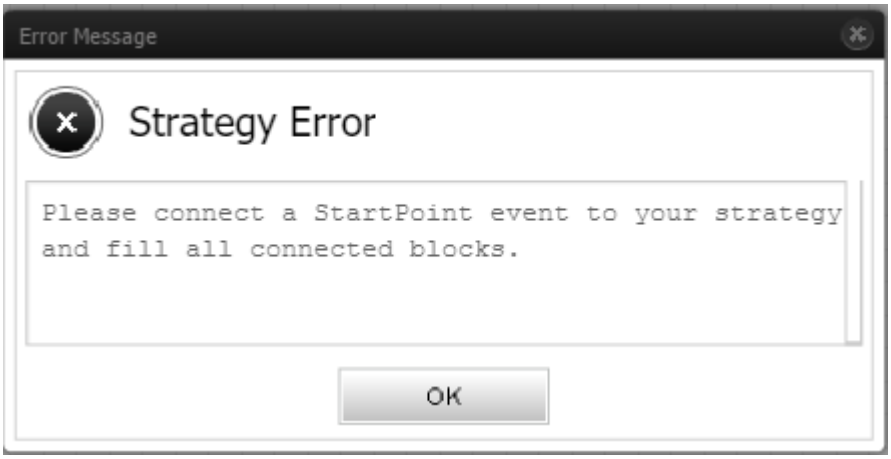
When the strategy is handling several placed orders then it would be much more efficient to work with “Position Viewer “block to be able to send a cancel request to all orders having the status “Placed” instantly.

Note that the Position Viewer block was used with “Pending Positions” as input.



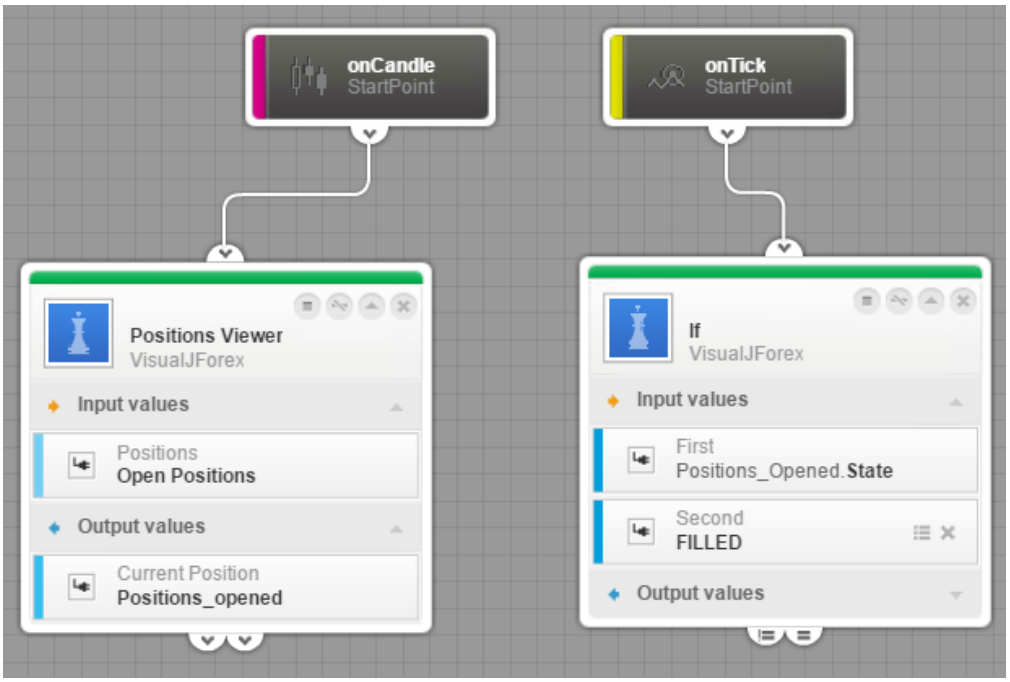
8. Error messages & troubleshooting

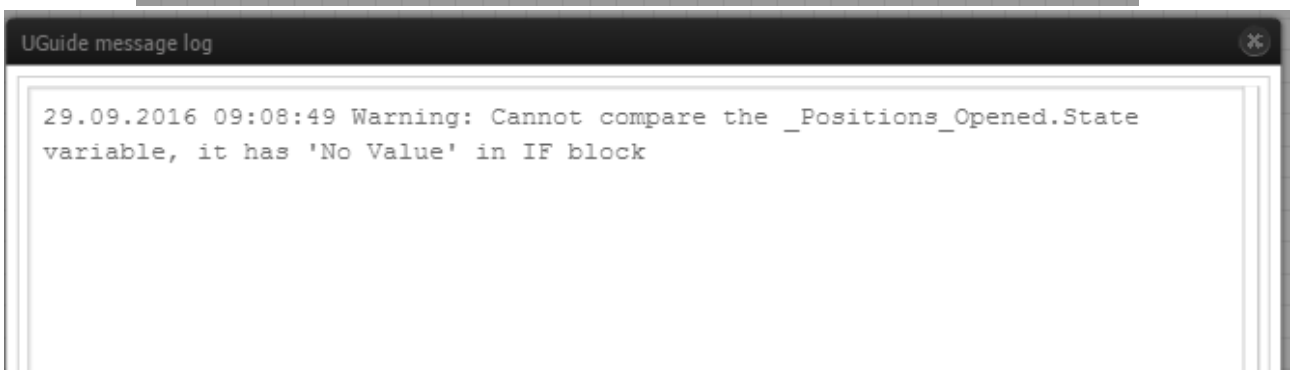
8.1. Connection issue or incomplete block:



