

Visual Trader Studio - Connect for MetaTrader 4.0

User Manual

Contents

Overview	5
What Is VTS?	5
What is MetaTrader?	5
Why Use VTS?	5
VTS System Requirements	6
Installing VTS	6
VTS User Interface	7
Main Menu	8
Property Window	9
Output Window	10
Toolbox.....	11
Toolbox NEW Pane	12
Toolbox FUNCTIONS Pane.....	13
Toolbox VARIABLES Pane	14
Toolbox LOGICS Pane.....	15
Welcome Screen	16
Drawing Pad	17
Everything You Need To Know About VTS	18
Expert Advisor Facts	18
VTS Elements	19
START Element.....	20
VARIABLE Element	20
LOGIC Element.....	24
END Element.....	27
FUNCTION Element.....	29
Custom Indicators	32
System Managers	46
INPUT Manager	47
TRADETIME Manager.....	48

TRADESIGNAL Manager	49
COMMUNICATION Manager	50
OPENTRADE Manager	51
Dialogs	54
Choose	54
Note	54
Message	55
Favorites	56
Power	56
Logic Power Tab	57
Function Power Tab	59
Element Name	61
Shift	61
Options	62
Order Selection	70
Special Functions	70
fnOpenOrder	71
fnModifyOrder	73
fnCloseOrder	74
fnGetOrderInfo	76
fnTrailingStop	77
init and deinit	78
Getting Started with VTS	79
Experts Menu	82
MQL Help	83
The MetaEditor	85
HowTos	87
VTS Plug-ins	95
Function and Logic Power Plug-in	95
Candlestick Library Plug-in	100
Easy Email Plug-in	109
FX TrendLine Plug-in	121

FX PowerGrid Plug-in	139
Script & Multi-Platform Plug-in	150
Enable the <i>Profit Exits</i> Plug-in	155
Profit Manager Plug-in.....	171
Client-Side Stops Plug-in	180
Fibonacci Trader Plug-in	194
Signal Aggregator Plug-in	215
MQL-Mentor Plug-in	228
Partial-Close Plug-in	236
EA-Indicator Plug-in	243
Chart Objects Plug-in	251
Market Score Plug-in	283
Order History Plug-in	294
EA Secure Plug-in	305
Chart Execute Plug-in.....	314
Money Manager Plug-in	327
Package Plug-in.....	337
Pivot Points Plug-in	344
File IO Plug-in.....	356
Glossary.....	369

Overview

- [What is VTS?](#)
- [Why use VTS?](#)
- [System Requirements](#)
- [Installing VTS](#)

What Is VTS?

Visual Trading Studio ([VTS](#)) is an application that allows you to create a MetaTrader **Expert Advisor** by dragging and dropping graphical [Elements](#) onto a [drawing pad](#).

Using VTS is similar to using applications like Microsoft Paint or Visio.

VTS offers a [Toolbox](#) of [elements](#) which you can drag, drop, configure and connect on a [drawing pad](#).

When you are satisfied with the drawing, you press a button and VTS builds an Expert Advisor. The Expert Advisor's MQL file is automatically generated by VTS. Now you can simply open your MetaTrader platform, attach the Expert Advisor to a currency chart, and you are running your trading strategy.

What is MetaTrader?

MetaTrader is a trading platform offered by many FOREX brokers. The trading platform is very comprehensive, offering just about every feature you would expect from a FOREX trading platform. Honestly, most FOREX trading platforms offer the same functionality and MetaTrader is generally the same or better when compared with other FOREX trading platforms.

What makes MetaTrader special is that it offers the trader the ability to create automatic (or mechanical) trading programs called **Expert Advisors**.

MetaTrader created its own programming language called [MQL](#) that is used to create these "Expert Advisors". The [MQL](#) language is very powerful and allows a trader to execute many strategies that simply could not be executed manually.

But best of all MetaTrader is free. So it is a great opportunity for a trader to evaluate systematic (or mechanical or algorithmic) trading strategies with little or no investment.

As previously mentioned, MetaTrader is offered by many brokers. Just run an internet search on "MetaTrader broker" and you'll see dozens.

Why Use VTS?

- [VTS](#) eliminates the need to learn the syntax of the [MQL](#) programming language.
- A VTS drawing is a visual representation of how a trading system works.
- The same drawing you create to build your trading system can be used to illustrate your strategy. This allows you to easily share ideas and collaborate on new strategies.
- VTS allows you to add your entire strategy, or parts of your strategy, to your own Toolbox. This allows you to reuse elements by simply dragging them on to your drawing.
- VTS allows you to incorporate professionally-developed, off-the-shelf trading strategies into your own custom trading system.

The bottom line is VTS allows you to transform a trading idea into a working system fast. And at the same time, the illustrative nature of VTS allows you to visually see your ideas and share them with others (both programmers and non-programmers.)

Is VTS Easy to Use?

It's pretty easy, but it does require some learning. You'll need to learn the four basic elements used to build all Expert Advisors as well as some general rules about how the elements can be linked together. Most traders can complete the tutorials in just a few hours and become reasonably proficient in less than a day.

"Reasonably proficient" means you are able to build a typical trading system without much trouble.

VTS simplifies the tedious programming required of the MQL language. However, at the same time VTS allows access to the full functionality and power of MQL.

Since MQL is a very powerful language, expect to continue learning new techniques for a long time.

VTS System Requirements

There are no special system requirements for VTS. If you are running the MetaTrader platform on your PC, then VTS should work fine.

In general, the system requirements are:

- Microsoft Windows operating system – Windows 98 or later
- An Internet connection
- 1 Gigabyte of memory (RAM)
- The MetaTrader platform, Version 4
- At least 5 Megabytes of hard drive memory available

Installing VTS

- **Install MetaTrader**

If you do not already have MetaTrader installed on your computer, you'll need to install it if you want VTS to build Expert Advisors.

If you only want to use the drawing features of VTS to flesh out a strategy you do not need to install MetaTrader.

- Follow the instructions provided by the broker whose MetaTrader platform you have downloaded.
- Take note of the folder in which you install MetaTrader.
- The default location for Alpari is C:\Program Files\ MetaTrader - Alpari UK.
- You'll need to provide the MetaTrader installation folder to VTS. (This is where VTS will place the Expert Advisors that it builds.)

- **Install VTS**

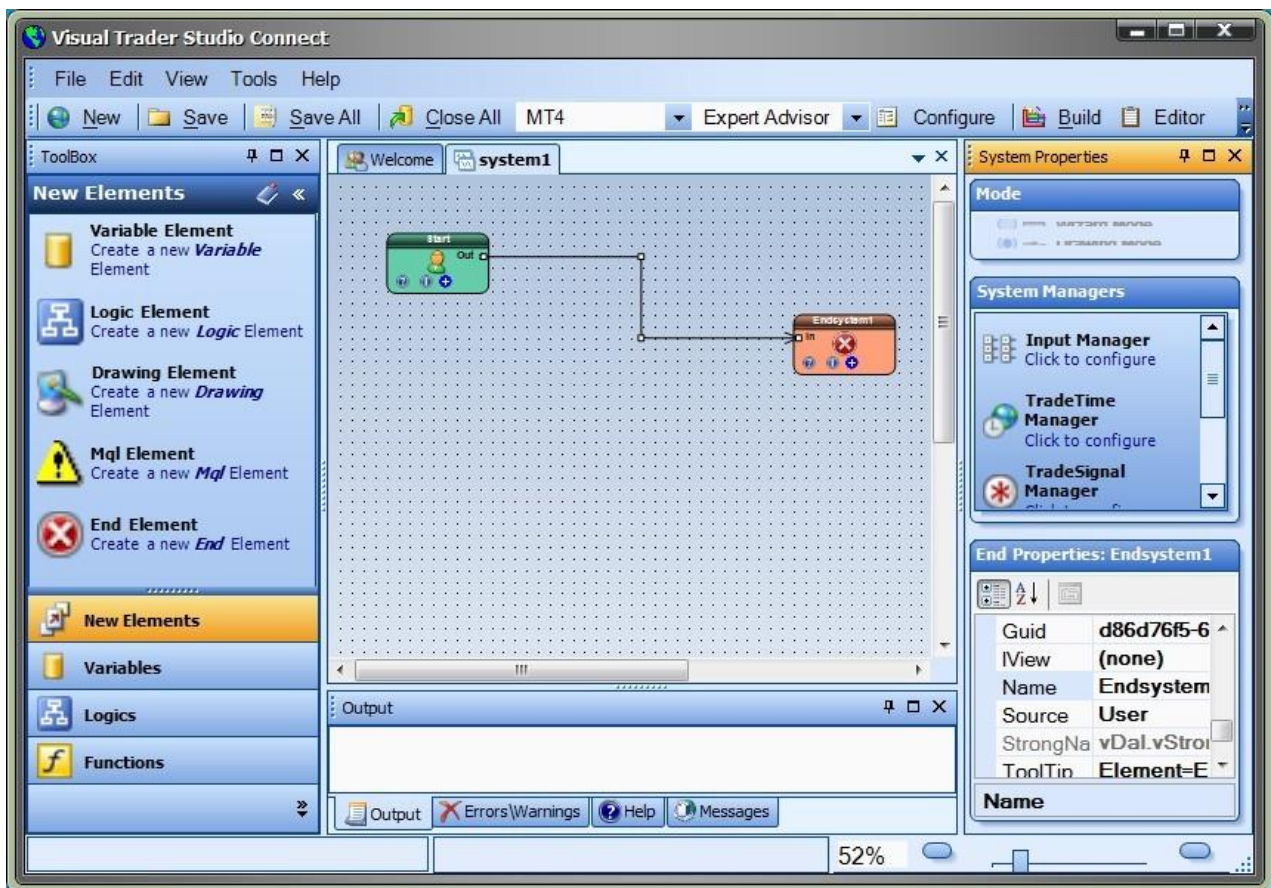
- Double click the setup program and VTS will start the installation process.
- Just follow the prompts – VTS installs just like any other windows program.

VTS User Interface

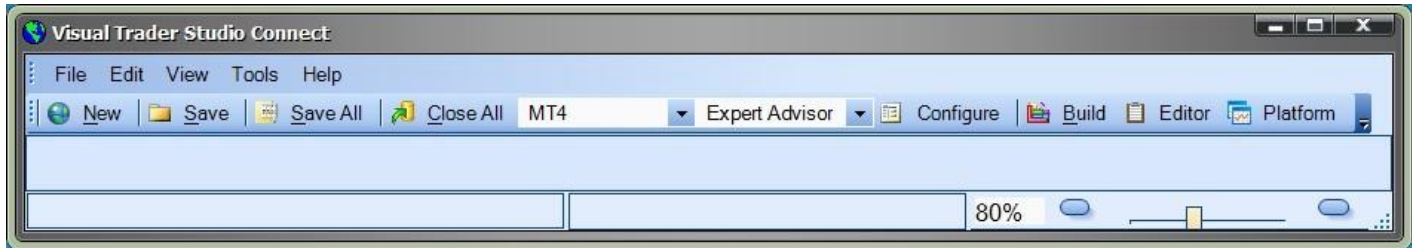
The **Visual Trader Studio** User Interface (UI) was designed according to commonly accepted UI practices. In other words, the UI should work similar to other Windows applications that you use.

The VTS main application consists of five areas:

- The [Main Menu](#) across the top.
- The [Toolbox](#) on the left.
- The [Output](#) window along the bottom.
- The [Properties](#) window on the right.
- The [Drawing Pad](#) in the center.



Main Menu



The main menu area offers the following options:

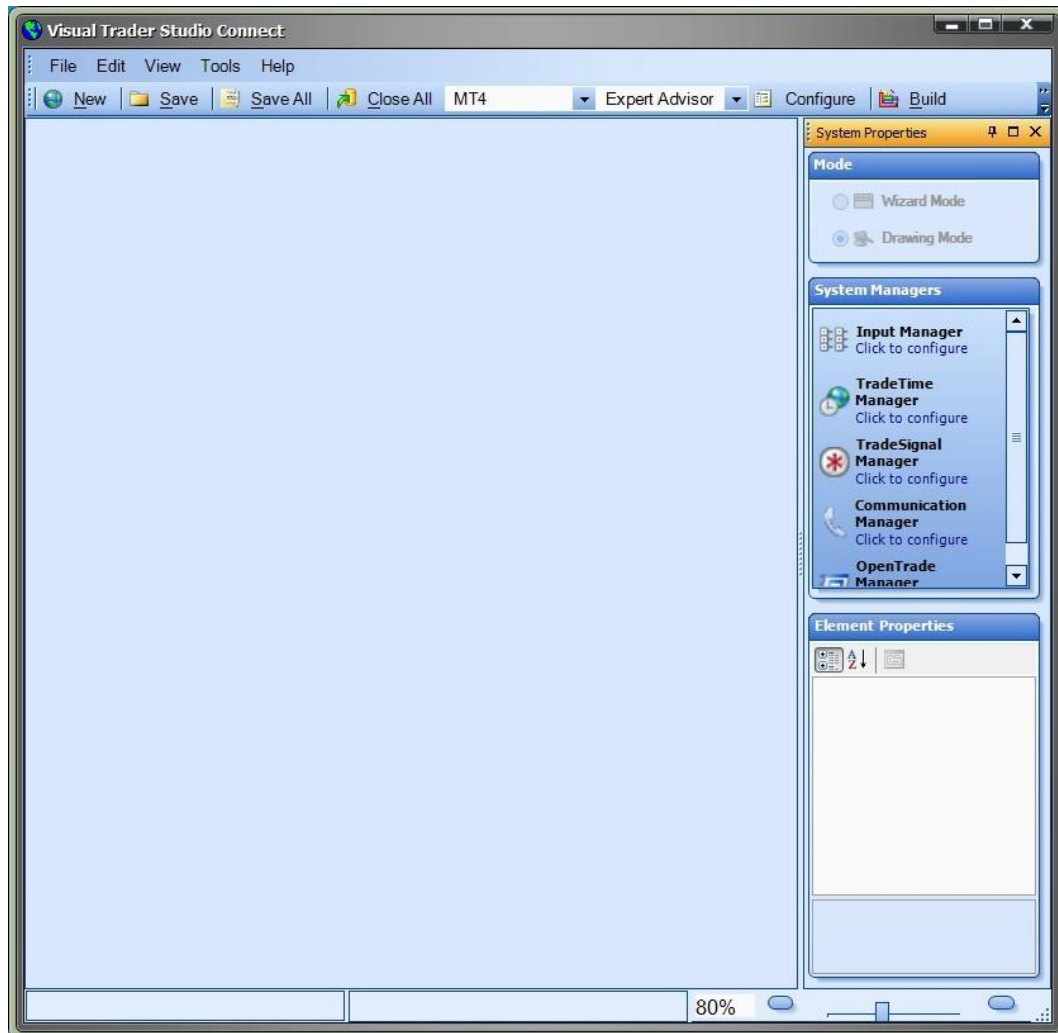
- New - Starts a new [VTS](#) system.
- Save – Saves the currently selected drawing.
- Save All - Saves all open windows.
- Close All – Closes all Drawings .
- Configure - Configure VTS [Options](#).
- Build – Builds the MQL code of the current system.
- Editor – Opens the MetaTrader text editor called [MetaEditor](#).
- Platform – Opens the MetaTrader trading platform ([Terminal](#)) .

Property Window

The **Property** window is located on the far right side of the application window.

There are three areas within the **Property** window:

- The **Mode** area shows the current mode of VTS: *Drawing* mode or *Wizard* mode
- The **System Managers** area is used to access System Managers. System Managers provide unique functionality to the entire EA (system) that can not be easily provided by dragging and dropping Elements.
- The **Element Properties** area is used to display information about the *selected* Element.

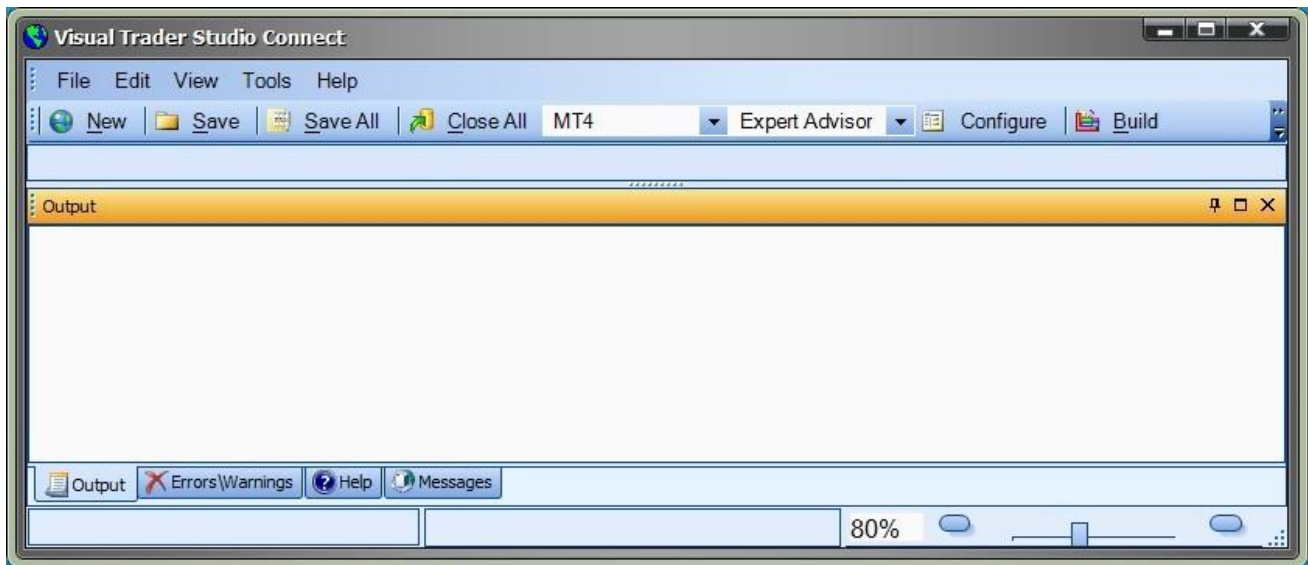


Output Window

The **Output** window is located at the bottom of the application window. The **Output** window is used to display run-time information about the state of the system.

There are four tabs on the **Output** window:

- Output
- ErrorsWarnings
- Help
- Messages



Toolbox



The VTS **Toolbox** is found on the left side of the application window.

The **Toolbox** is used to store and add Elements to the drawing pad.

There are four tabs on the **Toolbox**:

- New Elements
- Variables
- Logics
- Functions

Toolbox NEW Pane



The NEW Toolbox pane is used to drag new [Elements](#) onto the [Drawing Pad](#).

To drag a new [Variable](#), [Logic](#), [MQL](#), or [End](#) Element onto the pad, select it with your mouse and drag it an

This will create an [Element](#) at the approximate location of the mouse pointer.

When you drag a new Drawing Element onto the pad, you are prompted to open a new [Drawing Pad](#).

Toolbox FUNCTIONS Pane



The Functions pane of the [Toolbox](#) is used to drag [Function Elements](#) onto the [Drawing Pad](#).

There are a number of menus within the Function Pane:

Favorites:

This menu contains Functions that have been set as a Favorite. To set a function as a Favorite, select the function on the menu, and right-click and select Favorite.

Indicators:

This menu contains Technical Indicators. The standard indicators are stored under the **MQL** submenu. Custom Indicators are stored under the **Custom Indicators** submenu.

Trade: This menu contains Trade functions. The standard trade functions offered by MetaTrader are found under the **MQL** submenu. Super-convenient functions, provided by iExpertAdvisor, are found directly under the Trade menu.

Bar: This menu contains functions related to Bar or Candle values. The standard Bar functions offered by MetaTrader are found under the **MQL** submenu. Super-convenient functions, provided by iExpertAdvisor, are found directly under the Bar menu.

Account: This menu contains functions for getting Account information. The standard Account functions offered by MetaTrader are found directly under the Account menu.

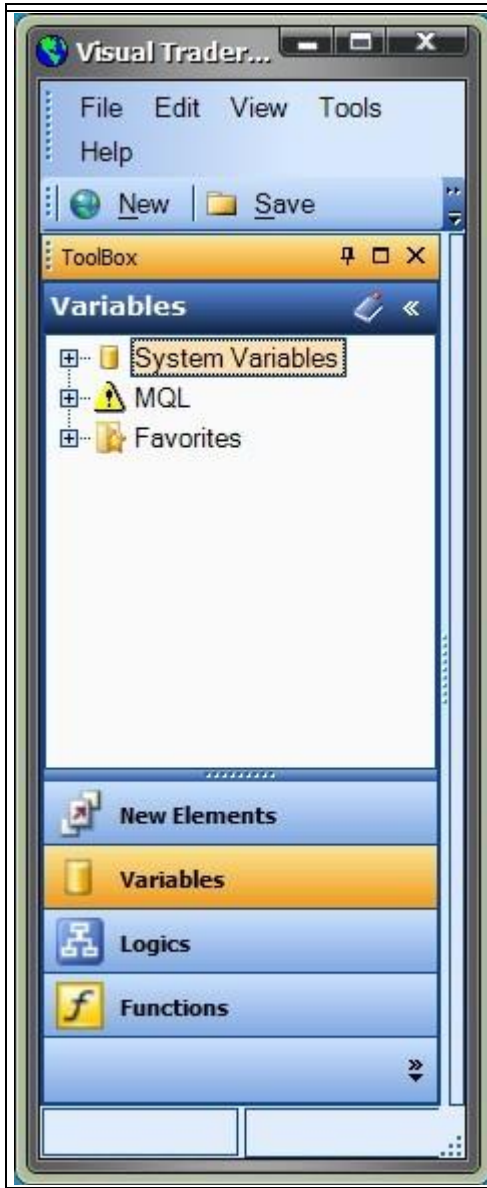
Common: This menu contains Common functions. The standard Common functions offered by MetaTrader are found directly under the Comment menu.

Time: This menu contains Time functions. The standard Time functions offered by MetaTrader are found directly under the Time menu.

Advanced: This menu contains a number of sub-menus which contain MQL functions that are not needed by most traders and require advanced knowledge of MQL to be used properly.

System Functions: This menu contains a number of sub-menus which contain the entire set of functions used to build a system. Each submenu is the name of a VTS system. The functions are stored on this menu for easy re-use.

Toolbox VARIABLES Pane



The Variables pane of the [Toolbox](#) is used to drag [Variables Elements](#) onto the [Drawing Pad](#).

There are a number of menus within the Variables Pane:

Favorites: This menu contains Variables that have been set as a Favorite. To set a Variable as a Favorite, select the Variable on the menu, and right-click and select Favorite.

MQL: This menu contains MQL built-in Variables.

System Variables: This menu contains a number of sub-menus which contain the entire set of variables used to build a system. Each submenu is the name of a VTS system. The Variables are stored on this menu for easy re-use.

Toolbox LOGICS Pane



The Logics pane of the [Toolbox](#) is used to drag [Logic Elements](#) onto the [Drawing Pad](#).

There are a number of menus within the Logics Pane:

Favorites: This menu contains Logics that have been set as a Favorite. To set a Logic as a Favorite, select the Logic on the menu, and right-click and select Favorite.

System Logics:

This menu contains a number of sub-menus which contain the entire set of Logics used to build a system.

Each submenu is the name of a VTS system.

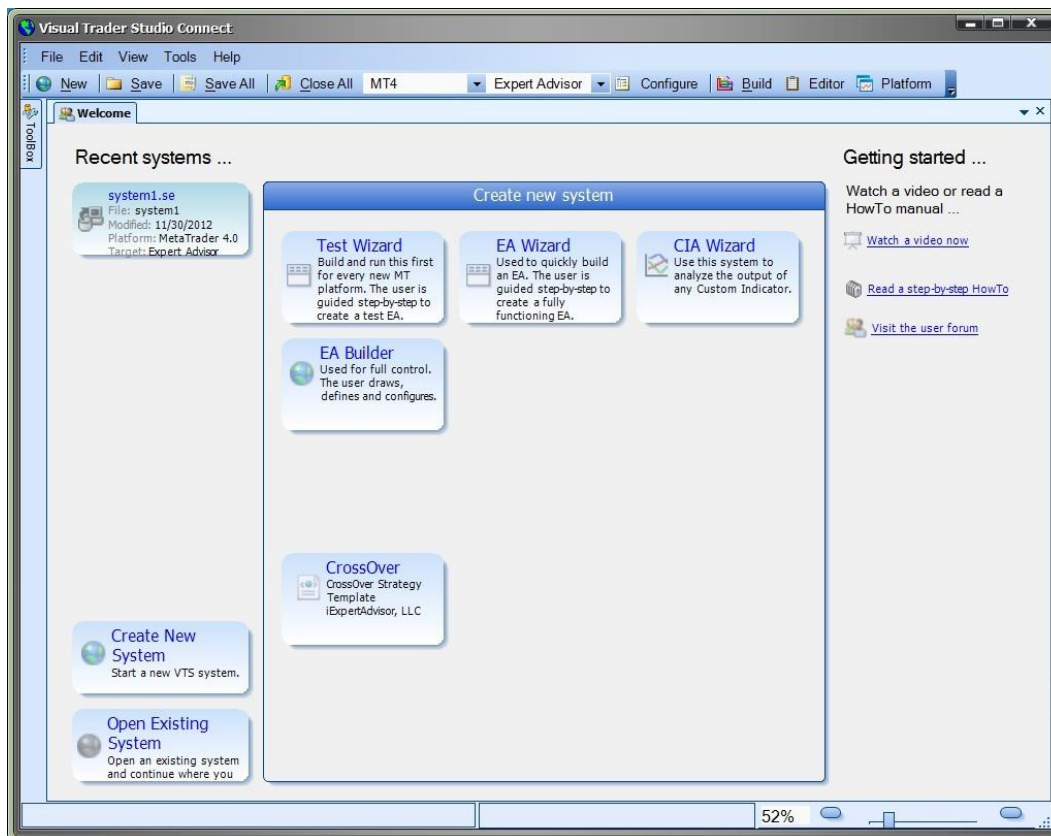
The Logics are stored on this menu for easy re-use.

Welcome Screen

The Welcome Screen is the starting point for using VTS. It is found in the center of the application. The options from the Welcome screen include:

- Open a **Recent** system. Up to four of the most recent systems are displayed.
- Create a **New** system. This will display a window for choosing the type of new system: Wizard, EA Builder, or Strategy.
- Open an **Existing** system. this will display a file-open dialog for opening a system system or package. To locate and open a system, the file filter found on the bottom right corner of the file-open dialog should be set to "VTS System Files (*.se)". To locate and open a package, the file filter found on the bottom right corner of the file-open dialog should be set to "VTS Package Files (*.zip)".
- The **Getting Started** section provides links to help resources.

Note: A Recent system can be **deleted** by right-clicking the system icon and choosing delete.



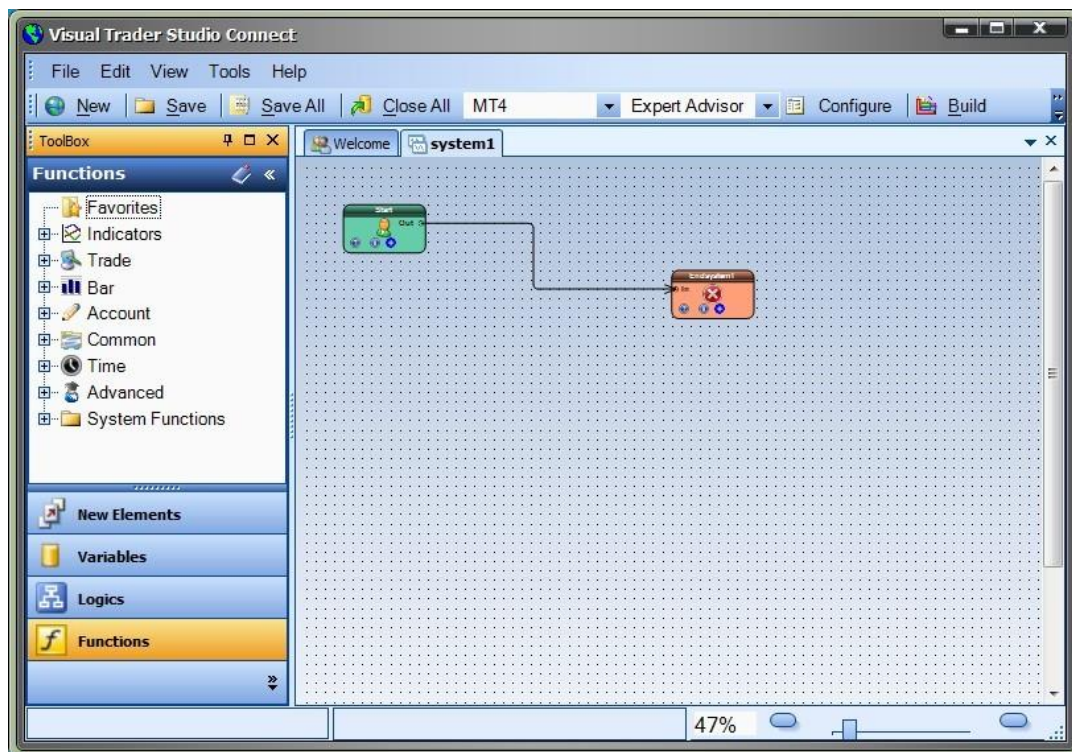
Drawing Pad

When VTS is in Drawing [mode](#), the Drawing Pad, found in the center of the application, is available for dragging, dropping and connecting [Elements](#).

- Elements can be dragged from the [Toolbox](#) onto the drawing pad.
- Right-clicking the mouse allows inserting a [Text](#), [Variable](#), [Logic](#), [MQL Function](#), or [EndElement](#) at the approximate location of the mouse.



- To **select** an [Element](#), click it once in the Caption area (the top part of the Element).
- To **select more than one Element**, left-click the mouse and enclose the elements within the mouse-drawn window.
- When an [Element](#) is selected, its border is highlighted.
- A selected [Element](#) can be moved by dragging and releasing the mouse.
- An [Element](#) can be removed from the Drawing Pad by selecting the [Element](#) and pressing the **Delete** key of the keyboard. Note: this does not delete the [Element](#) from the Toolbox - it only removes it from the Drawing.
- The [Start Element](#) can not be removed from a Drawing.
- There are Zoom buttons in the bottom right corner of the application. These are used to make the Drawing Elements larger or smaller.
- The VTS application can open multiple Drawing Pads. Each drawing has a tab with its name and can be selected by clicking the tab.



Everything You Need To Know About VTS

Basics

- [Expert Advisor Facts](#)
- [VTS Elements](#)
- [System Managers](#)
- [Special Functions](#)

[The Best Way to Learn VTS - click here](#)

<http://www.iexpertadvisor.com/best-way-to-learn-vts.html>

This web page contains a step-by-step tutorial (with videos) for learning to use VTS to build a working Expert Advisor.

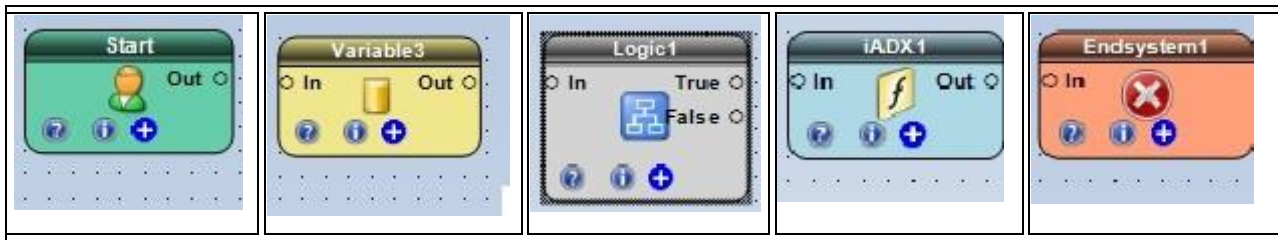
Expert Advisor Facts

What is an Expert Advisor?

- An Expert Advisor is built by creating an MQL file, with an extension of mq4, and then compiling the MQL file into Expert Advisor code.
- The extension of the Expert Advisor file is ex4.
- If the Alpari version of MetaTrader was installed on your PC, the folder "C:\Program Files\ MetaTrader - Alpari UK\experts" will contain both mq4 and ex4 files.
- The mq4 file is a human readable file.
- If you double click an mq4 file, the "MetaEditor" application will start and open the mq4 file.
- The MetaEditor application is part of the MetaTrader platform. It is automatically installed when the MetaTrader platform is installed.
- The "MetaEditor" application is used to edit MQL code. It has many useful features found in most code editors, including intelli-sense, context help and an online library.
- While editing an mq4 file from within the MetaEditor, an Expert Advisor is built by clicking the "Compile" button. If there are no syntax errors, the MetaEditor application will build the ex4 file.
- The ex4 file is not a human readable file. It is binary code interpreted by the MetaTrader platform.
- By compiling the mq4 file, you are converting the MQL language, which is human readable text, into a binary file that can be read by a machine.
- When the MetaTrader platform is started, any ex4 files in the "experts" folders are available under the "Expert Advisors" folder in the Navigator window of the MetaTrader platform.
- The VTS application converts its drawings into MQL code.
- VTS automatically creates an mq4 file and places it in the MetaTrader platform's "experts" directory. The mq4 file is then compiled into an ex4 file using the MetaEditor application.
- Both the MetaEditor and Terminal applications can be run from within VTS when VTS is configured correctly.

VTS Elements

An Element is the basic building block within a VTS System.



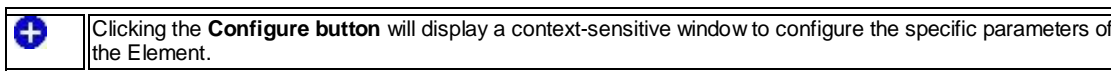
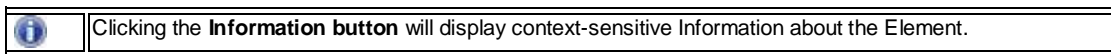
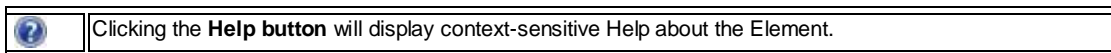
VTS Systems are built using five Elements:

- [Start](#)
- [Variable](#)
- [Logic](#)
- [Function](#)
- [End](#)

These basic Elements are [Linked](#) together to form simple or complex systems.

- The **Caption** of an Element is the upper portion of the Element that contains the name. The Caption area is used to select the Element (for moving or deleting) and is used for opening the drawing of a [Drawing Function](#) Element by double-clicking the Caption area.
- All Elements, except for the [Start Element](#), have a [Link](#) anchor labeled "In" on the left side of the Element. The "In" [Link](#) is used to connect a [Link](#) to the input of the Element.
- All Elements, except for the [Logic Element](#), have a [Link](#) anchor labeled "Out" on the right side of the Element. The "Out" [Link](#) is used to connect a [Link](#) to the output of the Element.
- All Elements, except for the [Logic Element](#), have a [Link](#) anchor labeled "Out" on the right side of the Element. The "Out" [Link](#) is used to connect a [Link](#) to the output of the Element.
- The [Logic Element](#) has two output [Link](#) anchors labeled "True" and "False" on the right side of the Element. Program execution will follow the output link that evaluates to True or False depending on the [logical condition](#).

Every Element has three buttons along the bottom: Help, Information and Configure.



START Element



- The Start Element is automatically added to every [Drawing](#). It can not be removed.
- The Start Element represents the logical start of execution of a **System** on the main system drawing.
- The Start Element represents the logical start of execution of a **Function** on a [Drawing Function](#).

VARIABLE Element

The Variable Element is used to store data.

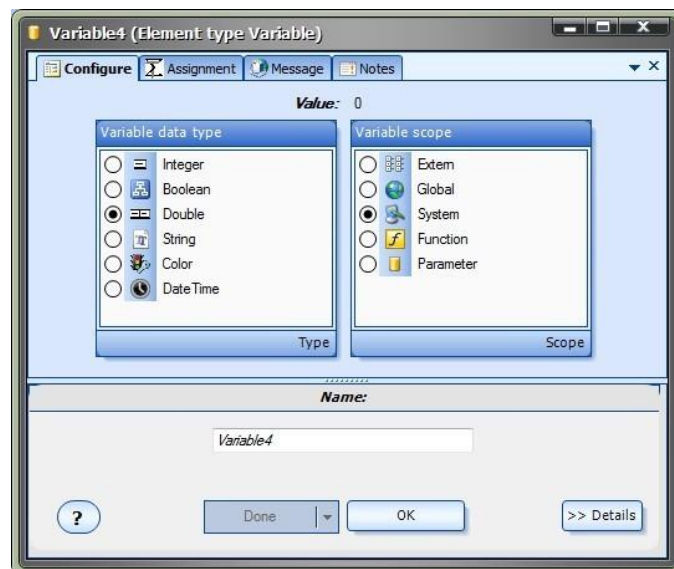


Selecting the configuration (+) button of a Variable Element will display the **Variable Configuration Window**.

There are four tabs on the **Variable Configuration Window**:

- The [Configure](#) Tab is used to set the data type and scope of a Variable.
- The [Assignment](#) Tab is used to assign a value to a Variable.
- The [Message](#) Tab is used to send a message from the Variable Element.
- The [Notes](#) Tab is used to add notes or comments to the Variable Element.

The bottom portion of the window allows the [Element Name](#) to be saved.



Configure (Variable)

The Variable [Configure](#) Tab is used to set the [data type](#) and [scope](#) of a [Variable](#).



Variable Type

A variable is used to store some useful information (or data). Its **type** is defined by the *kind of data* that needs to be saved.

A **datatype** is simply a section of memory used to store a value.

The MQL **data types** are:

Integer (int): An integer is a whole number. A whole number means the number does not have a decimal point. For example, the number of total open trades is an integer: 0, 1, 2 There is no concept of 1.5 open trades. However, the balance of a MetaTrader account is a number with a decimal point - such as 10,000.99. A number with a decimal point is called a real number in the field of mathematics. A real number data type is called a double in MQL.

Double (double): A double is a real number. A real number means the number has a decimal point. The balance of a MetaTrader account is a real number - such as 10,000.99.

Boolean (bool): A boolean is a data type that can one of two values: true or false.

String (string): A string is one or more characters. A string value is set using double quotes. For example: message="Hello World".

Void (void): A void type is a "nothing" type. It is used in to indicate that no data type is inferred.

Date and Time (datetime): A datetime data type is used to store calendar dates and times. The data type itself is actually an integer. The current time is an integer whose value is the number of seconds elapsed since midnight January 1, 1970. (This is a common approach, used in many computer languages, to define the current time.)

Color (color): A color data type is actually an integer type. This data type is defined to make color assignments easy.

Variable Scope

The word **scope** is used in programming to define the visibility of a data structure. That is, if a variable has *global scope* it can be accessed from anywhere within the program. If a variable has *function scope* it can only be accessed within the function that it is defined.

The levels of variable scope in VTS are:

extern: A extern variable is available as input to an [Expert Advisor](#).

global: A global variable saves its value between [Expert Advisor](#) invocations.

system: A system variable is accessible from any function of an [Expert Advisor](#).

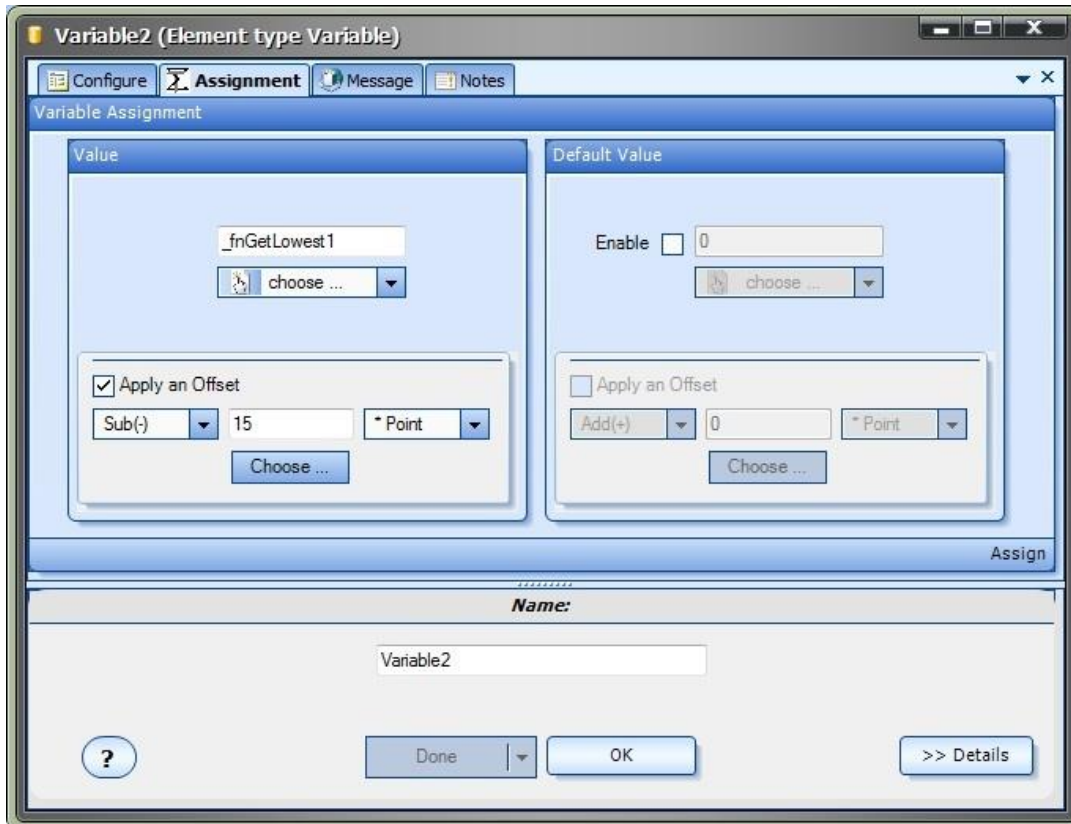
function: A function variable is only accessible from the function where it is defined.

parameter: A [parameter](#) variable is used to send or receive data to or from a function. The variable is only accessible from the function where it is defined. Note: a parameter variable can be set in [Function Drawings](#) only, not on the [main system drawing](#).

Assignment

Assignment (Variable)

The [Assignment](#) Tab is used to assign a value to a Variable.



- The Value and the [Default Value](#) can be entered on the Assignment Tab.
- The [Default Value](#) is the value that is set when additional copies of the [Variable](#) Element are added to a [Drawing](#).
- Note: A [Default Value](#) does not need to be assigned.
- A value can be entered manually by typing in the text-box.
- The [choose](#) button is used to select the value from an existing [Element](#).
- The [expression editor](#) is used to enter a [mathematical expression](#).
- The **Apply an Offset** section is used to *add or subtract* a value *to or from* the variable. A common usage is to add a number of [points](#) to a value such as the highest high in the last 24 bars.
 - Check the *Apply an Offset* checkbox to enable the option.

- The first pull-down menu is used to define the action as subtraction (**Sub(-)**) or addition (**Add(+)**).
- The middle entry box is used to enter the value to be added or subtracted. This can be a literal value, such as 15, or it can be another previously defined variable such as *MyOffset*. The value can also be set using the [Choose](#) button. Note: an [input/extern](#) variable can be defined and selected so that the actual value can be set each time the EA is attached to a chart.
- The last pull-down menu is used to select if the value is multiplied or divided by the MQL built-in variable [Point](#), or if neither is done. The available options are:
 - *** Point** (Multiply by Point)
 - **/ Point** (Divide by Point)
 - **None** (Do not multiply or divide)

A common usage is to multiply an Integer value, such as 12, by the MQL built-in variable Point to convert the number to a price value.

For Example, on a EURUSD chart: $12 * \text{Point} = 12 * 0.00001 = 0.00012$

For more information on MQL built-in variables and functions, see the [MQL Help](#) section.

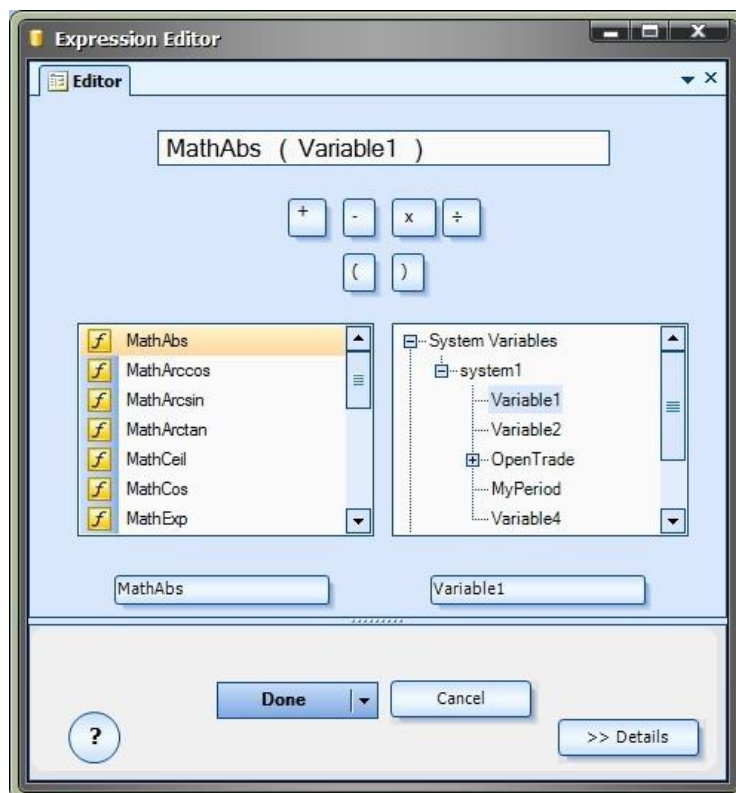
Expression Editor

The [expression editor](#) is used to enter a [mathematical expression](#) into the value of a [Variable](#).

An example of a simple expression is the sum of two numbers:
Bid + Ask

A more complicated expression may find the average of two numbers:
(Bid + Ask)/2

The [MQL](#) language offers many Math functions for forming complex expressions. A very complex expression would probably be easier to implement using an [MQL Function](#), however, the Expression Editor is convenient for simple expressions.

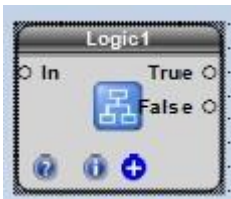


- The Expression Editor text-box can be manually edited.
- The left pane of the Expression Editor lists the available MQL Math functions. To insert a Math function, select it and it will appear in the button below the list; then select the button to add the value to the expression.

- The right pane of the Expression Editor lists the available [System Variables](#). To insert a variable, select it and it will appear in the button below the list; then select the button to add the value to the expression.
- The operators for addition, subtraction, multiplication and division can be added manually or by using the buttons along the top.
- Open and close parenthesis can be added manually or by using the buttons along the top.
- **NOTE: Some knowledge of MQL and mathematics is required to use the Expression Editor correctly. [Syntax errors](#) can be injected into the trading system if care is not taking when entering expressions.**

LOGIC Element

The **Logic** Element is used to define logical conditions. The logic of the strategy will follow the path of the logic that is true.

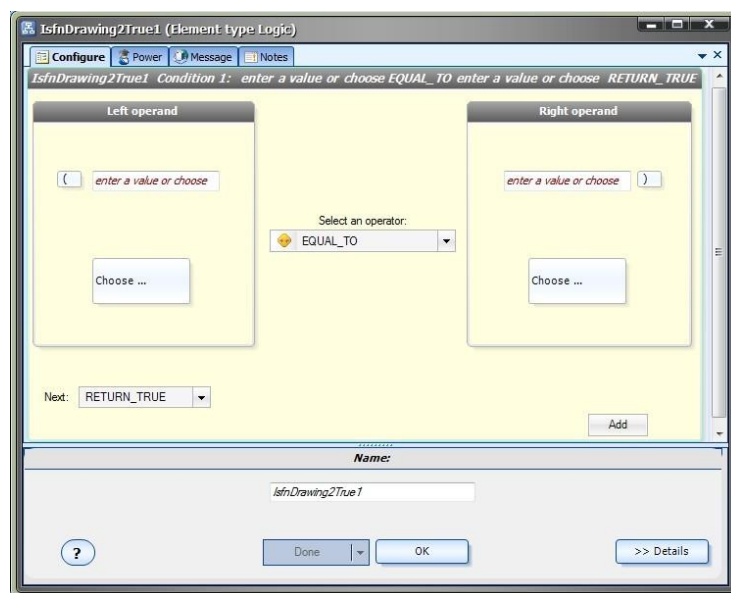


Selecting the configuration (+) button of a Logic Element will display the **Logic Configuration Window**.

There are up to four tabs on the **Logic Configuration Window**:

- The [Configure](#) Tab is used to add [logical conditions](#) to a Logic Element.
- The [Power](#) Tab is used to configure advanced (powerful) techniques to the Logic.
- The [Message](#) Tab is used to send a message from the Logic Element.
- The [Notes](#) Tab is used to add notes or comments to the Logic Element.

The bottom portion of the window allows the [Element Name](#) to be saved.



Condition (Logical)

A logical condition consists of:

- Left [Operand](#)
- Right [Operand](#)
- An [Operator](#)
- [Next](#) Value

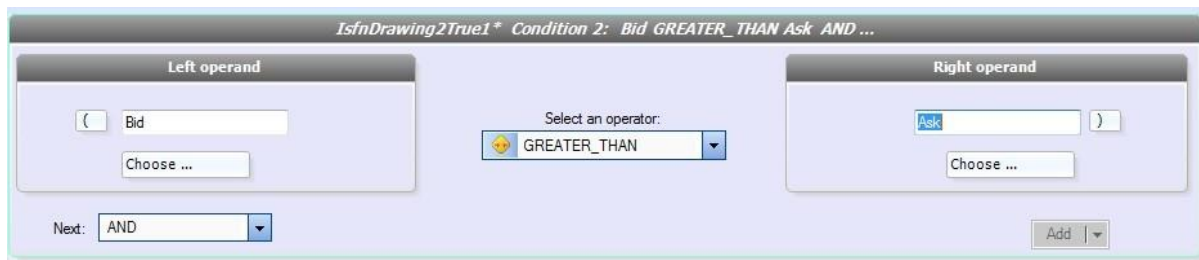
The Left and Right [Operand](#) can be entered manually, or the [Choose](#) button can be used to select a value.

The [Operator](#) menu offers the following values:

- EQUAL_TO
- NOT_EQUAL_TO
- GREATER_THAN
- GREATER_THAN_EQUAL (greater than or equal to)
- LESS_THAN
- LESS_THAN_EQUAL (less than or equal to)
- CROSS_UP (the left operand value crosses up through the right operand value)
- CROSS_DOWN (the left operand value crosses down through the right operand value)

The [Next](#) menu offers the following values:

- AND
Both (adjacent) conditions must be **true** for the [Logic](#) to return a **True** value
- OR
If either of the (adjacent) conditions are **true**, the [Logic](#) will return a **True** value.
- RETURN_TRUE
If the expression evaluates to true, the [Logic](#) result will be **True** and execution will follow the [TrueLink](#) of the [Logic](#) Element.

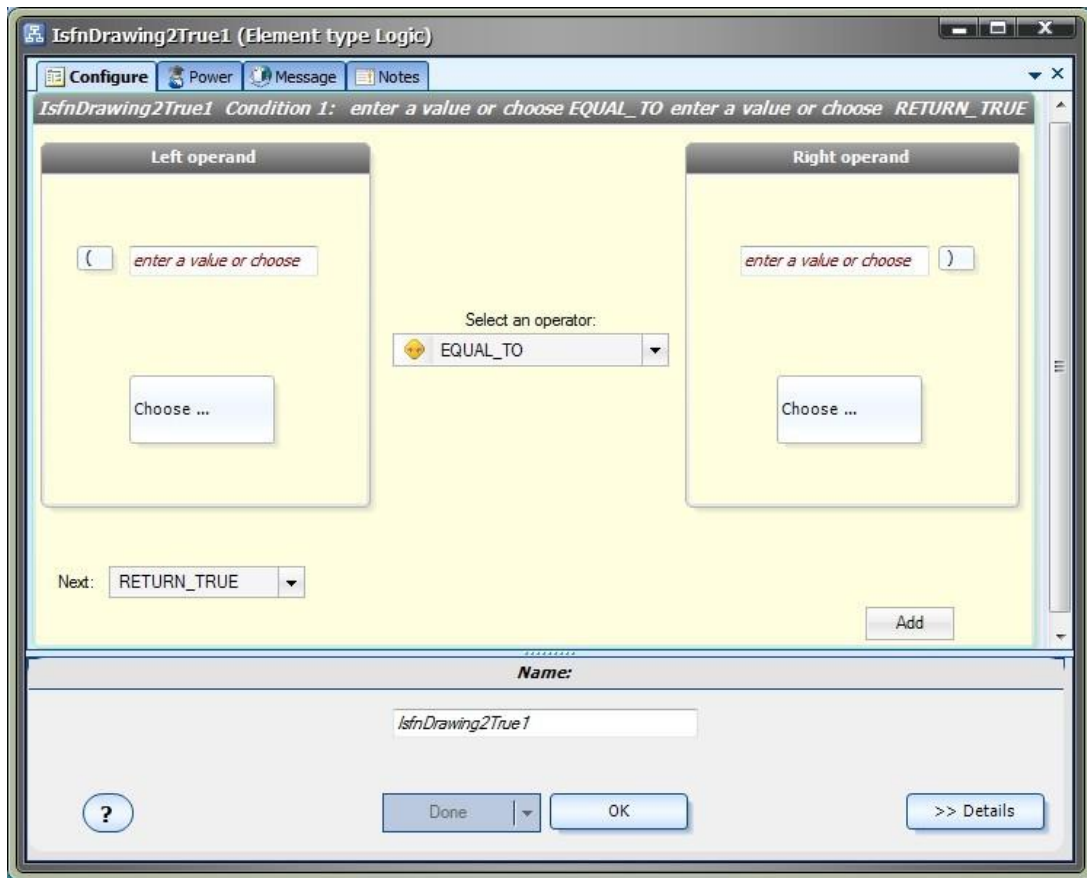


Configure (Logic)

The **Logic Configure Tab** is used to configure the [logical conditions](#) of a [Logic](#) Element.