This error occur when the user attempts to run a strategy while there is a missing parameter in a block or when there is no established connection to a start point. It also happens when clicking on "Run" where there is no strategy file opened yet.

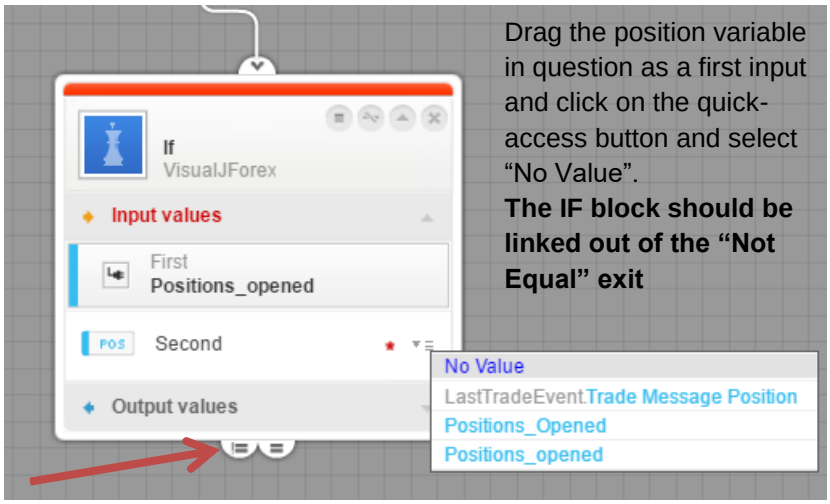
8.2. "No Value" errors





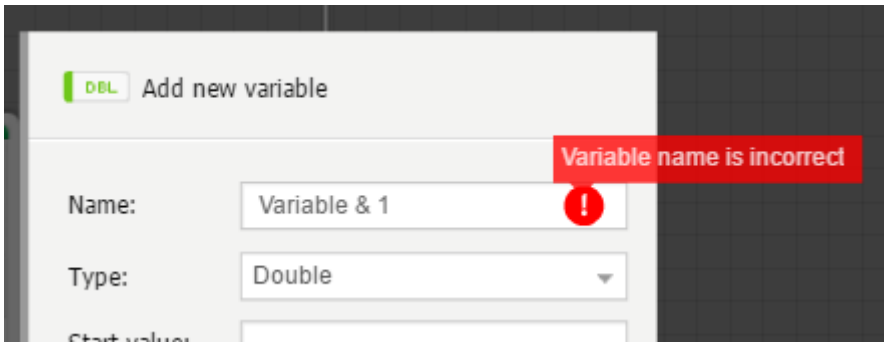
This type of error happens frequently when a given variable is used while it doesn't get a value yet. In the above error message the "Position_Opened.State" variable refers to the current status of the open positions whether if filled or not. The IF condition is trying to check the status of positions not yet created, thus it returns this type of error. To avoid such situation the user shall add one more verification level to make sure the "Opened positions" variable is updated before checking its status.

This is made as following:



Drag the position variable in question as a first input and click on the quick-access button and select "No Value".
The IF block should be linked out of the "Not Equal" exit

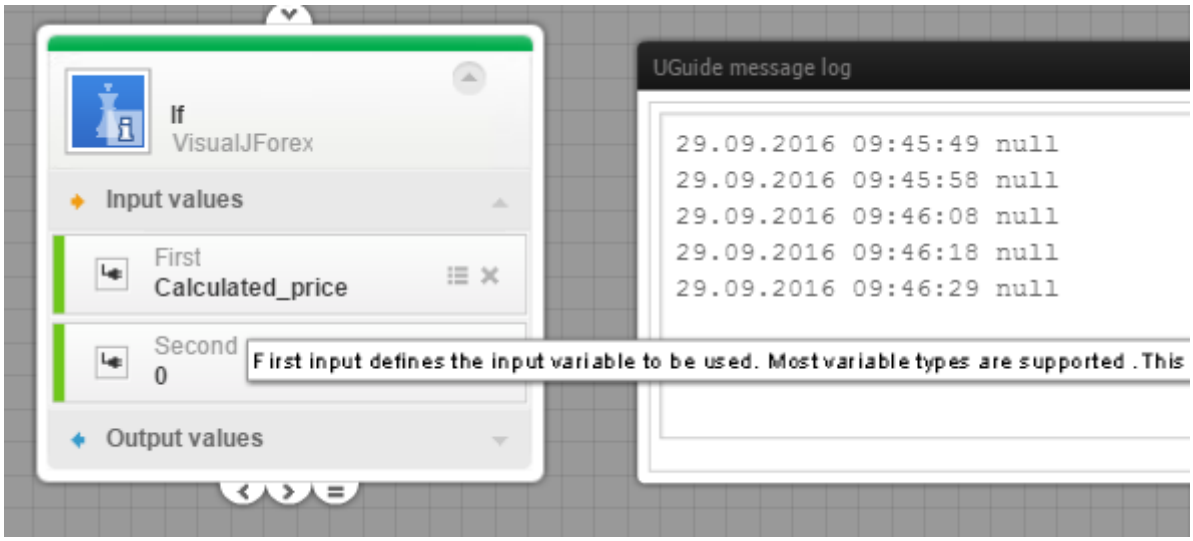
8.3. Variable naming:

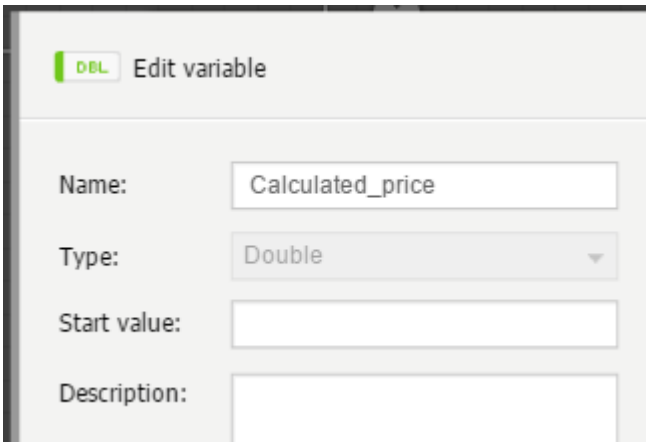


This error is related to the variable naming format which doesn't support spaces, symbols and shouldn't start by a digit.

8.4. Variable missing a "start value"

As detailed in section 6 related to variables creation and setup the following error is typically caused by a missing attribute when the variable was created: The start value.

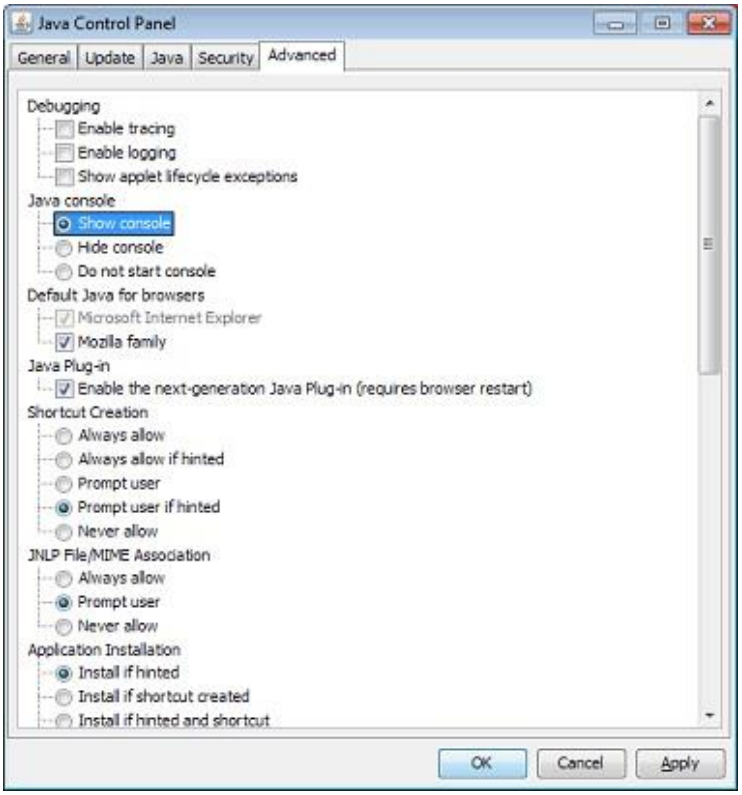




The start value is the initial value that is assigned to variable upon creation. the Variable “Calculated_price” was created with an undefined start value, the system will consider this as having “No_Value” thus any logical or mathematical operation becomes impossible.
In some cases (similar to the errors described in 8.2) a dynamic variable can generate such error while it doesn’t receive a value yet, but once updated automatically (position status for instance) the error won’t appear anymore.

8.5. Advanced debugging

he user can activate Java console in order to display Java error messages that help identifying error reasons. This option is activated from Java controle panel



.

When Visual Forex processor is running, a log window will appear and log any message related to the process with regards to the trace level set.

Errors with the header “java.lang.NullPointerException” may impact the strategy functioning and should be investigated thoroughly.

Further support and information is available in Visual JForex forum.