- When a Logic is first created it contains one [logical condition](#).
- [Logical conditions](#) can be added by selecting the **Add** button.
- When there is more than one [logical condition](#), a condition can be removed by selecting its **Remove** button.
- When there is more than one [logical condition](#), the **Next** selection must be set to **ADD** or **OR** for all conditions but the last one.
- When **Next** is set to **AND**, both (adjacent) conditions must be **true** for the [Logic](#) to return a **True** value.

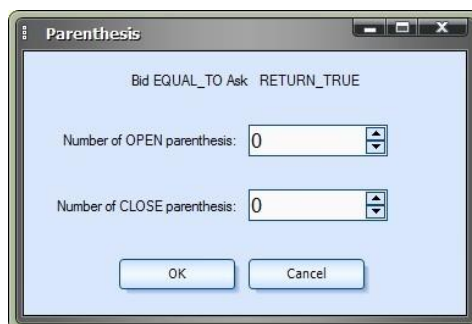
- When **Next** is set to **OR**, if either of the (adjacent) conditions are **true**, the **Logic** will return a **True** value.
- Selecting either the **Open "("** or **Close ")"** **Parenthesis** will allow adding parenthesis around any condition.



Parenthesis

Similar to mathematics, parenthesis alone can change the value of an expression. Understanding how parenthesis effect the evaluation of a logical expression allows complex logic to be implemented that might not be otherwise possible.

- Simple stated: expressions within parenthesis are evaluated first; and then their results are applied to the overall expression.
- The parenthesis dialog allows any number of parenthesis to be added to each [logical condition](#).
- The parenthesis do not need to be [balanced](#) for each condition, but they must be [balanced](#) for the entire Logical expression.
- **Balanced** parenthesis means the number of open parenthesis is equal to the number of close parenthesis.
- **NOTE: Unbalanced parenthesis will cause syntax errors.**



END Element

The End Element is used to terminate the logical execution of a system or [Drawing Function](#).



Selecting the configuration (+) button of a End Element will display the **End Configuration Window**.

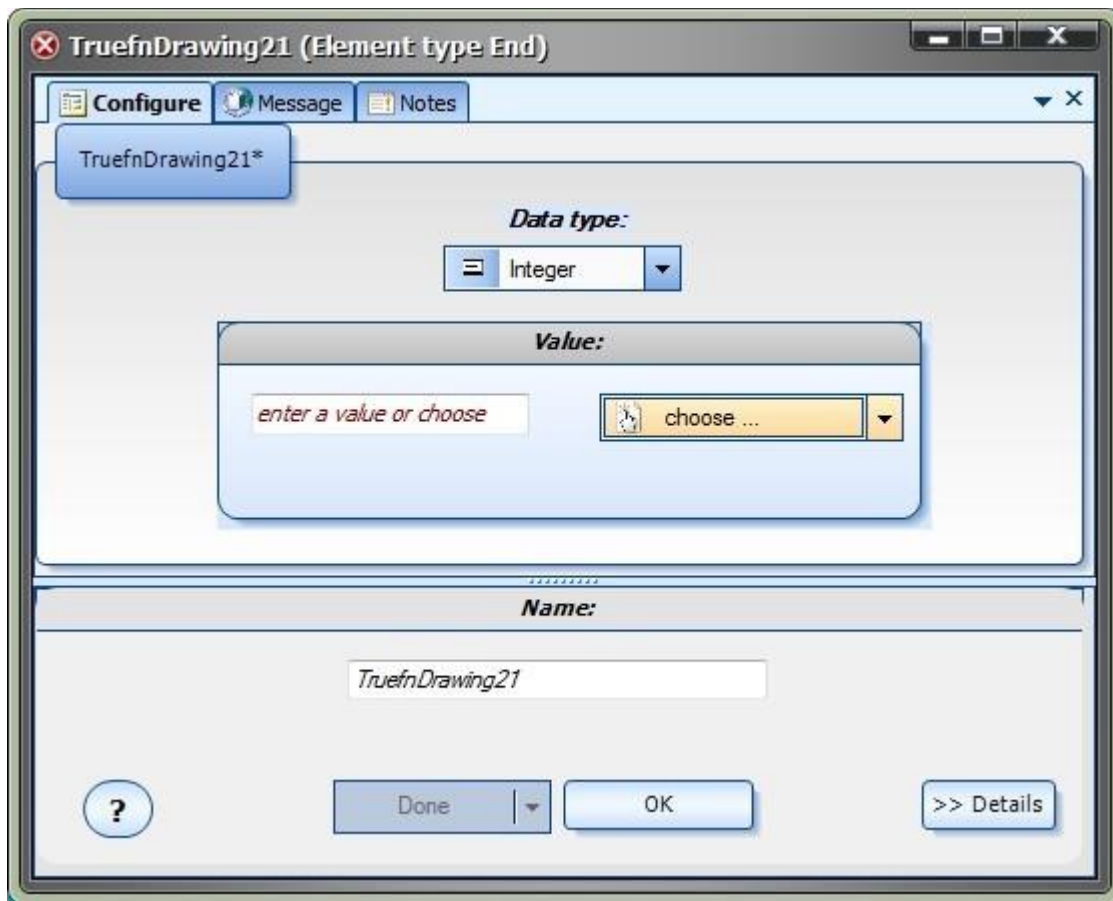
There are up to four tabs on the **End Configuration Window**:

The [Configure](#) Tab is used to set the [data type](#) and value of an End Element.

The [Message](#) Tab is used to send a message from the End Element.

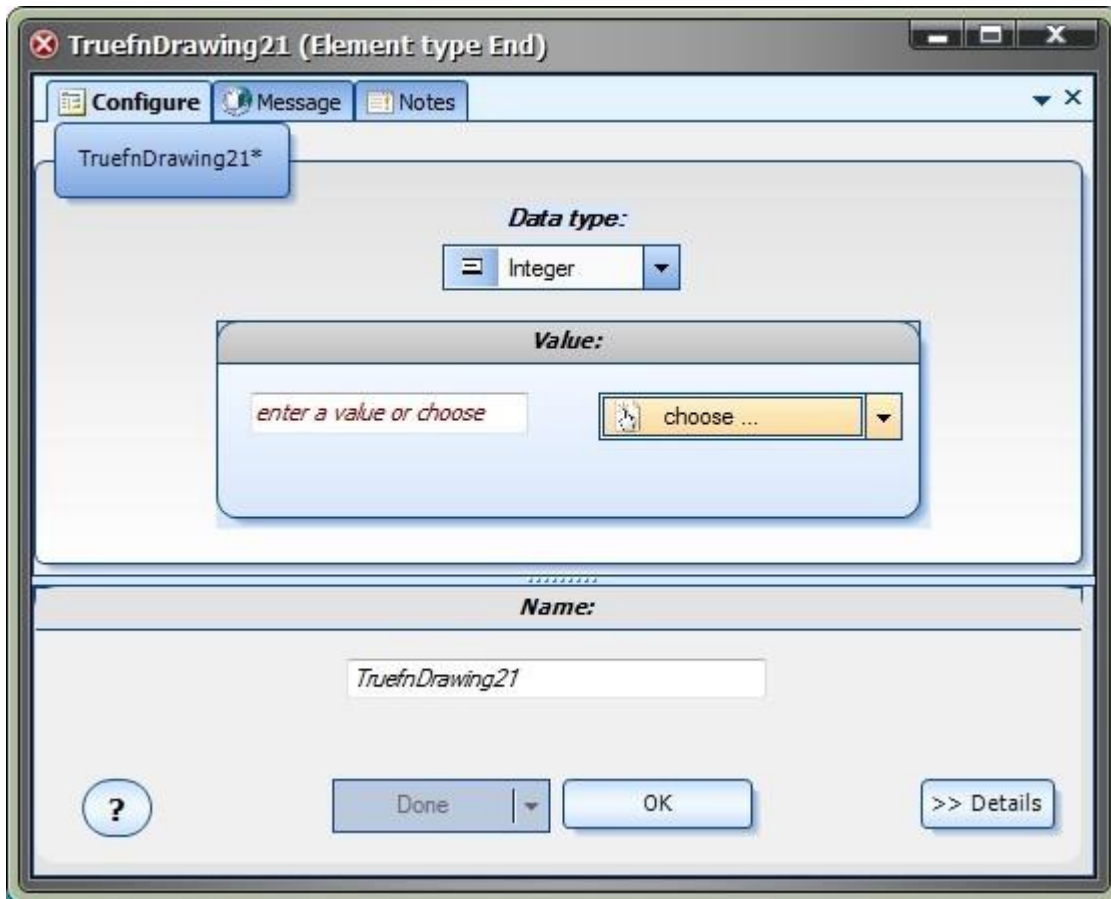
The [Notes](#) Tab is used to add notes or comments to the End Element.

The bottom portion of the window allows the [Element Name](#) to be saved.



Configure (End)

The End Configure Tab is used to set the [data type](#) and value of an [End](#) Element.



- The **Value** can be entered manually, or the [Choose](#) button can be used to select a value.
- The [data type](#) menu offers the following values:
 - Integer
 - Boolean
 - Double
 - String
 - Color
 - DateTime
 - Void
- The [data type](#) of an End Element must match the [data type](#) of the [Drawing Function](#) or [System Drawing](#) that the [End](#) Element is placed on.
- An [End](#) Element **returns** the value of a drawing. A simple example is a drawing that returns a Boolean value: the [Drawing Function returns](#) a value of **True** or **False**.
- Note: The main [System Drawing](#) (for an MQL Expert Advisor) returns an Integer value.

FUNCTION Element

Function [Elements](#) provide complete, re-usable functionality to a VTS System.

There are three type of functions in VTS:

- [Platform](#) functions: supplied by the MQL and VTS platforms
- [Drawing](#) functions: drawings, created by users
- [MQL](#) functions: native MQL code, created by users



PLATFORM Function

Platform functions are built-in functions. They are provided by the [MetaTrader](#) platform and by VTS. All platform functions are available for *Drag and Drop* from the [Function Toolbox](#).



Selecting the configuration (+) button of a Platform function Element will display the **Function Configuration Window**.

There are three to five tabs on the **Function Configuration Window**:

- The [Configure](#) Tab is used to set the values of the [parameters](#) of the Function.
- The [Advanced](#) Tab is used to set the values of the advanced [parameters](#) of the Function.
- The [Power](#) Tab is used to configure advanced (powerful) techniques to the Function.
- The [Message](#) Tab is used to send a message from the Function Element.
- The [Notes](#) Tab is used to add notes or comments to the Function Element.

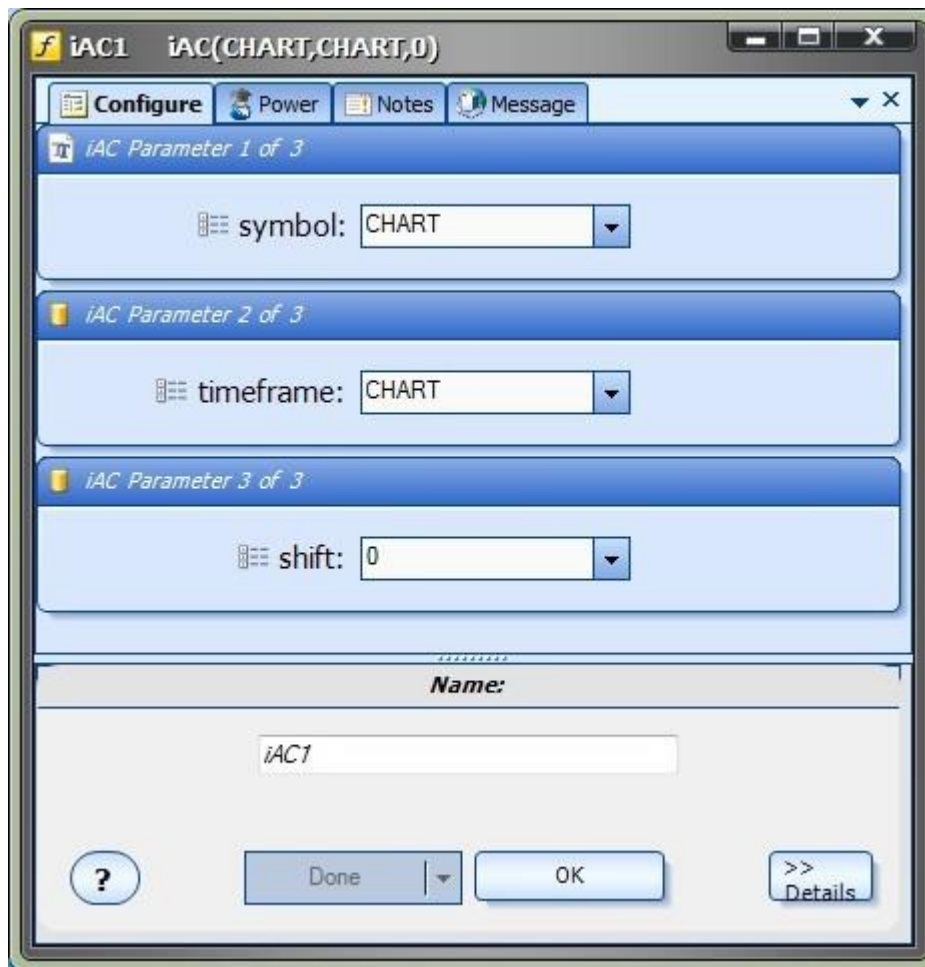
Note: If the Function does not have any [parameters](#), the [Configure](#) Tab will not be present.
If the Function does not have any advanced [parameters](#), the [Advanced](#) Tab will not be present.

The bottom portion of the window allows the [Element Name](#) to be saved.



Configure (PLATFORM Function)

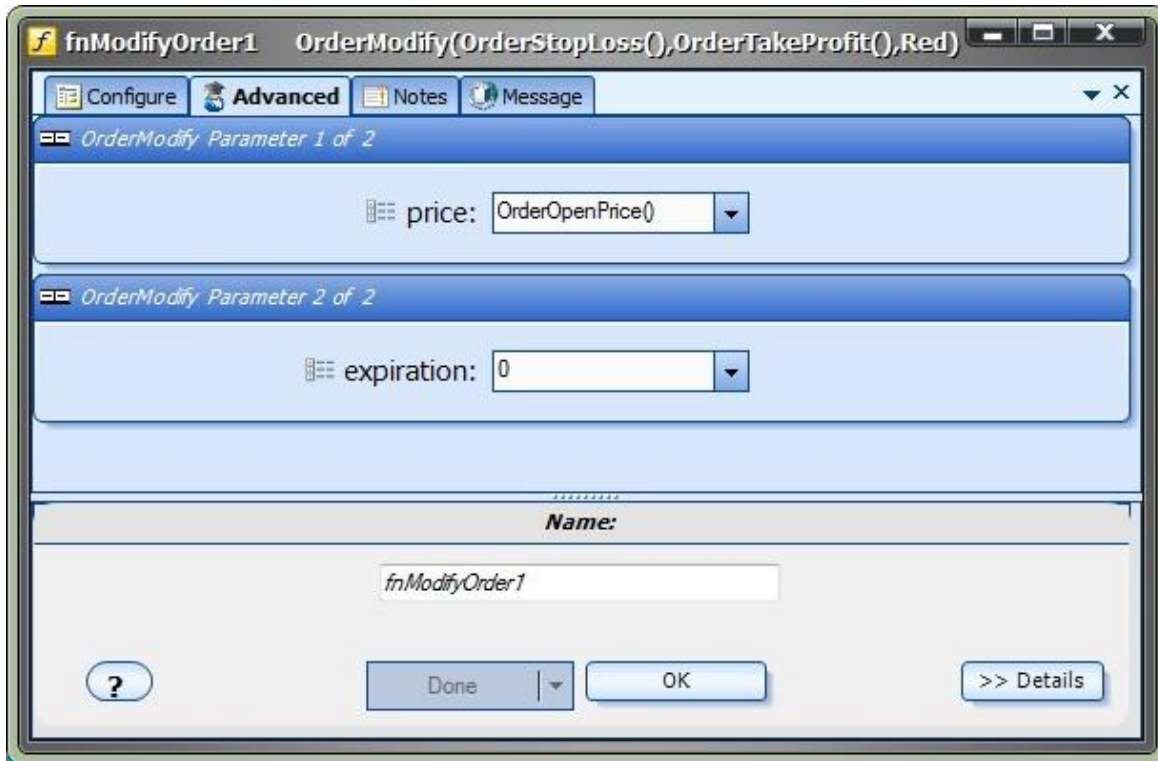
The Function [Configure](#) Tab is used to set the values of the [parameters](#) of the Function.



- For each MQL Platform function, VTS creates a [parameter](#) selection box for each [parameter](#) that is defined by the function.
- Whenever possible, VTS creates a pull-down menu with context-sensitive selections for each [parameter](#). For example, the symbol parameter provide these selections from its pull-down menu:
[CHART](#)
EURUSD
USDJPY
GBPUSD
...
- Selecting some [parameters](#) will open a window specific for entering that [parameter](#). For example, the shift [parameter](#) provides access to the [shift dialog](#).
- Entering an invalid value for a parameter will display an error message and disable the **Save** or **Done** button.

Advanced (PLATFORM Function)

- The Advanced Tab of the Platform function is **very** similar to the [Configure](#) Tab.
- The Advanced Tab is used to set the values of the advanced [parameters](#) of the Function.
- Some [parameters](#) are considered advanced because they rarely need to be changed from their [default](#) values.

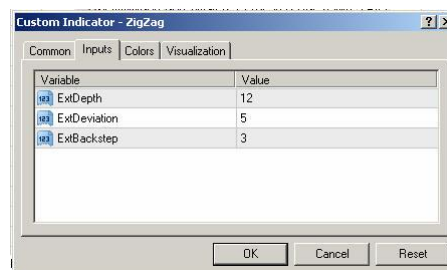


Custom Indicators

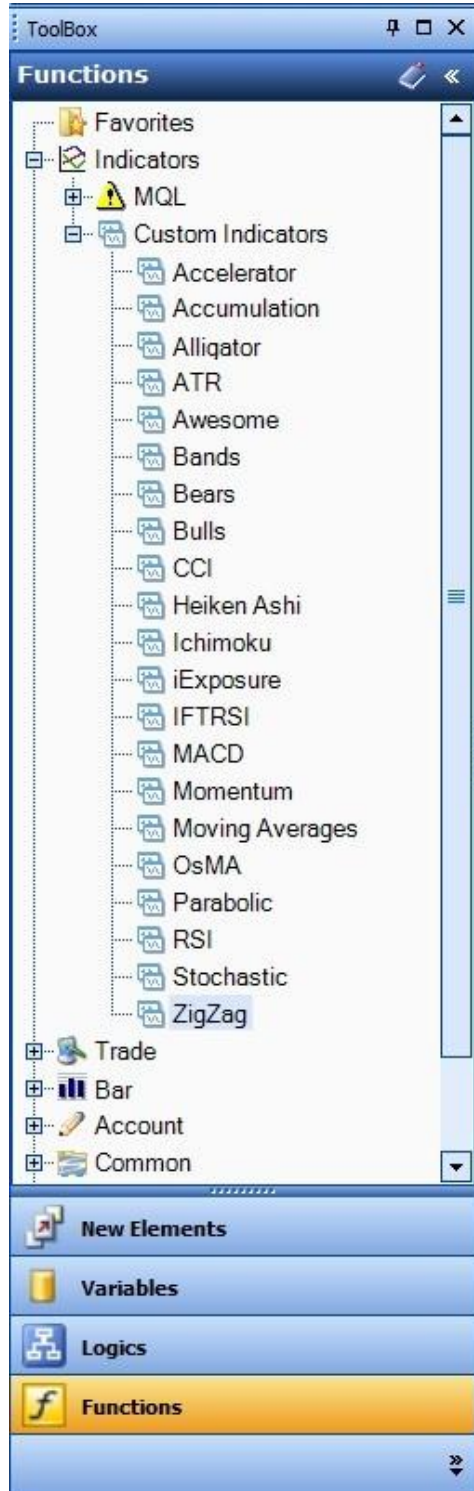
- [VTS](#) searches for **Custom Indicators** in the [MetaTrader](#) Platform's *Custom Indicator folder*.
- The *Custom Indicator folder* is the "**experts\indicators**" folder beneath the platform installation folder.
- Any file with an extension of **ex4** in the *Custom Indicator folder* is displayed in the [Function Toolbox](#), under the menu:
 - **Indicators-> Custom Indicators**
- **Custom Indicators** are similar to platform functions, except:
 - The [parameters](#) of a **Custom Indicator** must be defined (if any)
 - The **output lines** of a **Custom Indicator** must be defined (if any)
- **Custom Indicators** are dragged and dropped onto the Drawing Pad like other Elements.
- If the [parameters](#) or **output lines** of a **Custom Indicator** have **not** been configured, selecting the configuration (+) button of a **Custom Indicator** Element will display the **Custom Indicators Definition Window**
- If the [parameters](#) or **output lines** of a **Custom Indicator** have been configured, selecting the configuration (+) button of a **Custom Indicator** Element will display the [Function Configuration Window](#)
- [VTS](#) searches for **Custom Indicators** in the [MetaTrader](#) Platform's *Custom Indicator folder*.
- The *Custom Indicator folder* is usually something like :
C:\Program File\MetaTrader4\experts\indicators (This depends on where you chose to install [MetaTrader](#))
- Any file with an extension of **ex4** in the *Custom Indicator folder* is displayed in the [Function Toolbox](#), under the menu **Indicators-> Custom Indicators**
- **NOTE:** If the **Indicators->Custom Indicators** menu in the [VTSFunction Toolbox](#) tab is empty, it's most likely because the MetaTrader [MT Paths](#) have not been configured.
- To make your custom indicator available for selection from the [VTSToolbox](#), simply copy the custom indicator file into the **custom indicator folder**.
- If the [parameters](#) or **output lines** of a **Custom Indicator** have **not** been configured, selecting the configuration (+) button of a **Custom Indicator Element** will display the **Custom Indicators Definition Window**
- If the [parameters](#) or **output lines** of a **Custom Indicator** have been configured, selecting the configuration (+) button of a **Custom Indicator Element** will display the [Function Configuration Window](#)

Using a Custom Indicator – The ZigZag Example

- As an example, we'll define the [Input Variables](#) and *Output Lines* for the custom indicator **ZigZag**. The custom indicator **ZigZag** is included with the MetaTrader platform.
- Open the [MetaTrader](#) platform and attach the custom indicator **ZigZag** to any price chart. (From the Navigator window of the MetaTrader platform, expand "Custom Indicators" menu and double-click **ZigZag**.)
- When the indicator is attached to the price chart the following window, called the **Indicator Configuration** window, will appear. Leave this window open to help when entering the custom indicator's **Input Variables** and *Output Lines* into [VTS](#).
- This is the **Indicator Configuration** window.



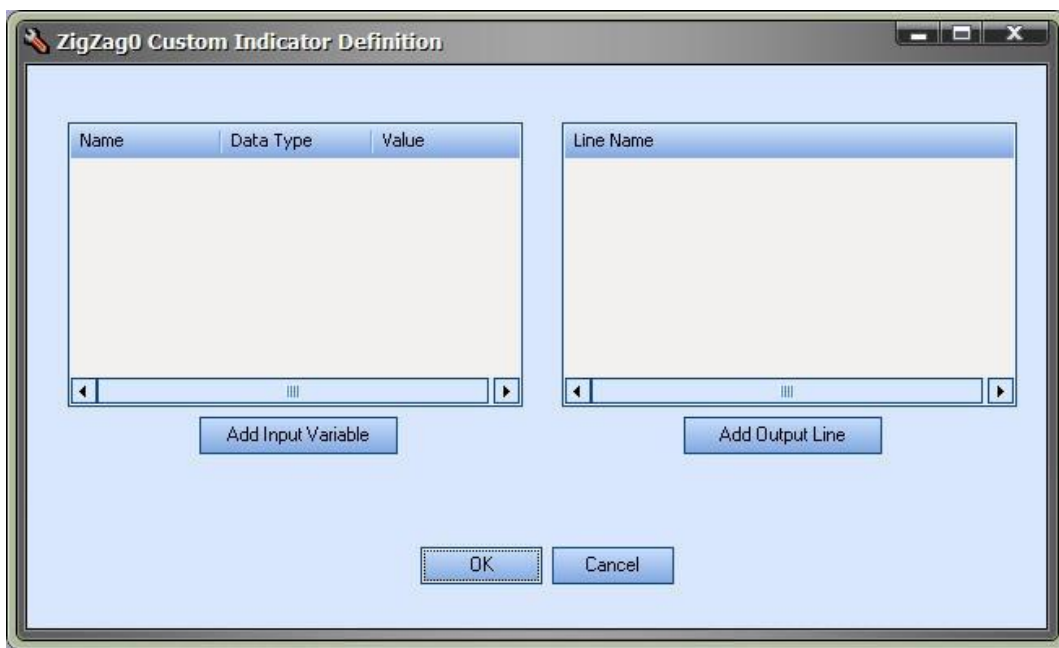
- Start a or open a Trading System in [VTS](#).
- Select the Functions Toolbox Tab and expand the **Custom Indicators** menu.
- This is the Toolbox with the **Custom Indicators** menu expanded:



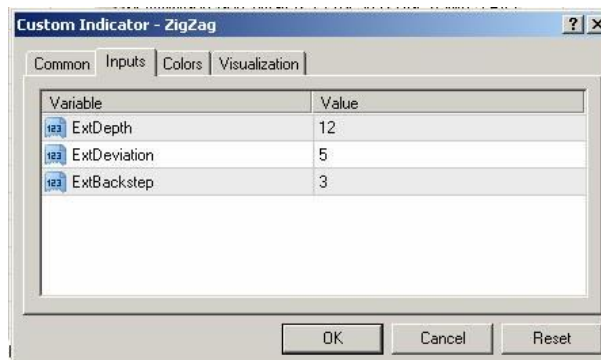
- NOTE: If the folder **Custom Indicators** is not shown from within the [VTS](#) Functions Toolbox tab, it most likely because the MetaTrader MT Pathshas not yet been configured.
- Select the **ZigZag** custom indicator and drag it onto the [VTS](#) drawing pad.



- If the custom indicator has not been configured, selecting the configuration (+) button will display the **Custom Indicator Definition** window.
- The **Custom Indicator Definition** allows entry of the indicator's **Input Variables** and **Output Lines**.



- Note, the input variables for your custom indicator are best identified from the **Indicator Configuration** window shown when attaching the indicator to a MetaTrader price chart.
- Select the **Inputs** tab to view the inputs.



The **Indicator Configuration** window for the **ZigZag** indicator shows three input variables: **ExtDepth**, **ExtDeviation** and **ExtBackstep**.

The data type of all three variables is **Integer**, as identified by the “123” symbol to the far left.

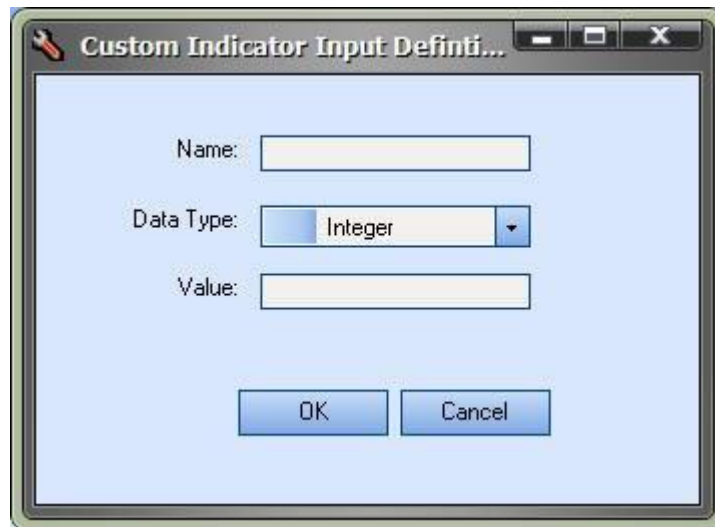
The default value for **ExtDepth** is 12.

The default value for **ExtDeviation** is 5.

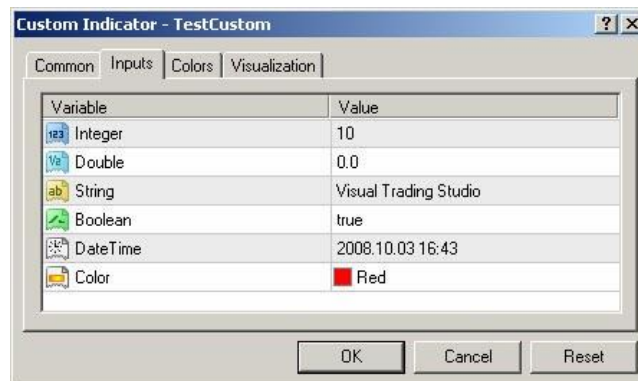
The default value for **ExtBackstep** is 3.

To add an input variable, select the **Add Input Variable** button.

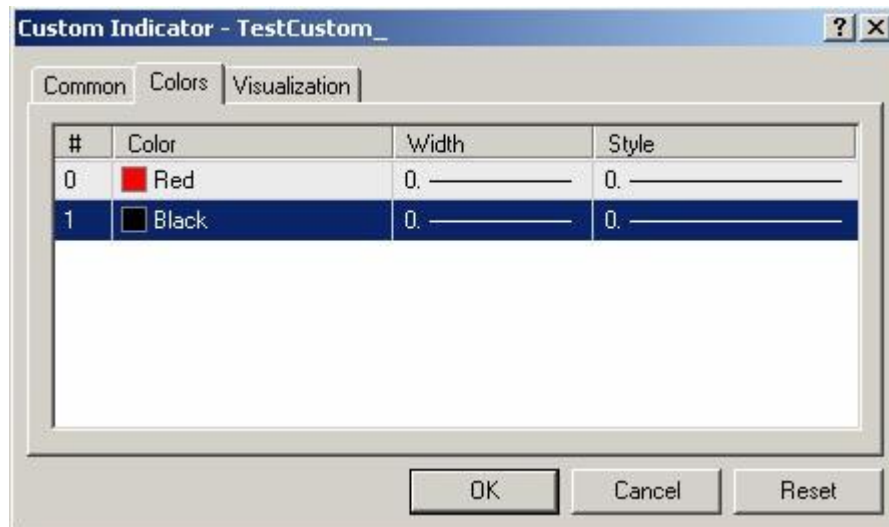
Return to the **Custom Indicator Definition** window and select the **Add Input Variable** button. The **Custom Indicator Input Definition** window will be shown.



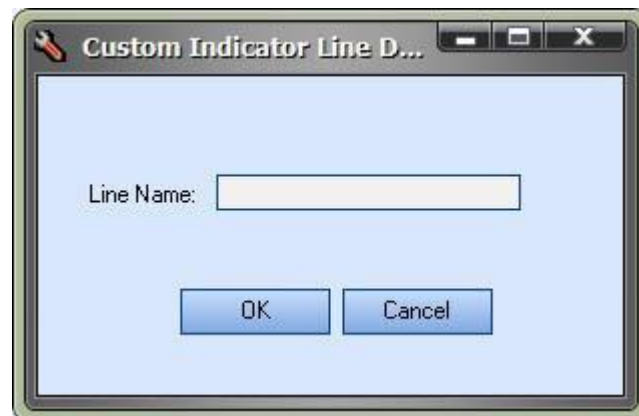
- Enter the values to define the Input variable **ExtDepth**:
 - Enter the *Name* as **ExtDepth**.
 - Select the *Data Type* as int.
 - Enter the *Value* as 12.
 - Select **OK** to save the values.
- Repeat this procedure for the other input variables **ExtDeviation** and **ExtBackstep**.
- To identify the various MQL data types, use the example below. Note the icon to the far left that defines the data type. In this example, the name of the data type is shown as the name of the variable.



- To enter the output lines for the **ZigZag** indicator, refer back to the **Indicator Configuration** window.
- Select the **Colors** tab to view the output lines.
- The **Indicator Configuration** window for the **ZigZag** indicator shows one output line: **Red**.

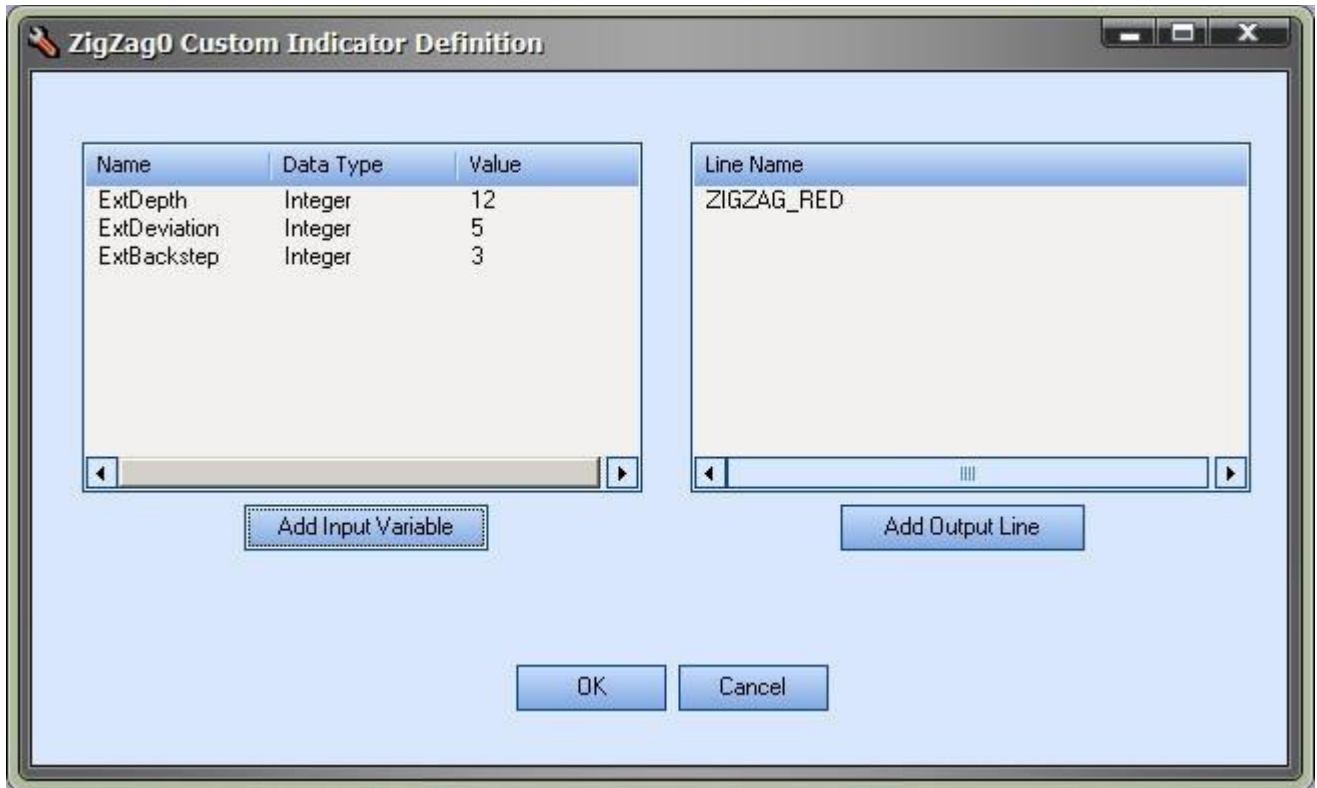


- To define the **Red** output line in VTS, return to the **Custom Indicator Definition** window and select the **Add Output Line** button.
- The **Custom Indicator Line Definition** window will be shown.

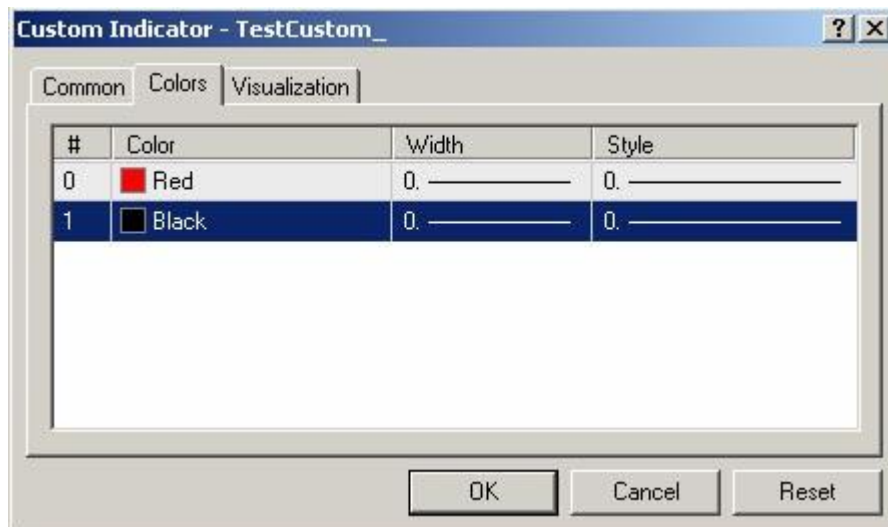


- Enter the value **Red** into the **Line Name** field and select **OK**.

- The final configuration is shown below.



- If entering values for a custom indicator with more than one output line, be sure to enter the values in the same order as shown in the **Indicator Configuration** window.
- In the below example, note the number (in the # column) to the far left of the **Color** name.



- The **ZigZag** indicator can now be dragged from the VTS Toolbox onto the VTS Drawing Pad just like any other indicator. The information entered through the previous windows is used to prompt the user for the correct configuration information for the custom indicator.
- The **Input Variables** appear as **input** parameters.
- The **Output Lines** appear as selection items for the **mode** parameter.



Note: If a mistake was made when entering any of the values, select the **(i)** button from the bottom of the **ZigZag** element to return to the **Custom Indicator Definition** window.



The iCustom MQL Function

The information for the *iCustom* function is in the following table.

double iCustom(string symbol, int timeframe, string name, ..., int mode, int shift)

Calculates the specified custom indicator and returns its value. The custom indicator must be compiled (*.EX4 file) and be in the terminal_directory\experts\indicators directory.

Parameters:

symbol - Symbol the data of which should be used to calculate indicator. NULL means current symbol.

timeframe - Timeframe. It can be any of [Timeframe enumeration](#) values. 0 means the current chart timeframe.

name - Custom indicator compiled program name.

... - Parameters set (if necessary). The passed parameters and their order must correspond with the declaration order and the type of extern variables of the custom indicator.

mode - Line index. Can be from 0 to 7 and must correspond with the index used by one of [SetIndexBuffer](#) functions.

shift - Index of the value taken from the indicator buffer (shift relative to the current bar the given amount of periods ago).

Sample:

```
double val=iCustom(NULL, 0, "SampleInd",13,1,0);
```

- What makes the *iCustom* function difficult to use is the fourth parameter. The ellipsis (...) indicates there can be zero to N number of parameters defined. The number depends on the specific custom indicator.
- The mode parameter is used to get the value of the specific line output of the custom indicator. Many indicators have more than one line drawn on the chart. This value must be defined correctly to receive the value of the correct line.
- The remaining parameters are straightforward.
- Note: An indicator does *not* need to be shown on a price chart to be used by an Expert Advisor.

DRAWING Function

Drawing Functions are drawings created by the user.

- When a **Drawing Element** is dragged from the Toolbox [New Elements](#) pane, the [New Drawing](#) window is displayed.
- When the [New Drawing](#) window is closed by clicking OK, a new [Drawing Pad](#) is created and added to the main application.
- The Drawing may be edited like any other drawing: an unlimited number of [Elements](#) may be added in any configuration.
- When the Drawing is saved, the **Drawing Function** Element's name will appear in the [Function Toolbox](#) under the [System Functions](#) menu.
- The **Drawing Function** can be dragged from the [Toolbox](#) and dropped onto a Drawing just like any other [Element](#).

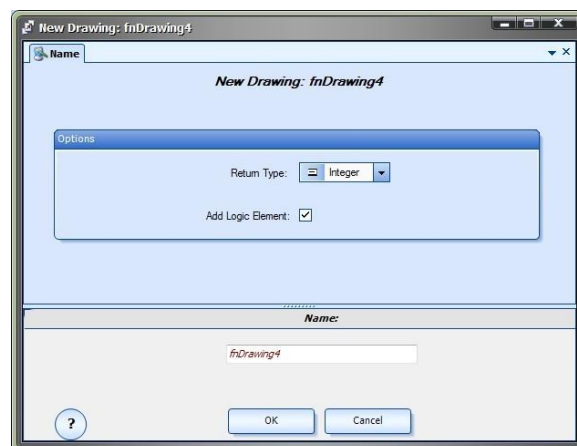


- Note: A **Drawing Function cannot be added to itself**.
- To open a **Drawing Function** whose [Element](#) is on the [Drawing Pad](#), double-click the [caption](#) of the [Element](#).
- Selecting the [configuration\(+\)](#) button of a **Drawing Function's** Element will display the [platform function configuration window](#).
 - This allows the parameters of an **Drawing Function** to be set the same way parameters are set for a platform function.
 - The Drawing is shown as [read-only](#) diagram.

New Drawing

New Drawing Window

The **New Drawing Window** is used to name, configure and open a new [Drawing Function](#).



- A [Start](#) and an [End](#) Element are automatically added to all new drawings.
- If checked, a [Logic](#) Element and two [End](#) elements are added and connected on the drawing.

- The **Return Type** menu offers the following data types:
 - Integer
 - Boolean
 - Double
 - String
 - Color
 - DateTime
 - Void
- The bottom portion of the window allows the [Element Name](#) to be saved.

MQL Function

MQL functions are function [Elements](#) that contain native [MQL](#) code.



Users need some [MQL](#) programming knowledge to create a [MQL](#) functions. [MQL](#) functions are provided so that a VTS system can use **any** functionality that is available in [MQL](#). Professional programmers can provide complex [MQL](#) functions that you can easily add to your [Drawing](#).

There are two types of [MQL](#) functions:

- [Code Snippet](#): Lines of MQL code that you want added in-line to your Expert Advisor.
- [Prototyped Function](#): A fully [prototyped](#) function that can be called by name elsewhere in your Expert Advisor.

Code Snippet (MQL Function)



- Selecting the configuration (+) button of an [MQL](#) function [Element](#) will display the **MQL Function Configuration Window**.
- There is a single tab on the **MQL Function Configuration Window**, the **MQL Tab**.
- The **MQL Tab** is used to edit native [MQL](#) code.
- VTS automatically determines if an [MQL](#) Element is a **Code Snippet** or a [Prototyped Function](#) when the [MQLElement](#) is saved.
- If the first uncommented line of code contains a correctly formed function [prototype](#), the [Element](#) will be saved as a [Prototyped Function](#). Otherwise the [Element](#) will be saved as a **Code Snippet**.
- A **Code Snippet** can be any valid [MQL](#) code.

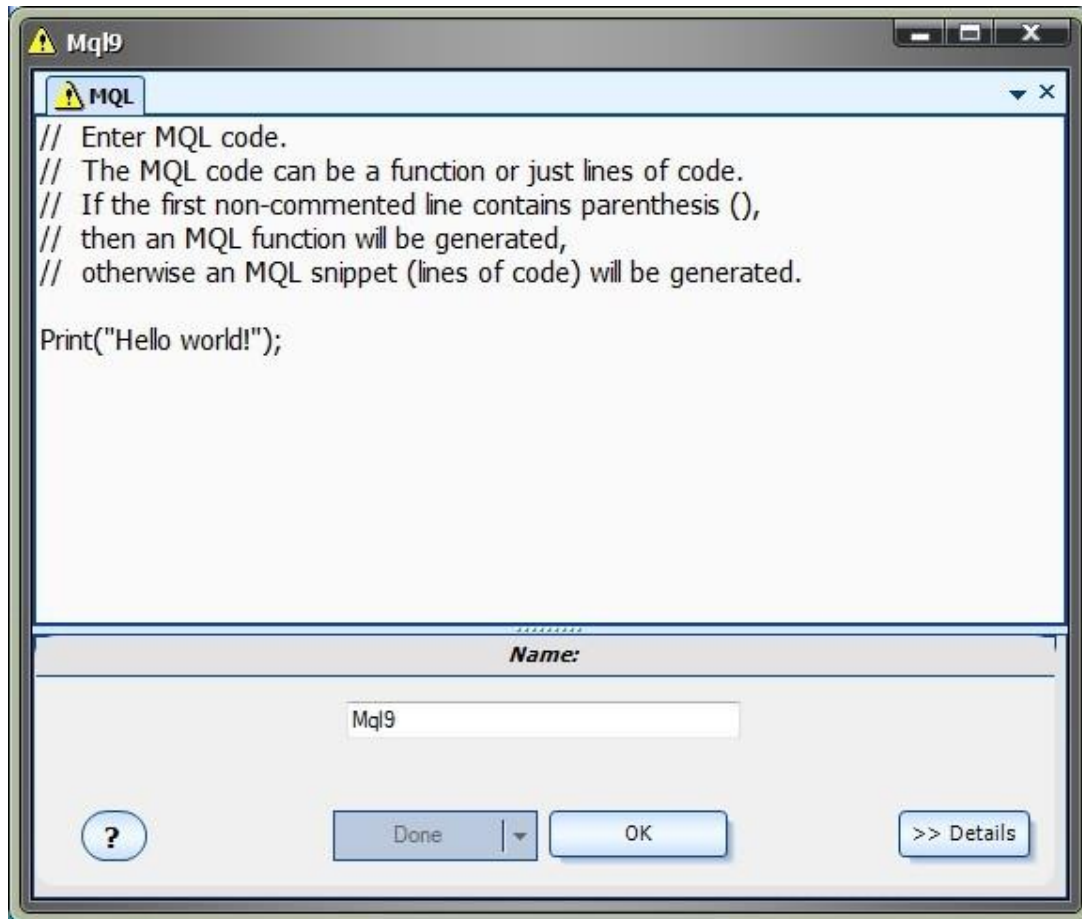
This example uses the MQL [platform function](#) **Print** :

```
Print("Hello world!");
```

This example uses a previously defined VTS Variable to assign a mathematical expression:

```
myprice = (Bid - Ask)/2;
```

- **NOTE:** Knowledge of MQL is required to create MQL Elements. [Syntax errors](#) can be injected into the trading system if care is not taking when using native MQL to create MQL Elements.



Prototyped Function (MQL Function)



- Selecting the [configuration\(+\)](#) button of an [MQL](#) function [Element](#) will display the **MQL Function Configuration Window**.
- There is a single tab on the **MQL Function Configuration Window**, the **MQL Tab**.
- The **MQL Tab** is used to edit native [MQL](#) code.
- VTS automatically determines if an [MQL](#) Element is a **Code Snippet** or a [Prototyped Function](#) when the [MQL Element](#) is saved.
- If the first uncommented line of code contains a *correctly formed function prototype*, the [Element](#) will be saved as a [Prototyped Function](#). Otherwise the [Element](#) will be saved as a **Code Snippet**.

- A *correctly formed function prototype* follows this format:

`return_type` Function_Name (`data_type` Parameter_name1, `data_type` Parameter_name2, ...)

There can be *zero to any number* of parameters.

- This is an example of a function without any parameters:

```
double GetSpread()
{
    return( MathAbs(Bid-Ask) );
}
```

- This is an example of a function with two parameters:

```
int GetSum( int a, int b)
{
    return( a + b );
}
```

- **NOTE:** The behaviour of the configuration (+) button differs if the MQL Element is a **prototyped** function:
 - The first time the [configuration\(+\)](#) button is selected, the editor is in *edit mode* allowing the MQL code to be edited.
 - After the MQL Element is saved, subsequent selections of the [configuration\(+\)](#) button will display the [platform function configuration window](#).
 - This allows the parameters of an MQL function to be set the same way parameters are set for a platform function.
 - The MQL code is shown as [read-only](#) text.
 - The Original MQL function is saved by name in the [Functions Toolbox](#) in the menu: **Advanced -> MQL**
 - The name of the function is the name taken from the [prototype](#), **not** the name of the MQL Element.
 - To edit the original MQL function, drag and drop the function from the **MQL** menu and use the [configuration\(+\)](#) button to open and edit the MQL code.

Summary

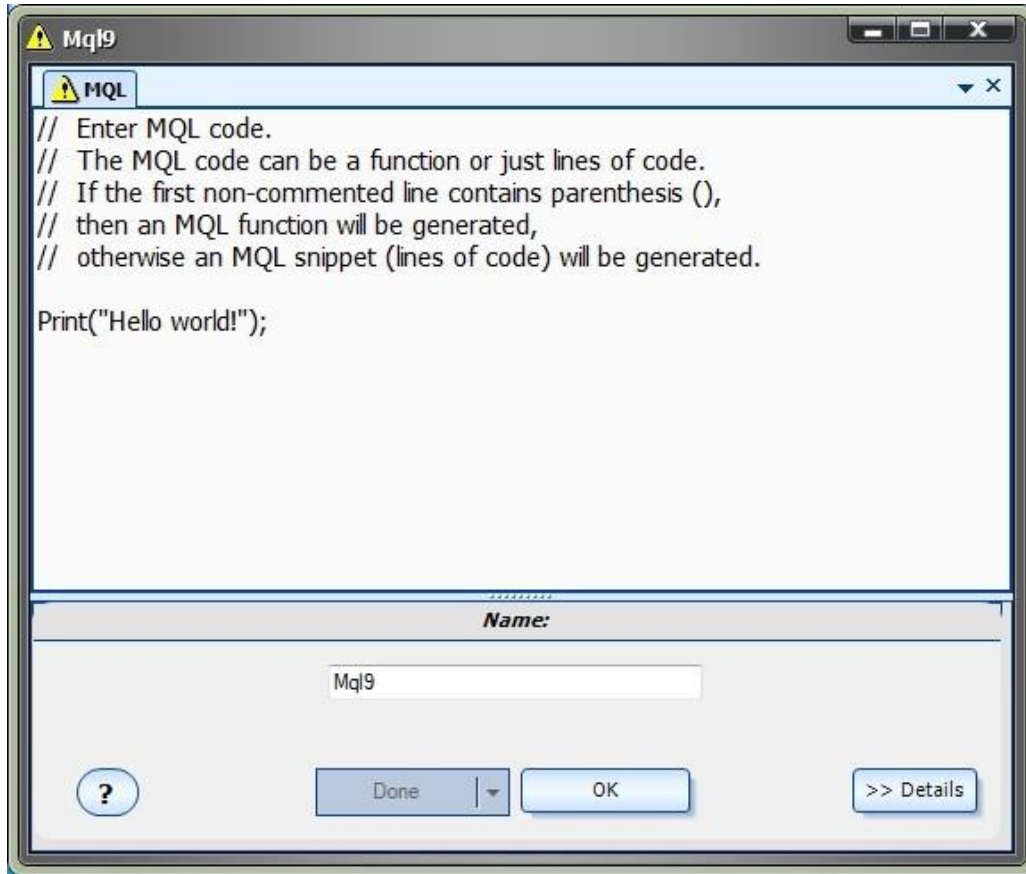
When you create an MQL function, the name of the function within the MQL code will be the name of the function on the **Advanced->MQL** menu.

The name of the MQL Element in which the MQL code was entered can be named any available name, for example *MyMql*.

The function on the **Advanced->MQL** menu holds the MQL source code and can be edited at any time.

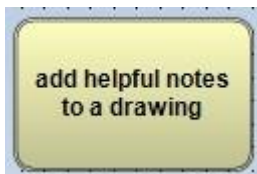
The function *MyMql* represents a call to the function on the **Advanced->MQL** menu. It does not contain the MQL code, only a call to the underlying function with the parameters set specifically.

- **NOTE:** Knowledge of MQL is required to create MQL Elements. [Syntax errors](#) can be injected into the trading system if care is not taking when using native MQL to create MQL Elements.



TEXT Element

- A [TextElement](#) is used to add helpful text to the [Drawing Pad](#).



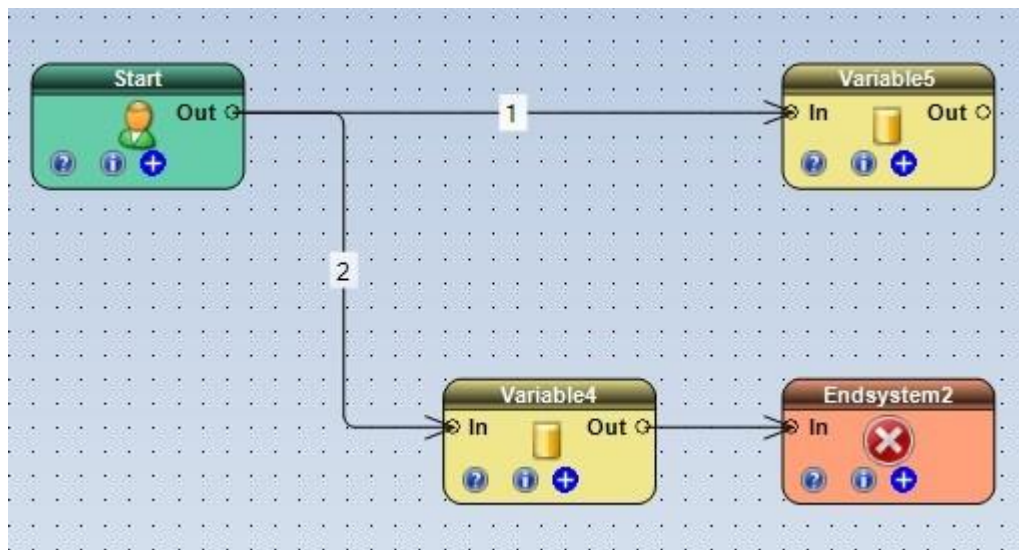
- To add a [TextElement](#) to the [Drawing Pad](#), right-click the mouse and select **Text**. A [TextElement](#) be added at the approximate location of the mouse.



- A [TextElement](#) can be selected, moved and [deleted](#) like other [Elements](#).
- **Note:** The text of a [TextElement](#) will *not* appear within the [MQL](#) code of the [Expert Advisor](#). To add text that will be inserted the MQL code of the Expert Advisor, use the [Note Tab](#) of any [Element](#).

Links

- **Links** are used to connect [Elements](#) on the [Drawing Pad](#).
- **Links** are created by pressing the left button of the mouse within the output area of an [Element's anchor](#), then dragging and releasing the mouse button near the input [anchor](#) of an [Element](#).
- **Links** are selected by pressing the left button of the mouse on the **Link**.
- **Links** are deleted by selecting the **Link** and then pressing the **Delete** key of the keyboard.
- Program execution follows the direction of the **Links**.
- **Links** can be numbered to control the order of execution. Double-click the **Link** to enter a number:



System Managers

System Managers are used to add functionality that does not lend itself to the drag-and-drop paradigm. **System Managers** implement **system-level** rules that would be inconvenient to implement on a [drawing](#).

- **System Managers** are found on the right side of the [VTS](#) application in the [Properties](#) window.

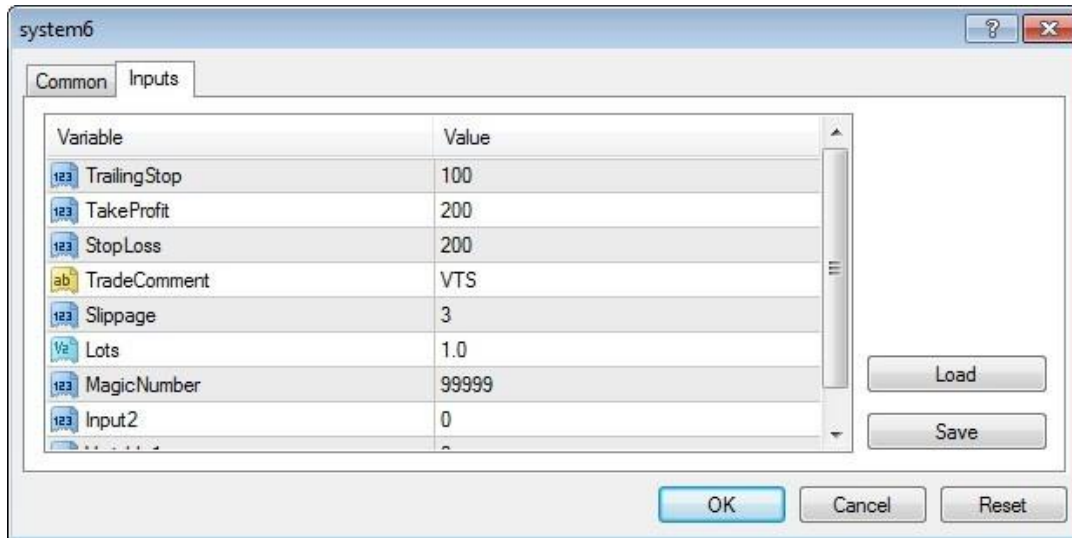


- The following **System Managers** are available:
 - [Input Manager](#)
 - [TradeTime Manager](#)
 - [TradeSignal Manager](#)
 - [Communication Manager](#)
 - [OpenTrade Manager](#)
- A **System Manager** is opened with a single mouse-click.

INPUT Manager

The **Input Manager** is used to add **Inputs** to an [Expert Advisor](#).

When an [Expert Advisor](#) is attached to a price chart, the following **Inputs window** is displayed:

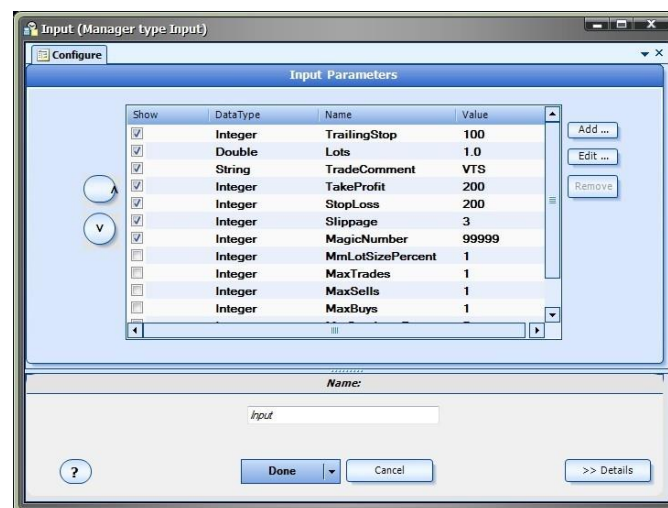


The **Input Manager** is used to add or remove [Variables](#) that appear on this list and are used throughout out the [VTS](#) System.

To open the **Input Manager**, click the **Input Manager** icon in the [System Managers](#) section of the [Properties](#) window.

- The **Add** button is used to add an **Input** variable.
- The **Edit** button is used edit an **Input** variable.
- The **Remove** button is used to remove an **Input** variable.
- The **Show** checkbox is used to *Hide* or *Show* Inputs. Unchecking the **Show** checkbox allows you to hide the **Input** without deleting the [Variable](#).

Note: An **Input** can also be defined by creating a [Variable](#) and setting its [scope](#) to [extern](#). Any variable whose scope is extern will appear on the [Expert Advisor Inputs](#) window.



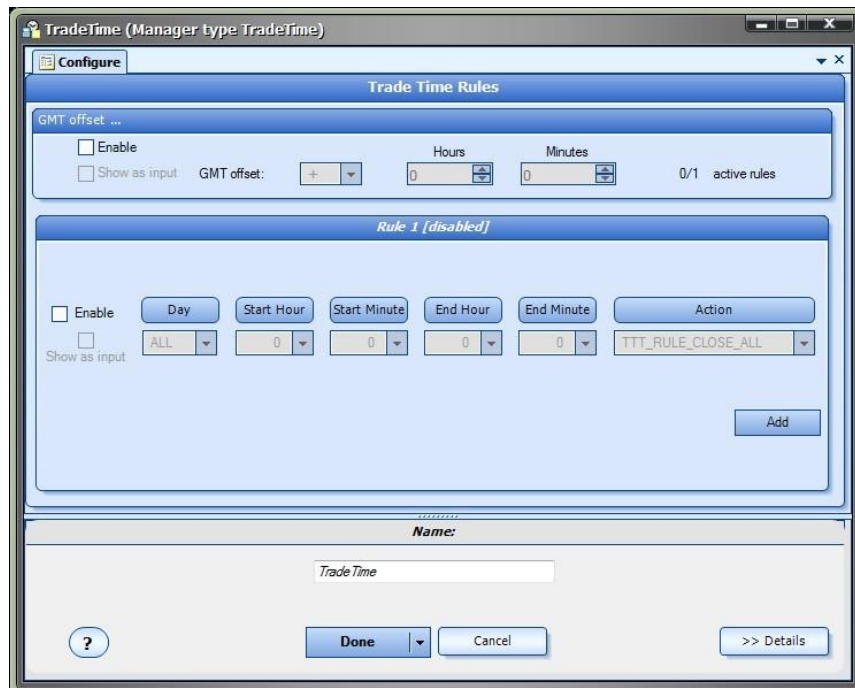
- The bottom portion of the window allows the [System Managers](#) data to be [saved](#).

TRADETIME Manager

The **TradeTime Manager** is used to add time-based rules to an Expert Advisor.

- **Trade Time Rules** are based on the [MetaTrader](#) broker's clock; the **GMT offset** feature is used to synchronize the brokers's time with the local PC running the [Expert Advisor](#).
- **Trade Time Rules** are enabled by checking the **Enable** checkbox
- **Trade Time Rules** are added by clicking the **Add** button. An unlimited number of rules can be added.
- Each **Trade Time Rule** is configured to be active for a specific time interval.
 - The **Day** value can set to **ALL** to run everyday, or set to a specific day of the week.
 - The **Start** and **End** hour values are 24-hour based.
 - The **Show as input** checkbox makes the time entry values available as [Inputs](#) to the Expert Advisor
- The following **Actions** can be set for each **Trade Time Rule**:
 - TTT_RULE_CLOSE_ALL**: close all open trades during this period.
 - TTT_RULE_CLOSE_BUYS**: close all open buy (or long) trades during this period.
 - TTT_RULE_CLOSE_SELLS**: close all open sell (or short) trades during this period.
 - TTT_RULE_OPEN_BUYS**: only allow buy (or long) trades to be opened during this period.
 - TTT_RULE_OPEN_SELLS**: only allow sell (or short) trades to be opened during this period.
 - TTT_RULE_OPEN_NONE**: do not open any new positions during this period.

NOTE: Any time period not explicitly defined by the **TradeTime Manager** has the normal default behavior of opening and closing all trades



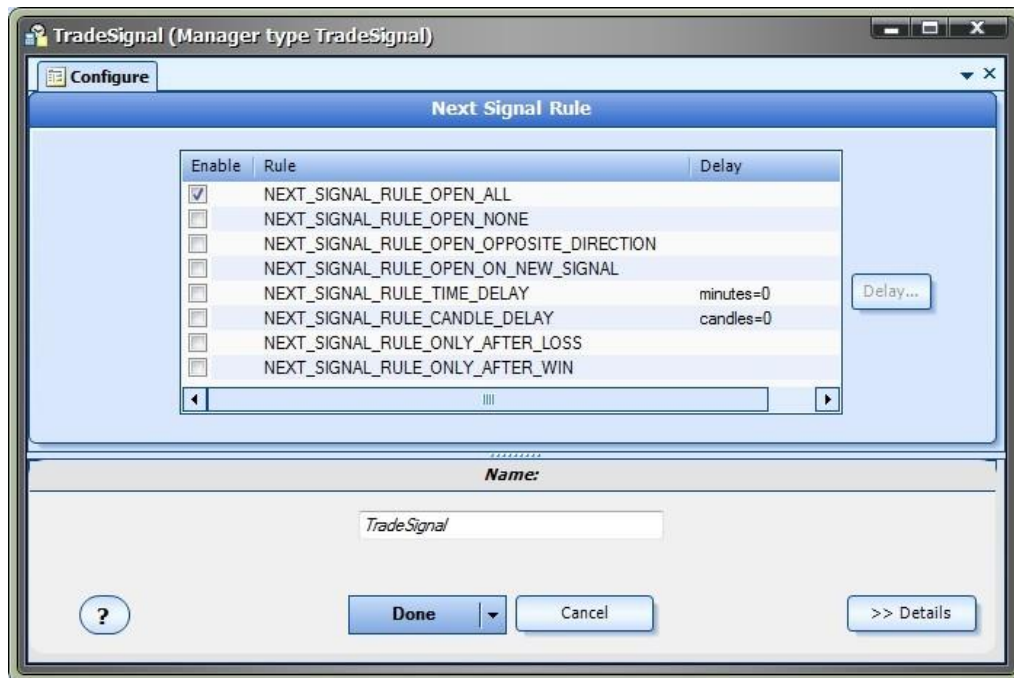
- The bottom portion of the window allows the [System Managers](#) data to be [saved](#).

TRADESIGNAL Manager

The **TradeTime Manager** is used to apply rules to subsequent [trade signals](#) (NEXT_SIGNAL) after an Expert Advisor has opened an initial trade.

- Some **Rules** can be combined with others. If a **Rule** can not be combined with another rule, **Rule** the **Rule** will be disabled.
- The **Delay** button is used to enter a delay period for rules that require a period.
- The following NEXT_SIGNAL **Rules** are available:

NEXT_SIGNAL_RULE_OPEN_ALL	Always open another trade when the next signal is generated. This is the default behaviour.
NEXT_SIGNAL_RULE_OPEN_NONE	Never open another trade when the next signal is generated. The EA will need to be re-attached to the chart to open trades.
NEXT_SIGNAL_RULE_OPEN_OPPOSITE_DIRECTION	Only open a trade in the opposite direction.
NEXT_SIGNAL_RULE_OPEN_ON_NEW_SIGNAL	Only open a trade on a new signal (the original signal must become invalid.)
NEXT_SIGNAL_RULE_TIME_DELAY	Wait a period of time before opening the next trade (in minutes).
NEXT_SIGNAL_RULE_CANDLE_DELAY	Wait a number of candles or bars before opening the next trade
NEXT_SIGNAL_RULE_ONLY_AFTER_LOSS	Only open another trade if the last trade resulted in a loss.
NEXT_SIGNAL_RULE_ONLY_AFTER_WIN	Only open another trade if the last trade resulted in a win (profit).



- The bottom portion of the window allows the [System Managers](#) data to be [saved](#).

COMMUNICATION Manager

The **Communication Manager** is used to send [System](#) level messages from a running [Expert Advisor](#). The messages are sent based upon the occurrence of specified Events.

Note: Messages can also be sent from any [Element](#) using the [Message](#) Tab.

There are two sections of the **Message** Tab, **Event** and **Output**:

Event section:

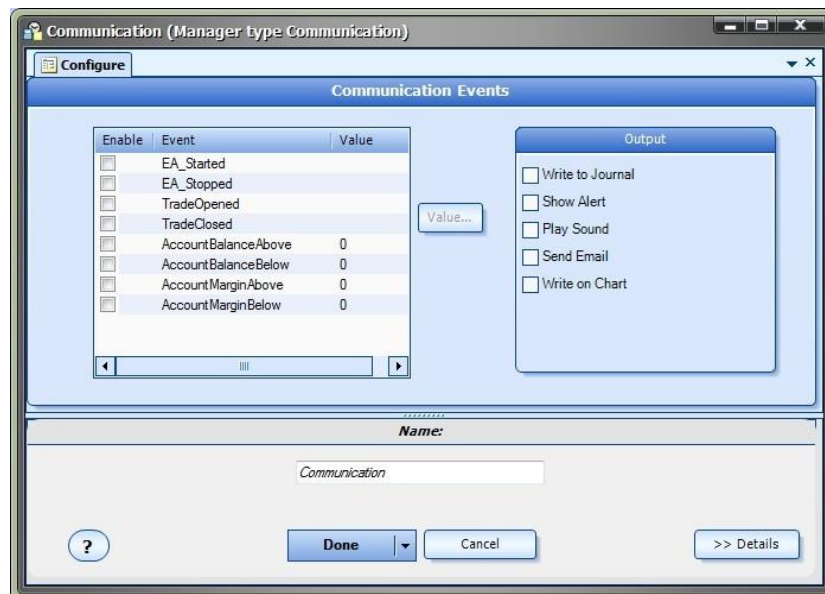
The following **Events** are available:

- **EA_Started** The [Expert Advisor](#) has started (called from the [init](#) function)
- **EA_Stopped** The [Expert Advisor](#) has stopped (called from the [deinit](#) function)
- **TradeOpened** A Trade on the account has opened.
- **TradeClosed** A Trade on the account has closed.
- **AccountBalanceAbove** Account balance threshold. value is entered using the **Value** button.
- **AccountBalanceBelow** Account balance threshold. value is entered using the **Value** button.
- **AccountMarginAbove** Account margin threshold. value is entered using the **Value** button.
- **AccountMarginBelow** Account margin threshold. value is entered using the **Value** button.

Output section:

- The **Write to Journal** checkbox will send the **Message** data to the [Journal](#) tab of the [MetaTrader](#) Platform.
- The **Show Alert** checkbox will display the Alert window on the [MetaTrader](#) Platform.
- The **Play Sound** checkbox will play the specified sound. The sound options are provided from the [MetaTrader](#) Platform sound file.
- The **Send Email** checkbox will send an Email of the data from the [MetaTrader](#) Platform. The [MetaTrader](#) Platform must be configured to send emails.
- The **Write on Chart** checkbox will display the data on the price chart. The **Message** data can be added to other messages on the chart, or it can clear all other message data.

Note: The [Journal](#) Tab is useful for debugging a running EA. Messages can be used to determine if the execution of the EA is following the desired path and if the values of variables are correct.



- The bottom portion of the window allows the [System Managers](#) data to be [saved](#).

OPENTRADE Manager

The **OpenTrade Manager** is used to define [System](#) level values that are used for opening, closing and managing trades.

- Each item in the **OpenTrade Manager** contains:
 - **Show as input** checkbox to allow the value to appear on the [Expert AdvisorInputs](#) window.
 - Text-box for entering a value.
 - [Choose](#) button for selecting an existing value.

There are three Tabs on the **OpenTrade Manager** window:

- **Trade**
 - This tab lists the default values used when opening a new trade.
- **Maximum**
 - This tab list the maximum values used to prevent the [Expert Advisor](#) from opening trades.
- **Management**
 - This tab lists Money Management values used to calculate money management **Lot** and **StopLoss** values:
 - MmStopLossPercent is used to calculate a **stoploss** based on a percentage of the Account Balance.
 - MMLotSizePercent is used to calculate a **lot** size based on a percentage of the Account Balance.

The **Trade** tab:

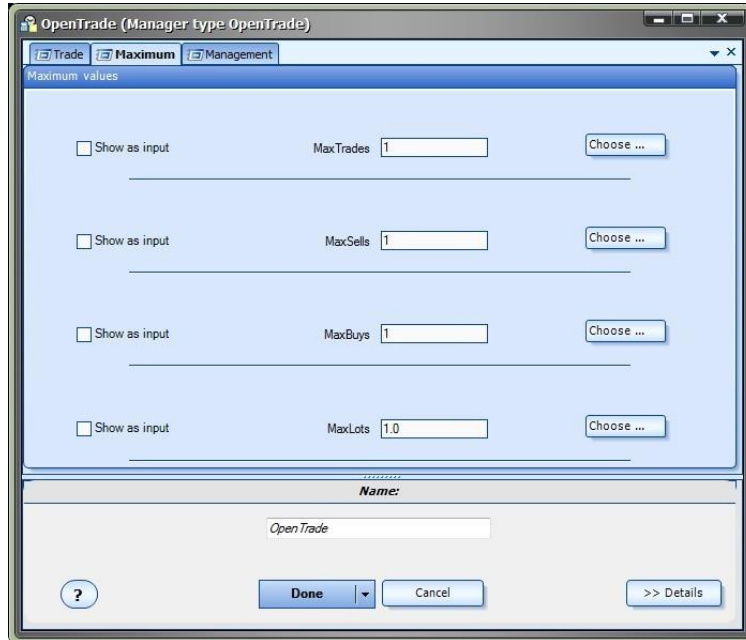


The screenshot shows the 'OpenTrade (Manager type OpenTrade)' window with the 'Trade' tab selected. The window contains a 'Trade values' section with the following fields and controls:

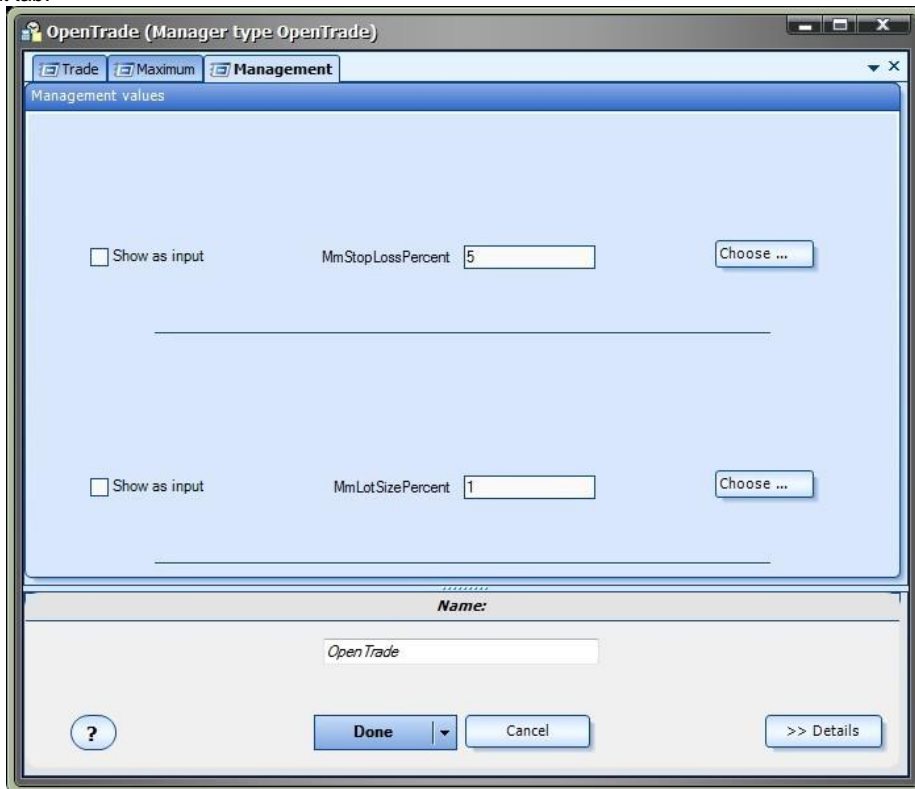
Field	Value	Control
MagicNumber	99999	Choose ...
Slippage	3	Choose ...
StopLoss	200	Choose ...
TakeProfit	200	Choose ...
TradeComment	VTS	Choose ...
Lots	1.0	Choose ...
TrailingStop	100	Choose ...

Below the 'Trade values' section is a 'Name:' field with the value 'Open Trade'. At the bottom of the window are buttons for '?', 'Done', 'Cancel', and '>> Details'.

The Maximum tab:



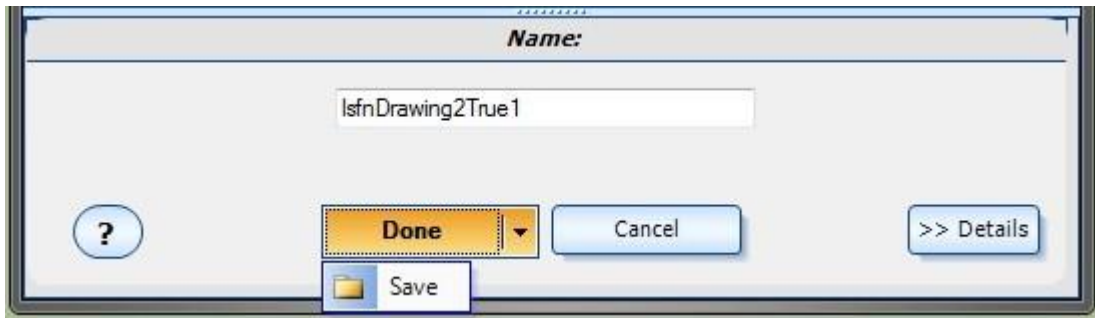
The Management tab:



- The bottom portion of the window allows the [System Managers](#) data to be [saved](#).

Manager Name

The bottom portion of each **System Manager Window** is used for naming and saving the [Manager](#).

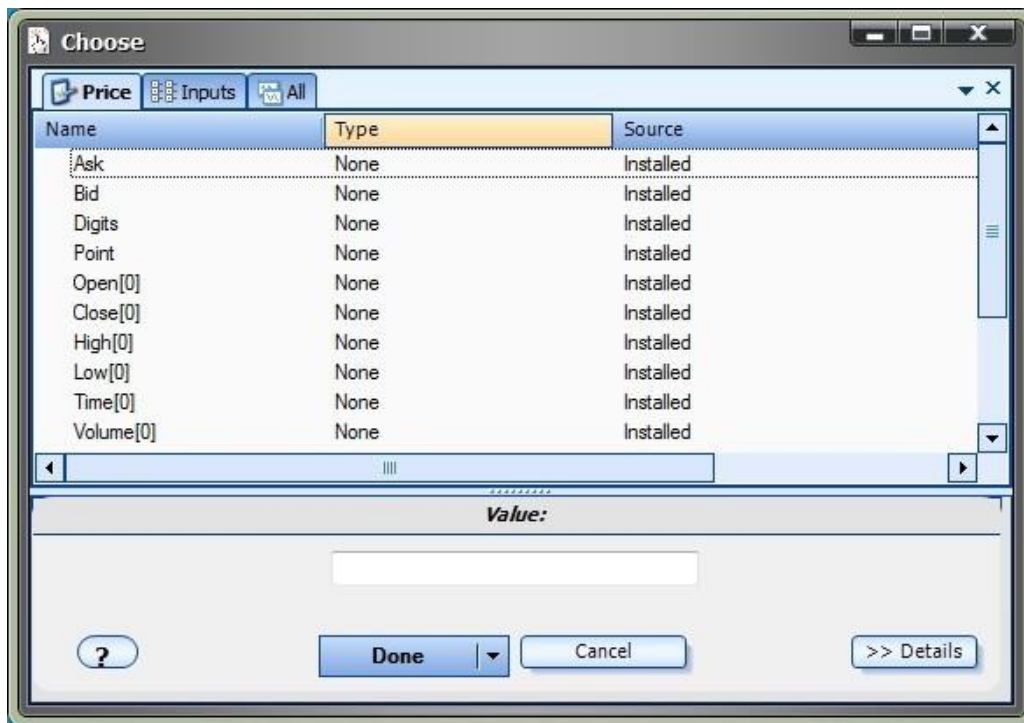


- All [Manager](#) values entered using a System [Manager](#) are saved with a [VTS](#) System.
- [VTS](#) does not yet support saving and loading System [Manager](#) data by name.

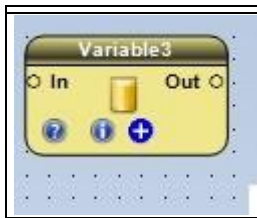
Dialogs

Choose

- The **Choose** dialog is used to select an existing [Elements](#), usually for the value of a [parameter](#) or [variable assignment](#).
- The **Choose** dialog is context-sensitive. It will display different values depending upon where it was selected.
- The **Price Tab** displays built-in [MQL](#) variables.
- The **Input Tab** displays all [VTSVariables](#) whose [scope](#) is defined as [extern](#), or where added using the [Input Manager](#).
- The **All Tab** displays all available [Elements](#).



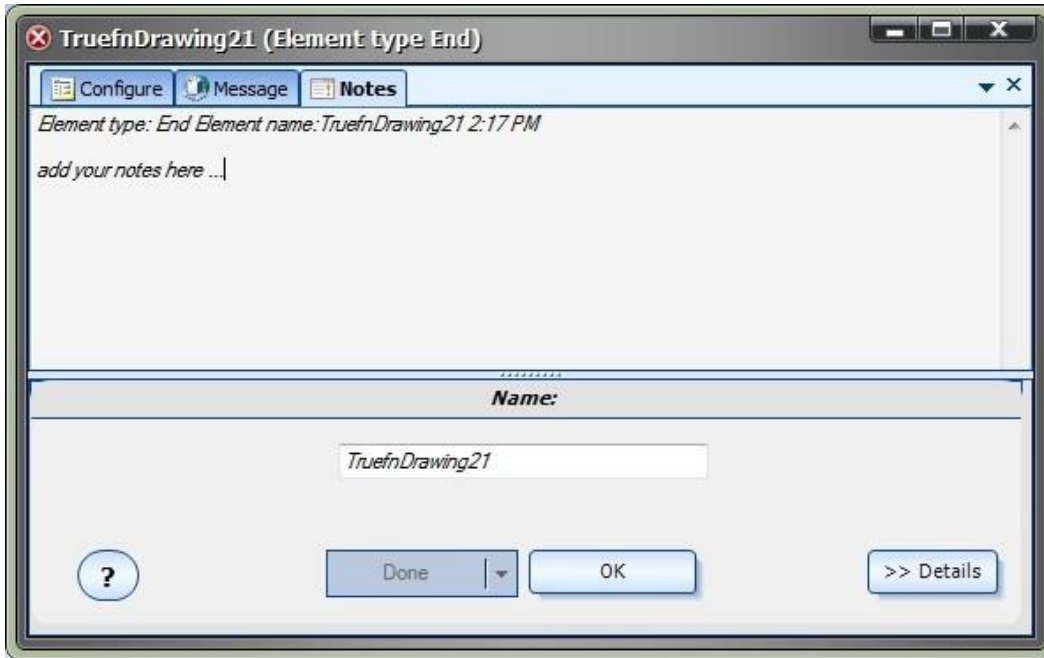
Note



Selecting the configuration (+) button of an Element will display the **Configuration Window**.

The **Note** Tab is found in the **Configuration window**.

The **Note** Tab of an [Element](#) allows text to be entered that will be saved with the [Element](#) and written as a [comment](#) with the [MQL](#) code.



Message

	<p>Selecting the configuration (+) button of an Element will display the Configuration Window.</p> <p>The Message Tab is found in the Configuration window.</p>
--	--

The **Message** Tab of an [Element](#) allows information to be sent from a running Expert Advisor.

There are two sections of the **Message** Tab, **Data** and **Output**:

Data section:

- A **Message** is enabled by selecting the **Send message** checkbox.
- Any text can be entered into the **Message** text-box.
- The **Data** text-box is [read-only](#). The [Choose](#) button is used to add a value to the **Data** text-box.
- The **Write last error** check box can be used to write out the last error that occurred in the running EA.
- [Function Elements](#) display a checkbox to write the value of the function.

Output section:

- The **Write to Journal** checkbox will send the **Message** data to the [Journal](#) tab of the MetaTrader Platform.
- The **Show Alert** checkbox will display the Alert window on the MetaTrader Platform.
- The **Play Sound** checkbox will play the specified sound. The sound options are provided from the MetaTrader Platform sound file.
- The **Send Email** checkbox will send an Email of the data from the MetaTrader Platform. The MetaTrader Platform must be configured to send emails.
- The **Write on Chart** checkbox will display the data on the price chart. The **Message** data can be added to other messages on the chart, or it can clear all other message data.

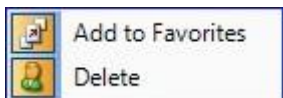
Note: The [Journal](#) Tab is useful for debugging a running EA. Messages can be used to determine if the execution of the EA is following the desired path and if the values of variables are correct.



Favorites

Favorites and Delete

Options to add an [Element](#) to the **Favorite** folder or **Delete** an [Element](#) are available by selecting an [Element](#) from the [Toolbox](#) menu and right-clicking.



- An [Element](#) can be removed from the **Favorite** menu by right-clicking and choosing **Delete**. This removes the [Element](#) from the menu but does not entirely **Delete** the [Element](#) from [VTS](#).
- To permanently **Delete** a [Element](#), select the [Element](#) from any non-**Favorite** menu and choose **Delete**. This will permanently **Delete** the [Element](#) from [VTS](#).

NOTE: A permanently **Deleted**[Element](#) can not be recovered.

Power

Power Tab

- **Power** Tabs offered extended functionality to VTS [Elements](#).
- **Power** Tab functionality is a [VTS](#) add-on. It is not included in basic [VTS](#).

- **Power** Tabs are available on these [Elements](#):
 - [Logic Power Tab](#)
 - [Function Power Tab](#)

Logic Power Tab

The **Logic PowerTab** offers extended functionality to the VTS [LogicElement](#).

- Generally speaking, when a [LogicElement](#) evaluates to **True**, the execution of an Expert Advisor follows a path to perform a specific action, such as open, close or modify a trade.
- The options available from the **Logic Power Tab** allow a trader to apply advanced rules to *how and when* the [LogicElement](#) evaluates to **True**. These rules are outside the typical *greater-than, less-than* evaluation of a [logical condition](#).
- The **Logic Power Tab** offers the following options:
 - **Logic Evaluation Frequency**
 - **Minimum required TRUE count**
 - **Maximum allowed TRUE count**
- **Note:** All times are determined by the MetaTrader Platform's clock, *not* the local clock of the running PC.

Logic Evaluation Frequency

This option defines how *often* a Logic is evaluated to be **True**.

On every new tick	The Logic is checked for True on every incoming tick . This is the default selection.
On every new bar	The Logic is checked for True only on the start of a new bar (or candle). The bar length is defined by the chart to which the EA is attached.
On every new hour	The Logic is checked for True only at the start of every hour.
On every new day	The Logic is checked for True only at the start of every day.

Minimum required TRUE count

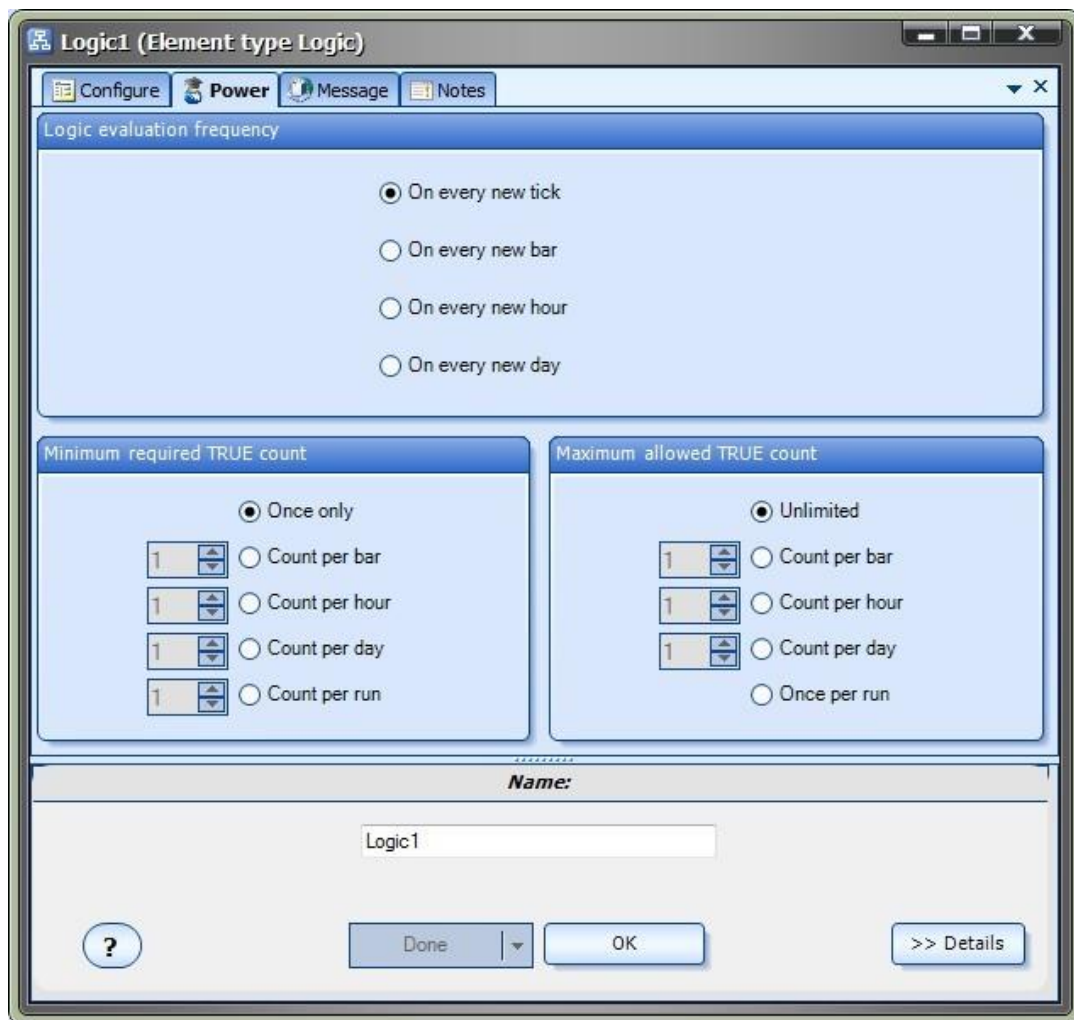
This option defines *how many times* a [Logic](#) must evaluate to **True** before the [Logic](#) returns a **True** value.

Once only	The Logic only needs to evaluate to True once. This is the default selection.
Count per bar	The Count value can be set to any integer above one. The Logic must evaluate to True Count times within a single Bar to return True . The bar length is defined by the chart to which the EA is attached. The running count is reset to zero at the start of a new Bar .
Count per hour	The Count value can be set to any integer above one. The Logic must evaluate to True Count times within an Hour to return True . The running count is reset to zero at the start of a new Hour .
Count per day	The Count value can be set to any integer above one. The Logic must evaluate to True Count times within a Day to return True . The running count is reset to zero at the start of a new Day .
Count per run	The Count value can be set to any integer above one. The Logic must evaluate to True Count times within one Run of the EA. The running count is reset to zero each time the EA is restarted.

Maximum allowed TRUE count

This option defines how many times a [Logic may evaluate](#) to True.

Unlimited	The Logic may return True and unlimited number of times. This is the default selection.
Count per bar	The Logic may return True once per Bar .
Count per hour	The Logic may return True once per Hour .
Count per day	The Logic may return True once per Day .
Once per run	The Logic may return True once per Run .



Function Power Tab

The **Function PowerTab** offers extended functionality to the VTS [Platform FunctionElement](#).

- The **Function Power Tab** offers the following options:
 - **Channel value**
 - **Trending value**
 - **Average value**

Channel value

- A channel is a collection of contiguous price bars (or candles).
- This option gets the **Highest** or **Lowest** value, within a specific channel, of the [platform function](#).
- **Channel type** is defined as **CHANNEL_HIGH** or **CHANNEL_LOW**.
- The **Start candle** is the first candle of the channel.
- The **End candle** is the last candle of the channel.

Note: Candles are numbered from right to left, starting at zero. The *currently forming candle* is candle number zero. See the [shift](#) dialog for a diagram.

Trending value

- A trend is defined when a value moves in a continuous direction, either up or down.
- This option determines if a [platform function](#) is trending or not. A value of one is returned if there is a trend; a value of zero is returned if there is not a trend.
- **Trend type** is defined as **TREND_UP** or **TREND_DOWN**.
- The **Start candle** is the first candle of the channel to be tested for the trend.
- The **End candle** is the last candle of the channel to be tested for the trend.
- The **minimum change** value defines the minimum change in value that must occur for a trend to be confirmed. The [default](#) value is 0.
- The **maximum outlier** value defines how many values within the channel can be against the trend and still confirm the trend. The [default](#) value is 0.

Average value

- This option calculates the moving average of the last number of bars of the [platform function](#).
- **Bars** defines the number of bars that are used to calculate the moving average.
- The selected **Method** defines the method used to calculate the moving average. The options are [SMA](#), [EMA](#), [SMMA](#), [LWMA](#).

iAC1* iAC(CHART,CHART,0)

Configure Power Notes Message

Channel value

Get channel value

Channel type

CHANNEL_HIGH Start candle: 0 End candle: 12

Trending value

Get trend value

Trend type

TREND_UP Start candle: 0 End candle: 12

Minimum change: 0

Maximum outliers: 0

Average value

Get average value

Method

MODE_SMA Bars: 12

Name:

iAC1

? Done Cancel >> Details

Element Name

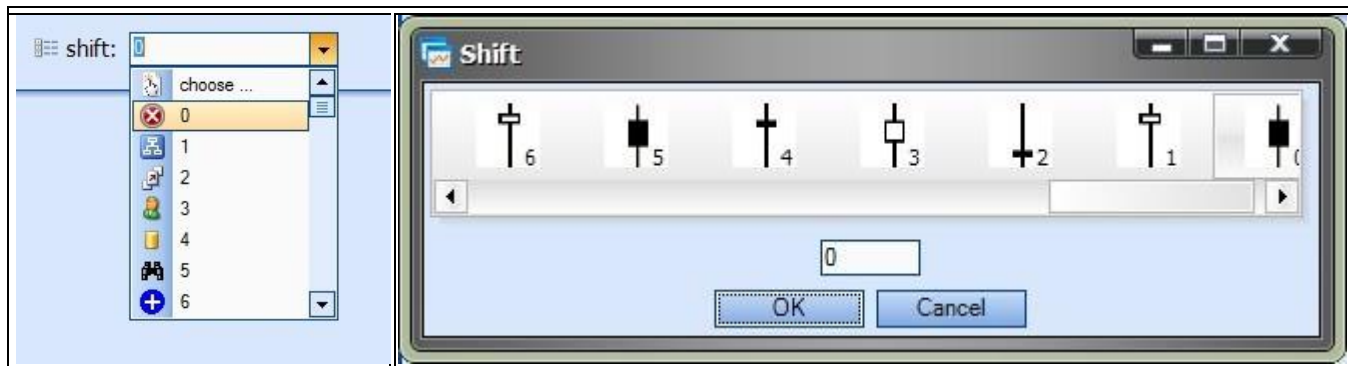
The bottom portion of each *Element Configuration Window* is used for naming and saving the [Element](#).



- Text is entered into the text-box to name the [Element](#).
- If there are no changes to the [Element](#), the **Done** button is disabled and the **Cancel** button is labeled **OK**.
- If there are any changes to the [Element](#):
 - **Done** will save and close the Configuration window;
 - **Save** will store the changes and the window will remain open;
 - **Cancel** will cancel the changes and close the window.
- A malformed, invalid or used name will show an error and will not allow saving.
- The **Question (?)** button will display context-sensitive help.
- The **Details** button will display context-sensitive details if any are available.
- When a [Variable](#), [Logic](#) or [FunctionElement](#) is saved, it is added to the [Toolbox](#) under the System Menu.
- **Note:** *VTS* conveniently generates a name for each [Element](#), *however care should be taken to give each Element a meaningful name.*

Shift

- The [Shift parameter](#) allows the shift value to be chosen using the **Shift** dialog.
- The **Shift** dialog may be viewed when defining a value for the shift parameter.
- From the **Shift** pull-down menu, scroll to the top and select **choose...**
- The **Shift** dialog portrays the shift number of each bars as identified on a price chart: from right to left, starting at 0. The currently forming bar is bar number 0.



Options

The VTS **Options** menu is available from the **Configure** button on the [Main Menu](#) or from **Tools->Option** menu selection.

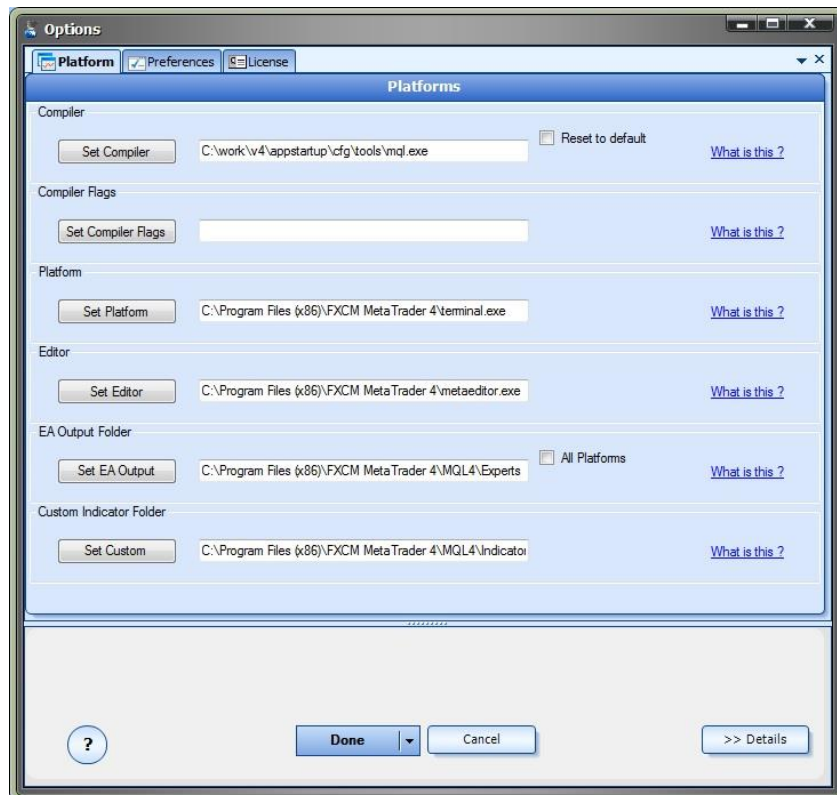
The **Options** menu has the following Tabs:

- [MT Paths](#)
- [Preferences](#)
- [License](#)
- [Easy Email Plug-in \(if enabled\)](#)

MetaTrader Platform Configuration

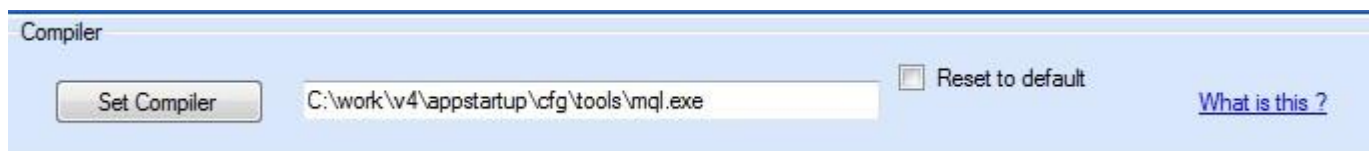
VTS uses the locally installed [MetaTrader](#) platform to build an [Expert Advisor](#).

- **Note:** If these values are not set correctly, VTS will not be able to build an [Expert Advisor](#)
- The following values can be set:
 - **Compiler:** After VTS converts Drawings into readable [MQL](#) code, VTS uses the [MetaTrader](#) compiler to convert the [MQL](#) into an [Expert Advisor](#). The **Build** button on the [Main Menu](#) launches the [MetaTrader](#) compiler.
[Click here for more information.](#)
 - **Compiler Flags:** Allows compiler options to be set.
[Click here for more information.](#)
 - **Editor:** The **Editor** button on the [Main Menu](#) launches the [MetaTrader](#) program *metaeditor.exe*. This program opens the [MQL](#) editor.
[Click here for more information.](#)
 - **Platform:** The **Platform** button on the [Main Menu](#) launches the [MetaTrader](#) program *terminal.exe*. This program opens the [MetaTrader](#) platform.
[Click here for more information.](#)
 - **EA Output Folder:** Defines where EA files are created.
[Click here for more information.](#)
 - **Custom Indicator Folder:** Defines where custom indicators are sourced.
[Click here for more information.](#)
- The **platform** value is also used to locate [Sound Files](#).



Compiler

Compiler Path



When you press the [Build](#) button in [VTS](#), two things happen:

- a. First, the [VTS](#) drawings are converted into MQL code.
- b. Second, the MQL code is "compiled" into an executable Expert Advisor.

- A "compiler" is a program that translates a file of readable text into a format that can be executed by a computer.
- [VTS](#) uses the MetaTrader compiler to build an executable EA.

Prior to MetaTrader build 600, the MetaTrader compiler was named "metalang.exe" and was included in the MetaTrader installation.

After MetaTrader build 600, the MetaTrader compiler was changed to "mql.exe" (or "mql64.exe") and is **not** included in the MetaTrader installation.

The MetaTrader compilers "mql.exe" or "mql64.exe" are included in the VTS installation and the value of the **Compiler Path** is automatically set to:

C:\Program Files (x86)\iExpertAdvisor\Visual Trader Studio Connect\cfg\tools\mql.exe

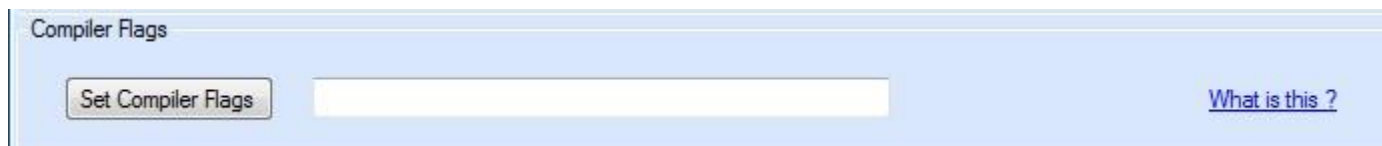
- The **Compiler Path** value is used by VTS to execute the compiler on the MQL code.
- The **Compiler Path** can be set to any valid MQL compiler.

Checking the **Reset to default** check box will set the value to:

C:\Program Files (x86)\iExpertAdvisor\Visual Trader Studio Connect\cfg\tools\mql.exe

Compiler Flags

Compiler Flags



[VTS](#) uses the MetaTrader [compiler](#) to convert your drawings into an Expert Advisor.

Here are the flags that can be passed to compiler:

Usage: mql.exe [<flags>] filename.mq5

```
/mql5 - compile mql5 source
/mql4 - compile mql4 source
/s - syntax check only
/!:<path> - set working directory
```

In general, you do not need to add any compiler flags.

VTS installs the compiler and the required "include" files in the correct locations. If you use the default values for the compiler, you do not need to add any compiler flags.

However, since the MetaTrader compiler can exist anywhere on your computer, it may be necessary to pass **path** information to the compiler to build your EA. For example, this is the correct syntax to find the include files in the VTS folder:

```
-i: C:\Program Files (x86)\iExpertAdvisor\Visual Trader Studio Connect\cfg\tools\MQL4\include
```

Again, **if you use the default values for the compiler, you do not need to add any compiler flags.**

Platform

Platform



[VTS](#) allows you to load the MetaTrader platform into a VTS tab. This makes it easier to navigate between windows, especially during EA development.

This value is the full path to the MetaTrader platform executable file "**terminal.exe**" on your computer.

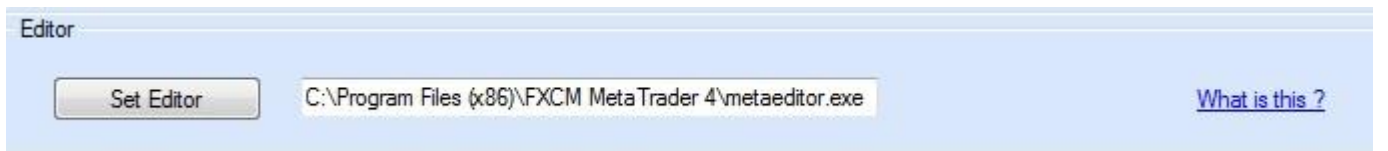
This is an example:

C:\Program Files (x86)\FXCM MetaTrader 4\terminal.exe

Note: When VTS starts, if a MetaTrader platform is not defined, VTS will search and set the first MetaTrader platform found to the **Platform** value. This value can be changed at any time.

Editor

Editor



[VTS](#) allows you to load the MetaTrader editor into a VTS tab. This makes it easier to navigate between windows, especially during EA development.

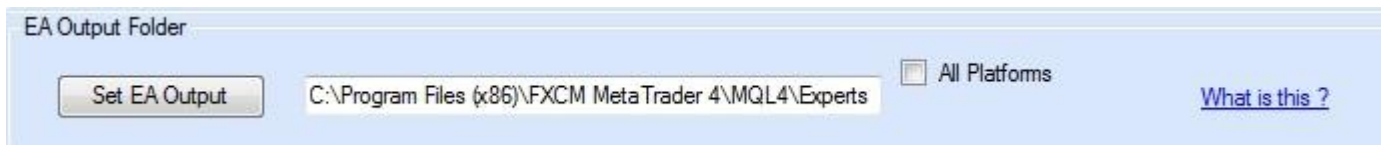
This value is the full path to the MetaTrader platform executable file "**metaeditor.exe**" on your computer.

This is an example:

C:\Program Files (x86)\FXCM MetaTrader 4\metaeditor.exe

Note: When VTS starts, if the MetaTrader editor is not defined, VTS will search and set the first MetaTrader editor found to the **Editor** value. This value can be changed at any time.

EA Output Folder



The **EA Output Folder** is the folder on your computer where [VTS](#) creates the [MQL and EX4 files](#).

Normally, this folder is the "Experts" folder of your installed MetaTrader platform.

For example, if you have FXCM's version of MetaTrader stored at this location:

C:\Program Files (x86)\FXCM MetaTrader 4

Then the location of the **EA Output Folder** would typically be this location:

C:\Program Files (x86)\FXCM MetaTrader 4\MQL4\Experts

When the EA files are placed in this location, they will appear in the MetaTrader platform under the "Experts" tab where they are

available to be attached to a price chart.

To copy your EA to all of your installed MetaTrader platforms, check the **All Platforms** check box.

Note: If your EA builds without errors and does not appear under the "Experts" tab, you may need to restart your MetaTrader platform.

Custom Indicator Folder

Custom Indicator Folder



VTS allows you to drag, drop and connect your.

- The **Custom Indicator Folder** is the folder on your computer where [VTS](#) locates your [Custom Indicators](#) to display in the VTS [Functions Toolbox](#)
- All files with an extension of "ex4" in the **Custom Indicator Folder** are displayed in the VTS [Functions Toolbox](#).

Normally, the **Custom Indicator Folder** is the "Indicators" folder of your installed MetaTrader platform.

For example, if you have FXCM's version of MetaTrader stored at this location:

C:\Program Files (x86)\FXCM MetaTrader 4

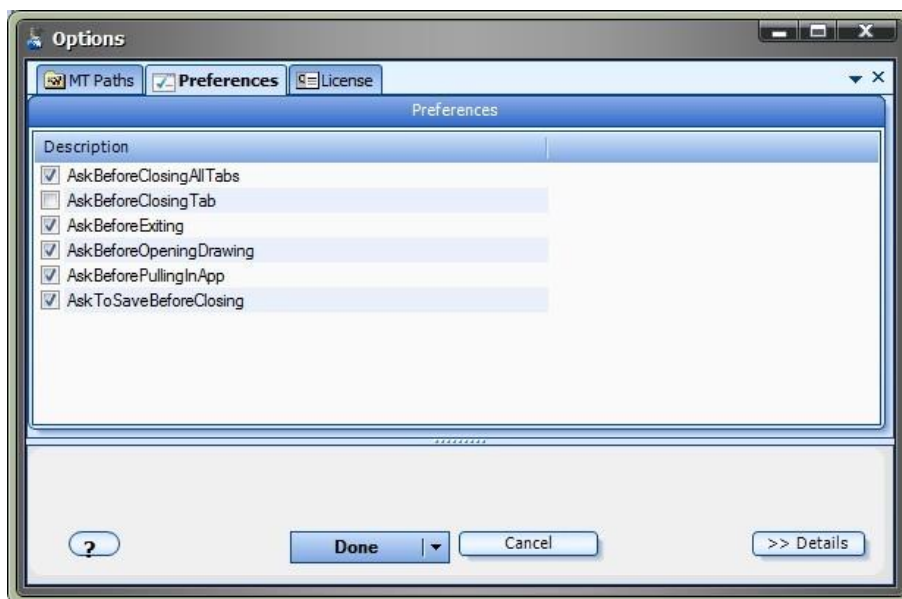
Then the location of the **Custom Indicator Folder** would typically be this location:

C:\Program Files (x86)\FXCM MetaTrader 4\MQL4\Indicators

Note: The [Custom Indicator](#) **MUST** exist in the **"MQL4\Indicators"** folder on the platform where the EA is running.

Preferences

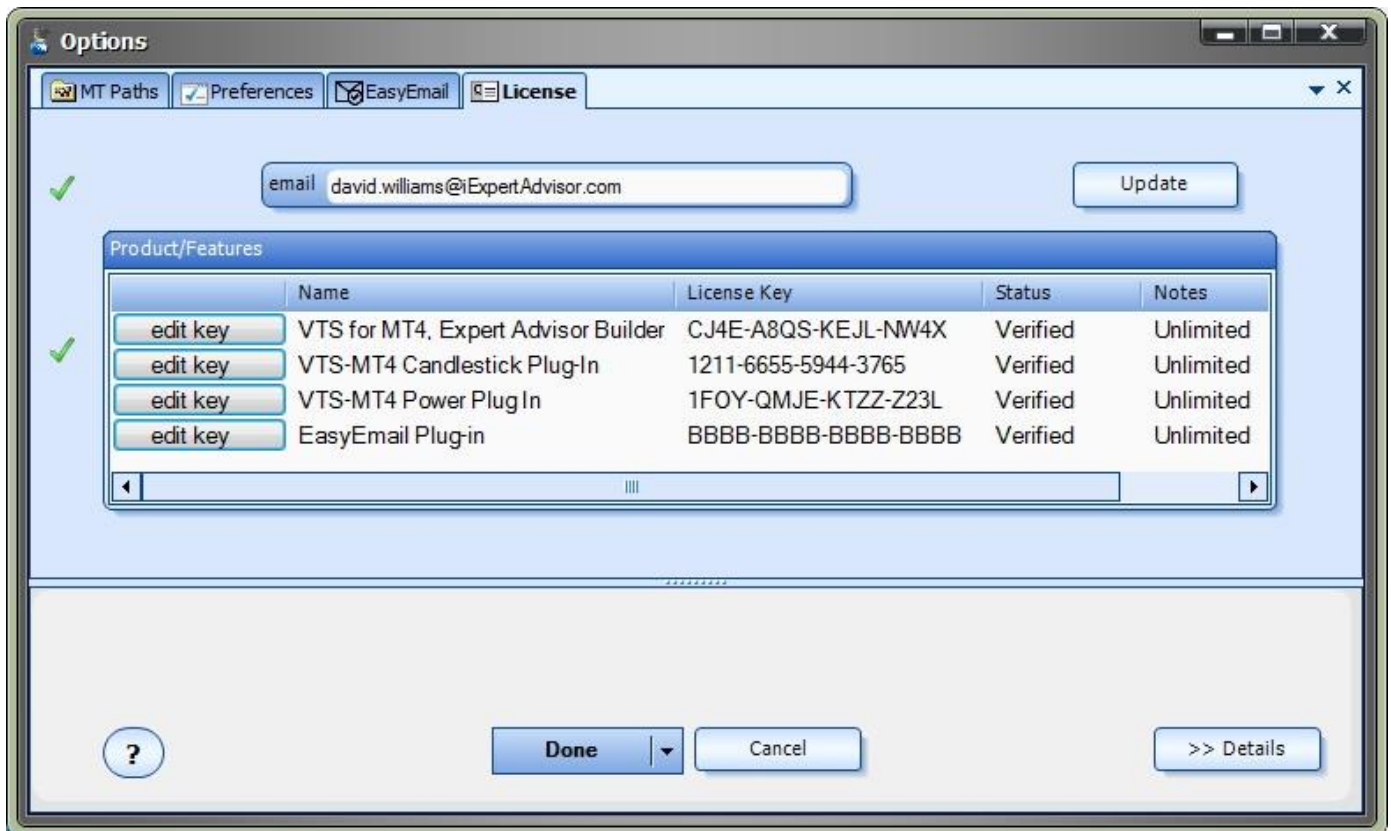
The Preferences allows application-level preferences to be set or reset.



License

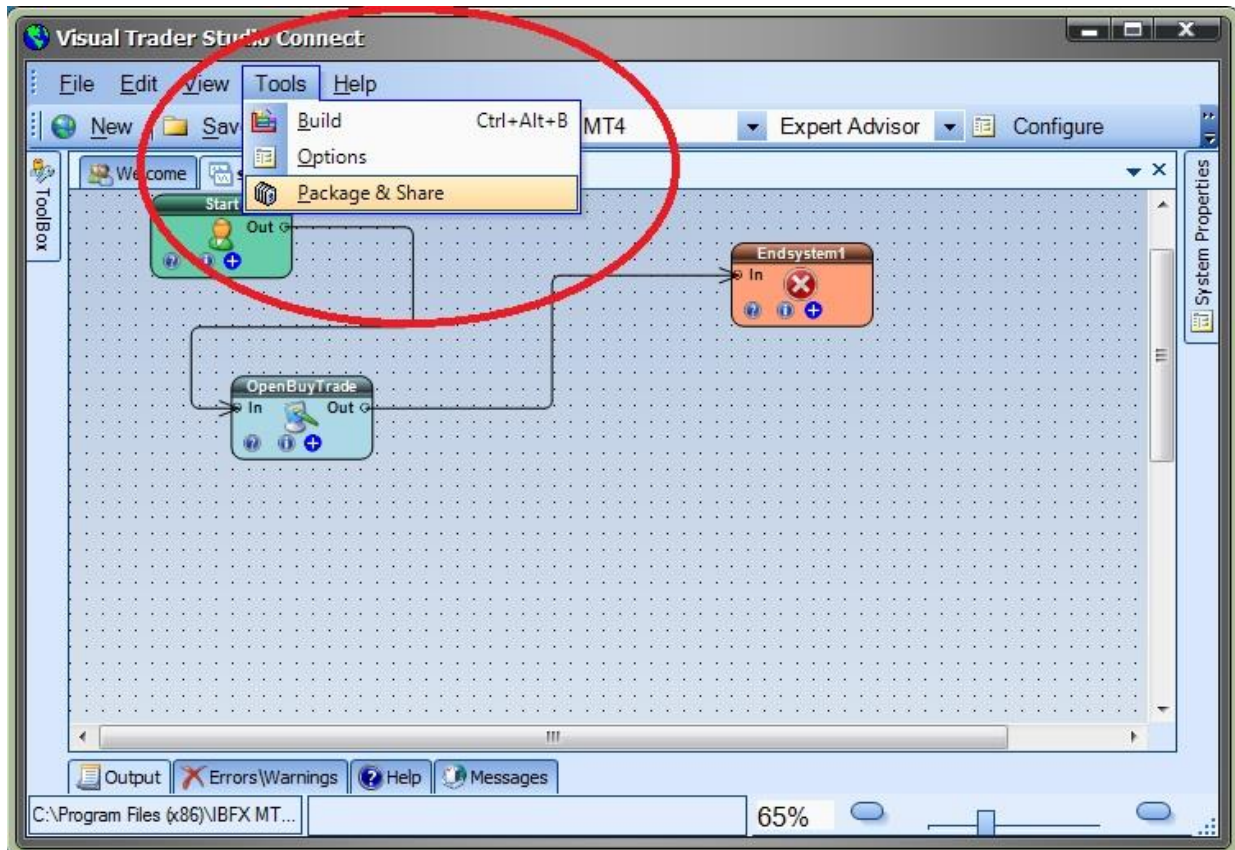
The License tab is used to enter license key information.

- The **email** address is the email address used to purchase [VTS](#).
- The **License Key** is the key listed in the Members Area.
 - A license key is provided for each purchased product (The main EA Builder and each Plug-in)
- The **Update** button is used to verify the email address and license key.
- The **edit key** button is used edit the key value.



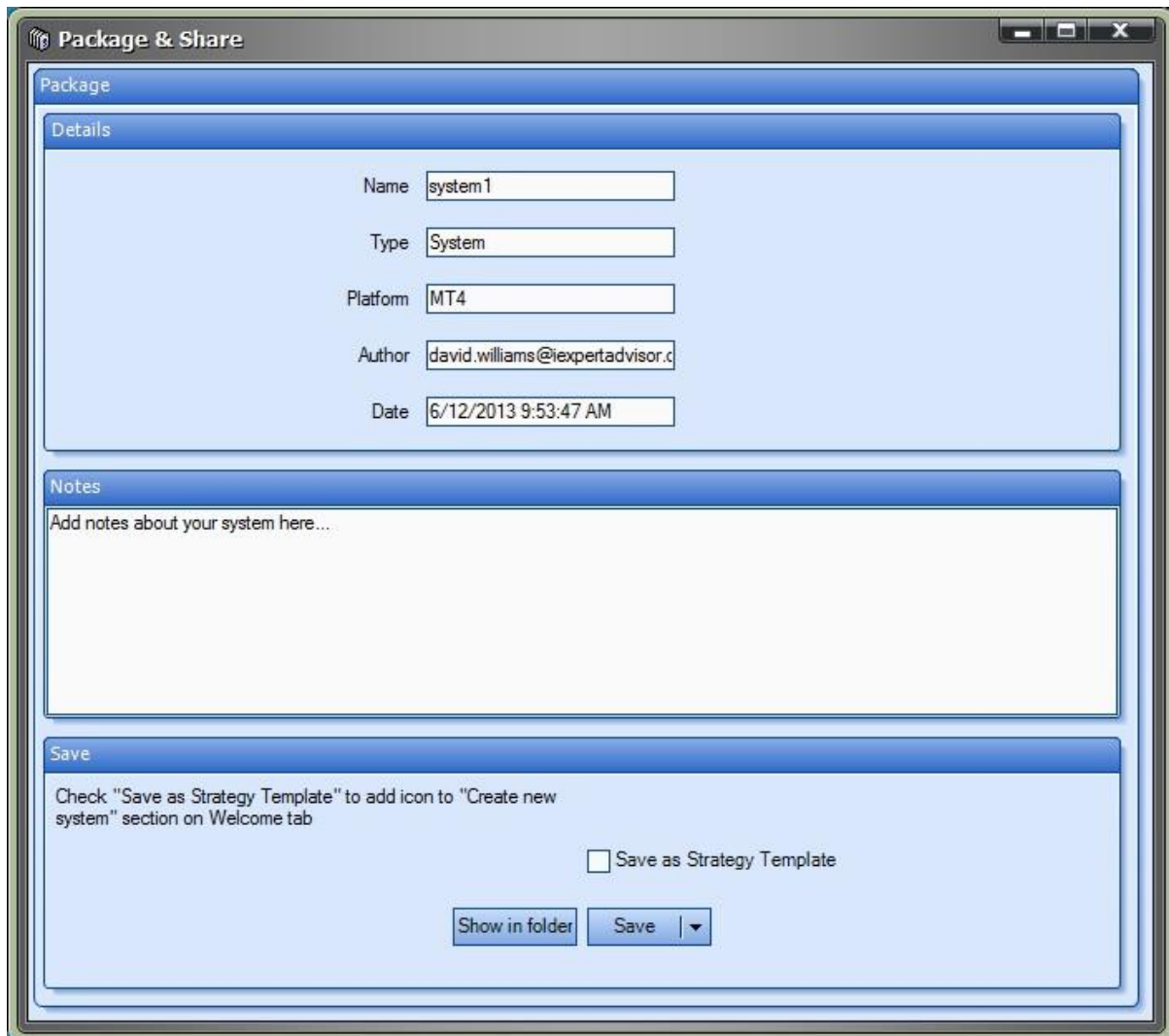
Package & Share

- The **Package & Share** dialog is used to Package an entire VTS System into a proprietary ZIP file.
- In addition, while Packaging a System, it can be configured as a [Strategy Template](#) and will appear on the *Create New System* section of the [Welcome](#) screen.
- The **Package & Share** dialog is opened from the *Tools* menu bar:



Package

The **Package** dialog is used to Package an entire VTS System into a proprietary ZIP file. Each section of the dialog window is described below.



The **Details** section of the **Package** dialog contains the following [Read-Only](#) items:

- **Name** The base name of the VTS [System](#).
- **Type** The type of Package (System, Template, Library)
- **Platform** The platform (MT4, MT5)
- **Author** The author of the System.
- **Date** The last saved date of the [System](#).

The **Notes** section of the **Package** is used to enter details about the Trading System.

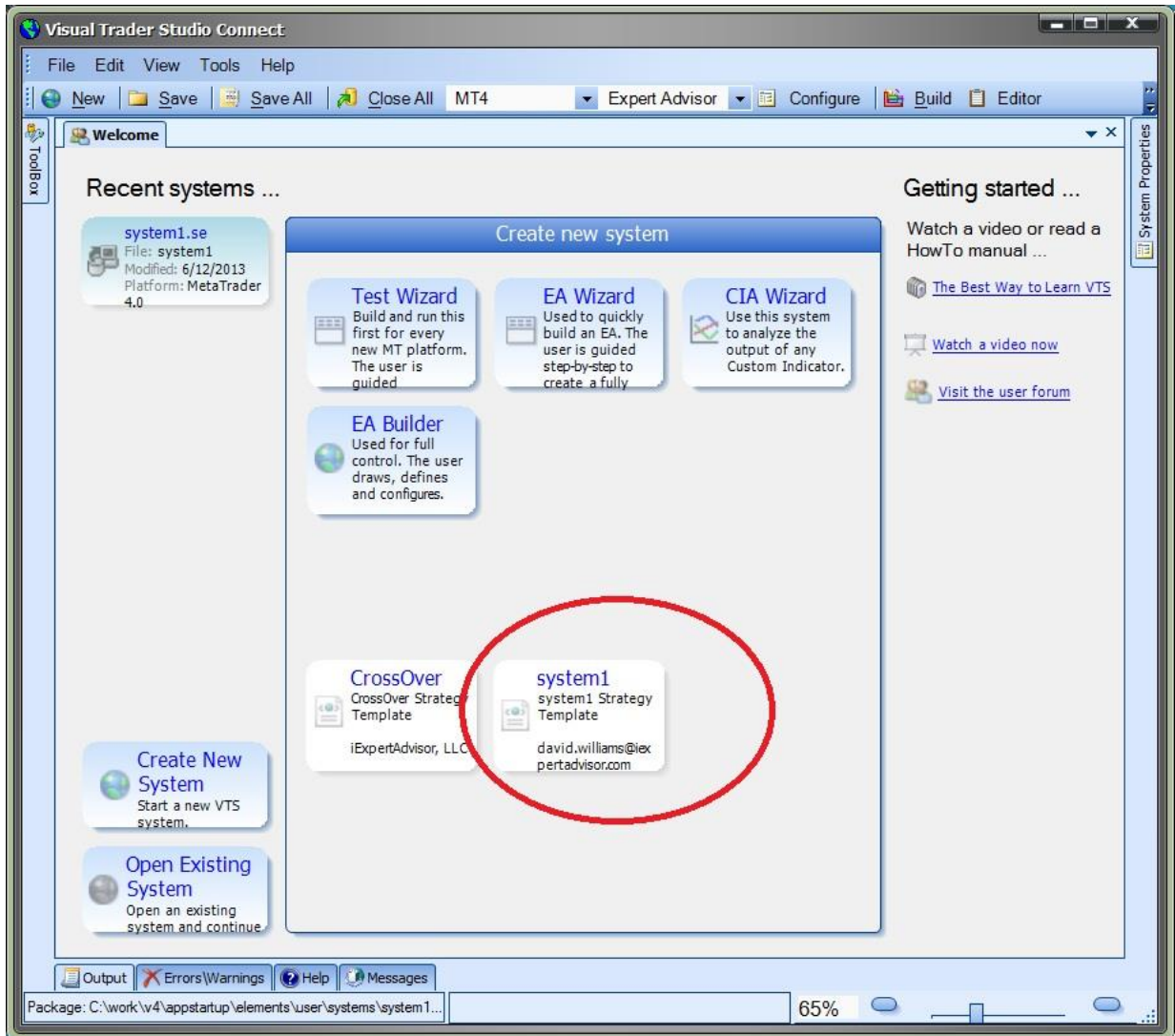
The **Save** section of the **Package** dialog contains the following items:

- **Save as Strategy Template** This check box will save the System as a [Strategy Template](#) and it will appear on the [Welcome](#) screen in the *Create New System* section. (See image below)
- **Show in folder** This button will open a *Windows File Explorer* window where the **Package** file is located.
- **Save** This will save the System as a **Package**.
- **Save->Cancel** This will cancel any non-saved changes and close the **Package** window.

- **Save->Save & Close**

This will save changes and close the **Package** window.

The **System1Strategy Template** on the **Welcome** screen in the **New System** section:



Order Selection

The **Order Selection** section is found on several configuration windows throughout the VTS application.

The **Order Selection** section allows you to define exactly what orders (or *Trades*, or *Positions*) are managed or monitored by the Function or Manager that is being configured.

- The checkbox **All** will manage all open positions on the account and will disable all other check boxes,
- The checkbox **Exact Ticket** will manage only a single position whose ticket matches the ticket value and will disable all other check boxes.
- The remaining check boxes, **Symbol**, **Order Type** and **MagicNumber**, can be used individually or in combination to manage the exact set of orders that you wish.



Order Selection	Determines what orders of the Account are managed or monitored. <ul style="list-style-type: none">• All: all open orders on the account.• Symbol: only open orders matching the selected symbol.• Order Type: only open orders of the selected order type.• MagicNumber: only open orders with a matching magicnumber.• Exact Ticket: only open orders with the matching order ticket. Note: Selecting Exact Ticket disables all other selection criteria.
------------------------	---

Special Functions

VTS provides the following special functions to ease [Expert Advisor](#) development.

- [fnOpenOrder](#) This function it used to open a new order (position).
- [fnModifyOrder](#) This function is used to modified an existing open order (position).
- [fnCloseOrder](#) This function it used to fully or partially close an existing open order (position).
- [fnTrailingStop](#) This function it used to place a trailing stop on an existing open order (position).
- [fnGetOrderInfo](#) This function is used to get information about an existing open order (position).

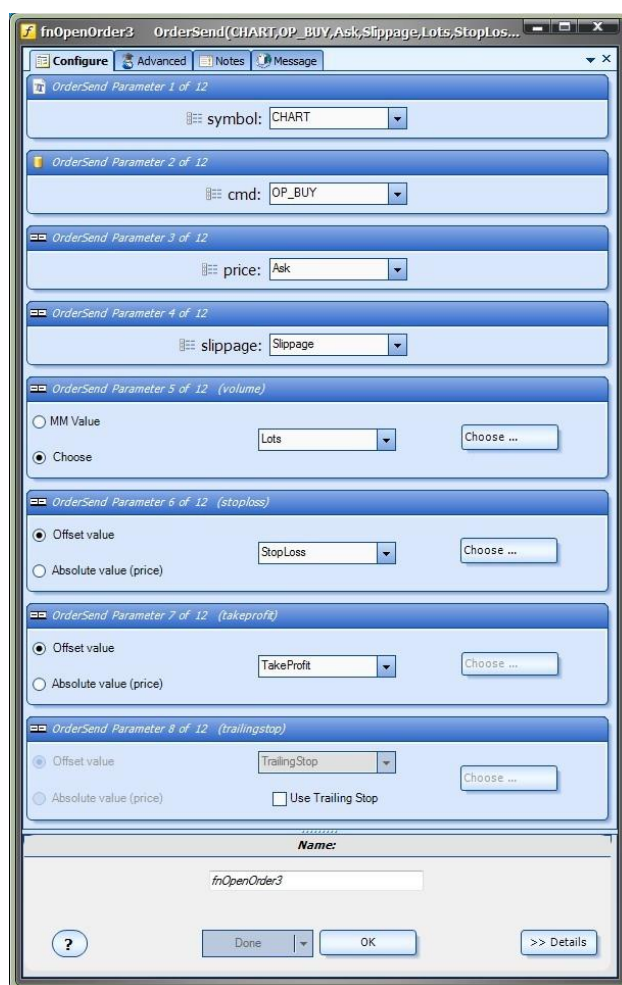
fnOpenOrder

- The **fnOpenOrder** function is used to open a new order (position).
- The **fnOpenOrder** function is found in the [Functions Toolbox](#) under the **Trade** menu.
- The **fnOpenOrder** function is a user-friendly front end for the [MQL](#) function **OrderSend**.
- Selecting the configuration (+) button of **fnOpenOrder** function will display the **fnOpenOrder Configuration Window**.
- There are four tabs on the **fnOpenOrder Configuration Window**:
 - The [Configure](#) Tab is used to set the values of the [parameters](#) of the Function.
 - The [Advanced](#) Tab is used to set the values of the advanced [parameters](#) of the Function.
 - The [Message](#) Tab is used to send a message from the Function Element.
 - The [Notes](#) Tab is used to add notes or comments to the Function Element.

The following table provides information about each [parameter](#) of the **fnOpenOrder** function.

symbol	The symbol name of the currency. CHART specifies the the currency of the chart to which the EA is attached.
cmd	<p>The type of order:</p> <ul style="list-style-type: none"> • OP_BUY: a market buy, or long, position • OP_SELL: a market sell, or short, position • OP_BUYLIMIT: a buy limit order • OP_SELLLIMIT: a sell limit order • OP_BUYSTOP: a buy stop order • OP_SELLSTOP: a sell stop order
price	<p>The price at which to enter the order.</p> <p>MetaTrader usually requires Buy orders to use the Ask price and Sell orders to use the Bid price. These are the VTSdefault values for the price parameter.</p>
slippage	<p>The amount of points, as an integer, the price can move from the requested open price.</p> <p>VTS creates an extern (or Input) variable named Slippage that can be used for this parameter: its default value is 3.</p>
lots	<p>The number of lots to open.</p> <ul style="list-style-type: none"> • VTS creates an extern (or Input) variable named Lots that can be used for this parameter: its default value is 1.0. • A value can be manually entered into the text-box. • A value can be selected using the Choose button. • A Money Management value can be entered by selecting MM Value, a percentage and an Account choice. This will dynamically calculate a lot size using the selected information. <p>A Money Management value can be entered by selecting the Choose button, selecting the MoneyMgt tab, and selecting the value MmLotSizePercentage. The variable MmLotSizePercentage is an extern (or Input) variable that can be entered in the Inputs window when attaching the Expert Advisor to a chart. This will dynamically calculate a lot size using the Account Balance and the value of MmLotSizePercentage</p>
stoploss	<p>The value at which to close the trade for a loss.</p> <ul style="list-style-type: none"> • VTS creates an extern (or Input) variable named StopLoss that can be used for this parameter: its default value is 200. • A value can be manually entered into the text-box. • A value can be selected using the Choose button. • A value can be entered as an integer or as a price level by selecting the Offset value or Absolute value button. The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].
takeprofit	<p>The value at which to close the trade for a profit.</p> <ul style="list-style-type: none"> • VTS creates an extern (or Input) variable named TakeProfit that can be used for this parameter: its default value is 200. • A value can be manually entered into the text-box. • A value can be selected using the Choose button. • A value can be entered as an integer or as a price level by selecting the Offset value or Absolute value button. The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].
trailingstop	The trailing stop option is enabled by checking the Use Trailing Stop checkbox.

	<ul style="list-style-type: none"> • The trade is not trailed until the trade is profitable. • VTS creates an extern (or Input) variable named TrailingStop that can be used for this parameter; its default value is 100. • A value can be manually entered into the text-box. • A value can be selected using the Choose button. • A value can be entered as an integer or as a price level by selecting the Offset value or Absolute value button. The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].
comment	A string of text that is assigned to the order. This text is displayed in the Trade tab of the MetaTrader platform. Note: The MetaTrader broker may change this text when the order is modified.
magicnumber (advanced)	The unique integer number used to identify an open trade. <ul style="list-style-type: none"> • VTS creates an extern (or Input) variable named MagicNumber that can be used for this parameter; • its default value is 99999.
expiration (advanced)	A datetime value that causes the order to expire. Only used for non- market orders.
arrow_color (advanced)	The color of the arrow drawn on the price chart.



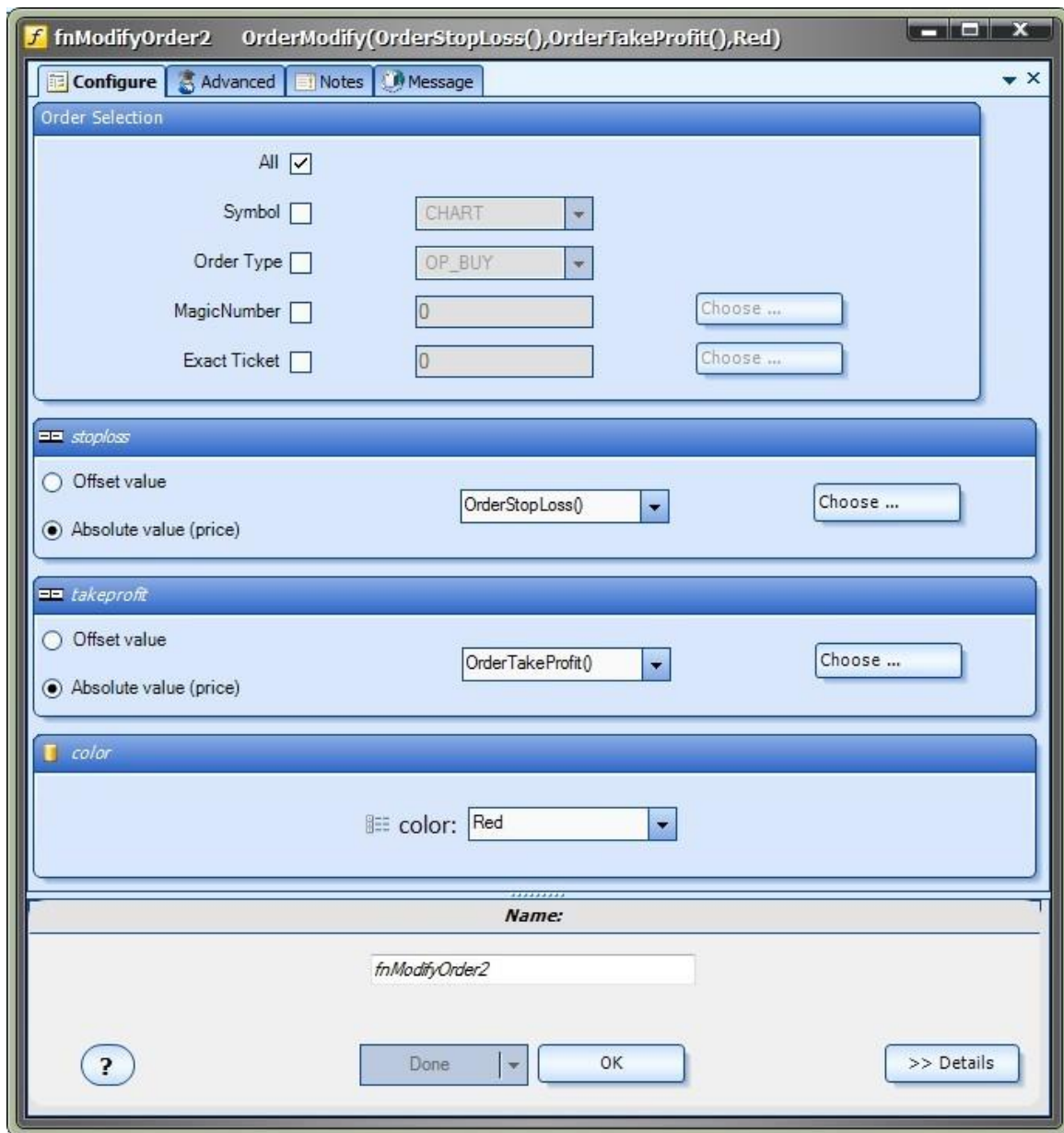
The bottom portion of the window allows the [Element Name](#) to be saved.

fnModifyOrder

- The **fnModifyOrder** function is used to modify the stoploss or takeprofit of an open order (position).
- The **fnModifyOrder** function is found in the [Functions Toolbox](#) under the **Trade** menu.
- The **fnModifyOrder** function is a user-friendly front end for the [MQL](#) function **OrderModify**.
- Selecting the configuration (+) button of **fnModifyOrder** function will display the **fnModifyOrder Configuration Window**.
- There are four tabs on the **fnModifyOrder Configuration Window**:
 - The [Configure](#) Tab is used to set the values of the [parameters](#) of the Function.
 - The [Advanced](#) Tab is used to set the values of the advanced [parameters](#) of the Function.
 - The [Message](#) Tab is used to send a message from the Function Element.
 - The [Notes](#) Tab is used to add notes or comments to the Function Element.

The following table provides information about each [parameter](#) of the **fnModifyOrder** function.

Order Selection	<p>Determines what orders of the Account are modified.</p> <ul style="list-style-type: none"> ● All: all open orders on the account. ● Symbol: only open orders matching the selected symbol. ● Order Type: only open orders of the selected order type. ● MagicNumber: only open orders with a matching magicnumber. ● Exact Ticket: only open orders with the matching order ticket. <p>Note: Selecting Exact Ticket disables all other selection criteria.</p>
stoploss	<p>The value at which to close the trade for a loss.</p> <ul style="list-style-type: none"> ● VTS creates an extern (or Input) variable named StopLoss that can be used for this parameter: its default value is 200. ● A value can be manually entered into the text-box. ● A value can be selected using the Choose button. ● A value can be entered as an integer or as a price level by selecting the Offset value or Absolute value button. The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].
takeprofit	<p>The value at which to close the trade for a profit.</p> <ul style="list-style-type: none"> ● VTS creates an extern (or Input) variable named TakeProfit that can be used for this parameter: its default value is 200. ● A value can be manually entered into the text-box. ● A value can be selected using the Choose button. ● A value can be entered as an integer or as a price level by selecting the Offset value or Absolute value button. The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].
comment	<p>A string of text that is assigned to the order. This text is displayed in the Trade tab of the MetaTrader platform. Note: The MetaTrader broker may change this text when the order is modified.</p>
color	<p>The color of the arrow drawn on the price chart.</p>
price	<p>New open price of the pending order. Only used for non-market orders.</p>
expiration	<p>A datetime value that causes the order to expire. Only used for non-market orders.</p>



The bottom portion of the window allows the [Element Name](#) to be saved.

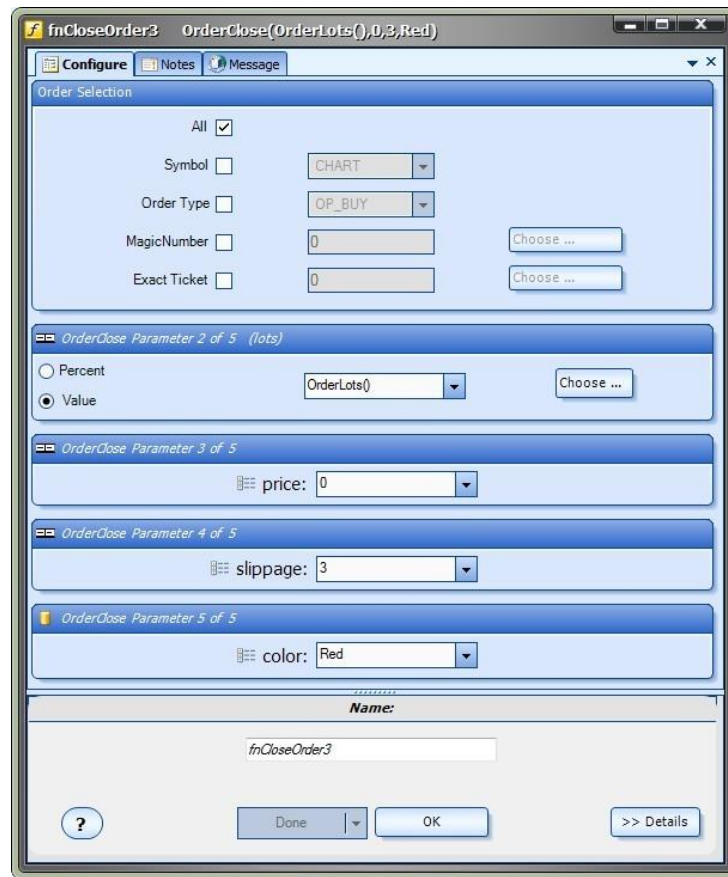
fnCloseOrder

- The **fnCloseOrder** function is used to fully or partially close an open order (position).
- The **fnCloseOrder** function is found in the [Functions Toolbox](#) under the **Trade** menu.
- The **fnCloseOrder** function is a user-friendly front end for the [MQL](#) function **OrderClose**.
- Selecting the configuration (+) button of **fnCloseOrder** function will display the **fnCloseOrder Configuration Window**.
- There are three tabs on the **fnCloseOrder Configuration Window**:

- The [Configure](#) Tab is used to set the values of the [parameters](#) of the Function.
- The [Message](#) Tab is used to send a message from the Function Element.
- The [Notes](#) Tab is used to add notes or comments to the Function Element.

The following table provides information about each [parameter](#) of the *fnModifyOrder* function.

Order Selection	<p>Determines what orders of the Account are closed.</p> <ul style="list-style-type: none"> ● All: all open orders on the account. ● Symbol: only open orders matching the selected symbol. ● Order Type: only open orders of the selected order type. ● MagicNumber: only open orders with a matching magicnumber. ● Exact Ticket: only open orders with the matching order ticket. <p>Note: Selecting Exact Ticket disables all other selection criteria.</p>
lots	<p>The number of lots to close.</p> <ul style="list-style-type: none"> ● VTS creates an extern (or Input) variable named Lots that can be used for this parameter; its default value is 1.0. ● A value can be manually entered into the text-box. ● A value can be selected using the Choose button. ● The value can be entered as a percentage by selecting the Percent button. For example, if the original size of the order is 2 lots, selecting 50% will close 1 lot.
price	<p>The price at which to enter the order. MetaTrader usually requires Sell orders to use the Ask price and Buy orders to use the Bid price.</p>
slippage	<p>The amount of points, as an integer, the price can move from the requested close price. VTS creates an extern (or Input) variable named Slippage that can be used for this parameter; its default value is 3.</p>
color	<p>The color of the arrow drawn on the price chart.</p>



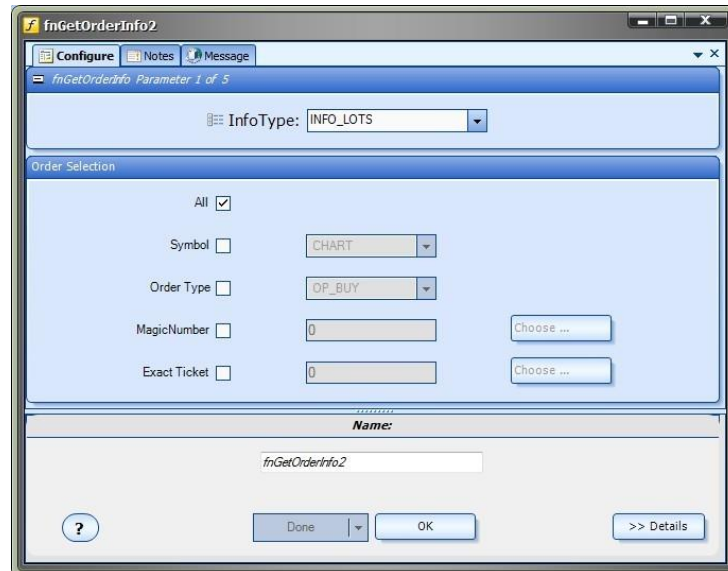
The bottom portion of the window allows the [Element Name](#) to be saved.

fnGetOrderInfo

- The *fnGetOrderInfo* function is used to get information about an open order.
- The *fnGetOrderInfo* function is found in the [Functions Toolbox](#) under the **Trade** menu.
- Selecting the configuration (+) button of *fnGetOrderInfo* function will display the *fnGetOrderInfo Configuration Window*.
- There are three tabs on the *fnGetOrderInfo Configuration Window*:
 - The [Configure](#) Tab is used to set the values of the [parameters](#) of the Function.
 - The [Message](#) Tab is used to send a message from the Function Element.
 - The [Notes](#) Tab is used to add notes or comments to the Function Element.

The following table provides information about each [parameter](#) of the *fnGetOrderInfo* function.

InfoType	<p>The type of information requested (that matches the Order Selection criteria):</p> <p>INFO_LOTS: Get the lot size of an open order. INFO_MAGICNUMBER : Get the magic number of an open order. INFO_OPENPRICE : Get the lot size of an open order. INFO_OPENTIME : Get the open time of an open order. INFO_PROFIT : Get the profit of an open order (this value can be negative) INFO_STOPLOSS : Get the stoploss of an open order. INFO_TAKEPROFIT : Get the takeprofit of an open order. INFO_SWAP : Get the swap value of an open order. INFO_TICKET : Get the ticket of an open order. INFO_ORDERTYPE : Get the order type of an open order. INFO_TOTALORDERS : Get the total number of orders that match the selection criteria.</p>
Order Selection	<p>Determines what orders of the Account are queried.</p> <ul style="list-style-type: none"> • All: all open orders on the account. • Symbol: only open orders matching the selected symbol. • Order Type: only open orders of the selected order type. • MagicNumber: only open orders with a matching magicnumber. • Exact Ticket: only open orders with the matching order ticket. <p>Note: Selecting Exact Ticket disables all other selection criteria.</p>



The bottom portion of the window allows the [Element Name](#) to be saved.

fnTrailingStop

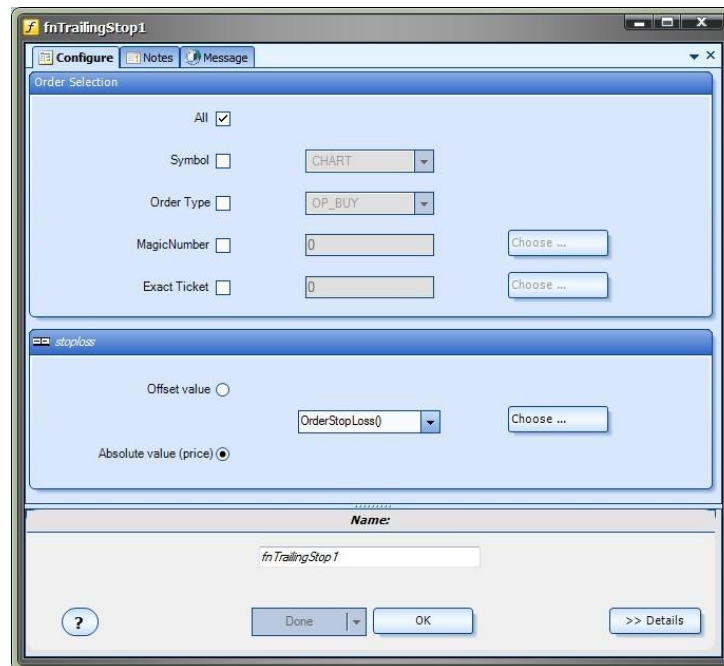
- The **fnTrailingStop** function is used to trail (continually update) the stoploss of an open trade.
- The **fnTrailingStop** function is found in the [Functions Toolbox](#) under the **Trade** menu.

NOTE: A Trailing Stop can be added to any order using the Trailing Stop Parameter of the [fnOpenOrder](#) function.

- Selecting the configuration (+) button of **fnTrailingStop** function will display the **fnTrailingStop Configuration Window**.
- There are three tabs on the **fnTrailingStop Configuration Window**:
 - The [Configure](#) Tab is used to set the values of the [parameters](#) of the Function.
 - The [Message](#) Tab is used to send a message from the Function Element.
 - The [Notes](#) Tab is used to add notes or comments to the Function Element.

The following table provides information about each [parameter](#) of the **fnTrailingStop** function.

Order Selection	<p>Determines what orders of the Account are closed.</p> <ul style="list-style-type: none"> • All: all open orders on the account. • Symbol: only open orders matching the selected symbol. • Order Type: only open orders of the selected order type. • MagicNumber: only open orders with a matching magicnumber. • Exact Ticket: only open orders with the matching order ticket. <p>Note: Selecting Exact Ticket disables all other selection criteria.</p>
stoploss	<p>The value at which to trail the open trade.</p> <ul style="list-style-type: none"> • VTS creates an extern (or Input) variable named StopLoss that can be used for this parameter: its default value is 200. • VTS creates an extern (or Input) variable named TrailingStop that can be used for this parameter: its default value is 100. • A value can be manually entered into the text-box. • A value can be selected using the Choose button. • A value can be entered as an integer or as a price level by selecting the Offset value or Absolute value button. • The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].



The bottom portion of the window allows the [Element Name](#) to be saved.

init and deinit

The functions **init** and **deinit** are special MQL functions that can be used in any Expert Advisor.

- The **init** function is called one time when the Expert Advisor is first attached to a chart.
- The **deinit** function is called one time when the Expert Advisor is removed from a chart.

To use the init or deinit functions in VTS

- Create a new drawing name it **init** or **deinit**.
- Modify the drawing to perform the required functionality.
- Save the drawing.
- From the [ToolBox](#), under the **System Functions** menu, find the **init** or **deinit** menu item and drag it onto the main system drawing.
- **Do not connect init or deinit Elements!** Simply existing on the main system drawing will cause each function to execute properly. If you connect the Elements, they will be called on each tick just like any other [Element](#).

Note: For Expert Advisors that created Objects (such as vertical and horizontal lines), it can be convenient to create a **deinit** function that uses the MQL function **ObjectsDeleteAll** to remove all lines from the price chart when the EA is removed.

Getting Started with VTS

- [Installation](#)
- [Set the License Key](#)
- [Set the MT Paths](#)
- [The User Interface](#)
- [The Basics](#)
- [HowTos](#)

MetaTrader Platform

What is MetaTrader?

The MetaTrader Platform is a Forex Trading platform developed by the MetaQuotes company.

The platform offers the functionality you would expect from a trading platform, such as:

- The ability to open, modify and close trades.
- The ability to view price currency charts for many currency pairs and in many timeframes (1 minute to 1 month)
- The ability to attach technical indicators to price charts

What allows the MetaTrader platform to stand out is its ability to create [Expert Advisors](#) and [Custom Indicators](#) using the [MQL](#) programming language.

There have been 3 major releases of the MetaTrader platform: 3.0, 4.0 and 5.0.

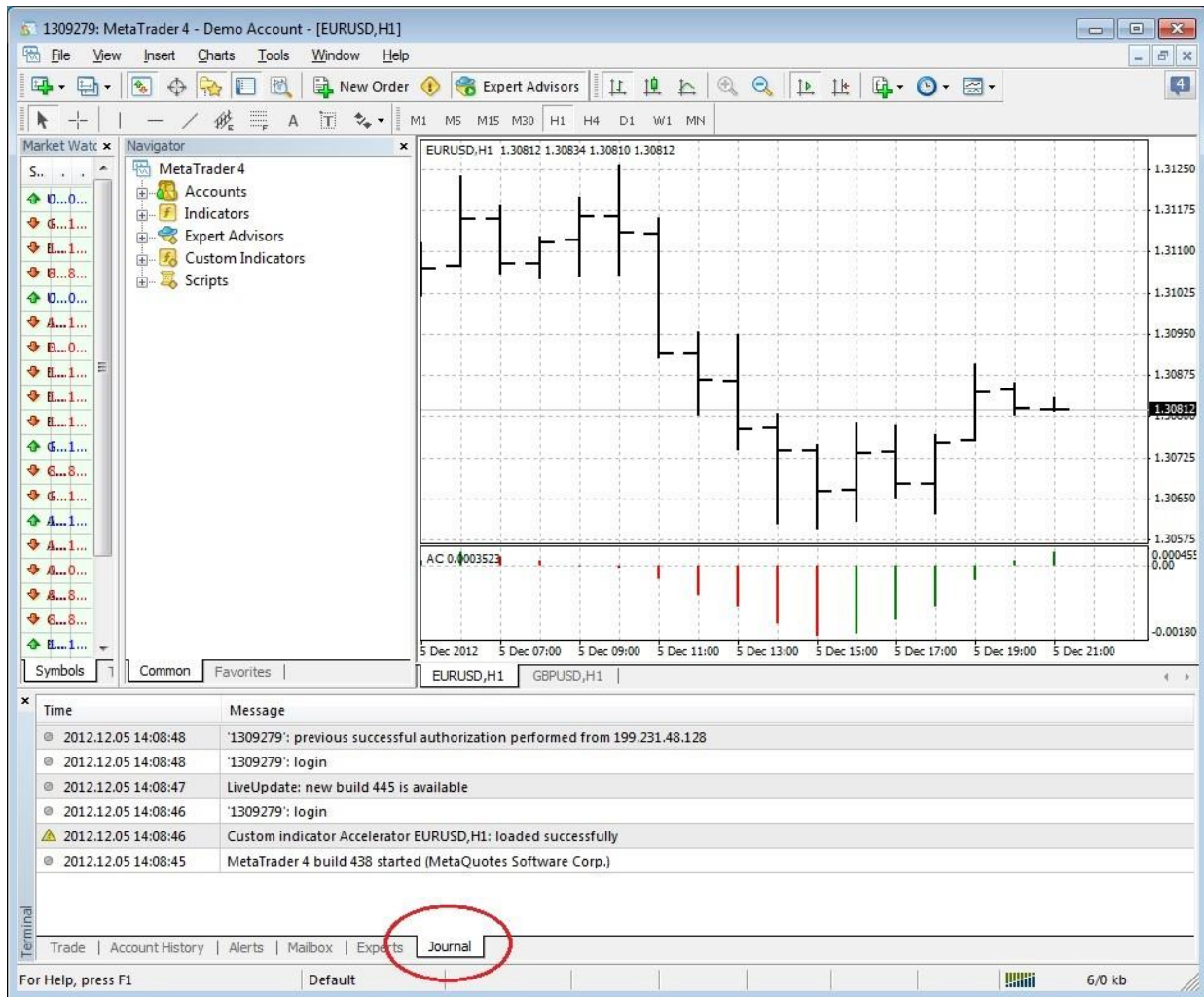
What is MQL?

MQL is a programming language. The language is similar to the C programming language. It supports a number of English-based keywords and uses a typical control flow, including if/then clauses.

MQL is used to build [Expert Advisors](#) and Custom Indicators. A file containing human-readable MQL code is compiled into an executable file that is run on the MetaTrader Platform.

Journal Tab

- The **Journal** Tab is found at the bottom of the [MetaTrader](#) Platform.
- Messages related to order status, logins and EA/Indicator load status appear in the **Journal** Tab.



Experts Tab

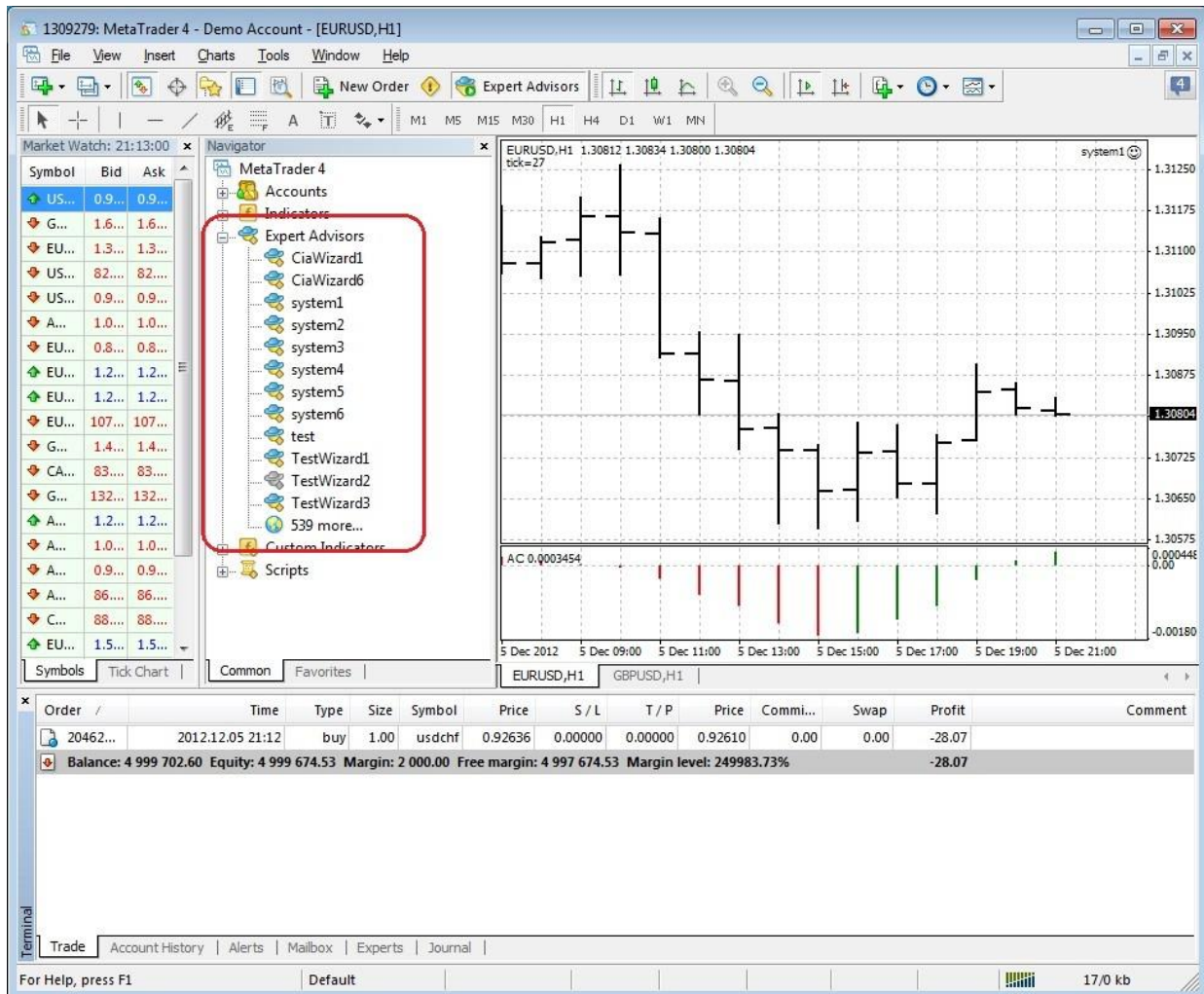
- The **Experts** Tab is found at the bottom of the [MetaTrader](#) Platform.
- All messages from an EA using the [MQLPrint](#) function appear in the **Experts** Tab.

The screenshot shows the MetaTrader 4 interface. The main chart area displays a candlestick chart for EURUSD,H1. The Navigator panel on the left shows a tree view of Expert Advisors, including system1 through system6 and TestWizard1 through TestWizard3. The Experts Tab at the bottom shows a list of messages for system1 and Accelerator EURUSD,H1. The 'Experts' tab name is circled in red.

Time	Message
2012.12.05 14:11:20	system1 EURUSD,H1: initialized
2012.12.05 14:11:20	system1 EURUSD,H1 inputs: MagicNumber=99999; TakeProfit=200; TrailingStop=100; StopLoss=200; Lots=1; Slippage=3; TradeComment="VTS";
2012.12.05 14:11:18	system1 EURUSD,H1: loaded successfully
2012.12.05 14:08:46	Accelerator EURUSD,H1: initialized
2012.12.05 14:08:46	Accelerator EURUSD,H1: loaded successfully

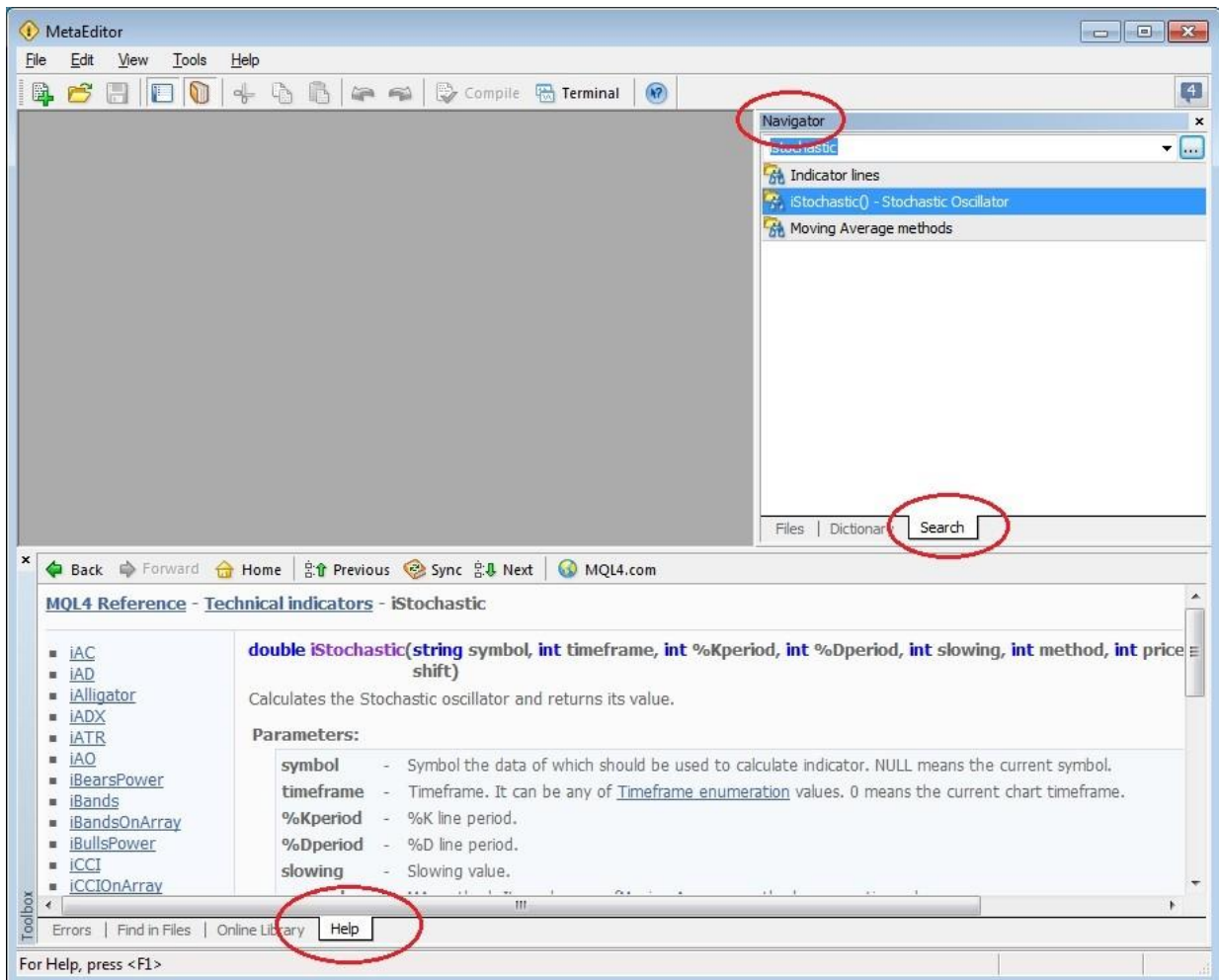
Experts Menu

- The **Experts Advisor** menu is found in the **Navigator** window of the **MetaTrader** Platform.
- All **Experts Advisors** found in the **MetaTrader** Platform appear under the **Experts Advisor** menu.



MQL Help

- The **Help** available from the [MetaEditor](#) is useful for learning the details of any native [MetaTrader](#) Platform function.



Trade Tab

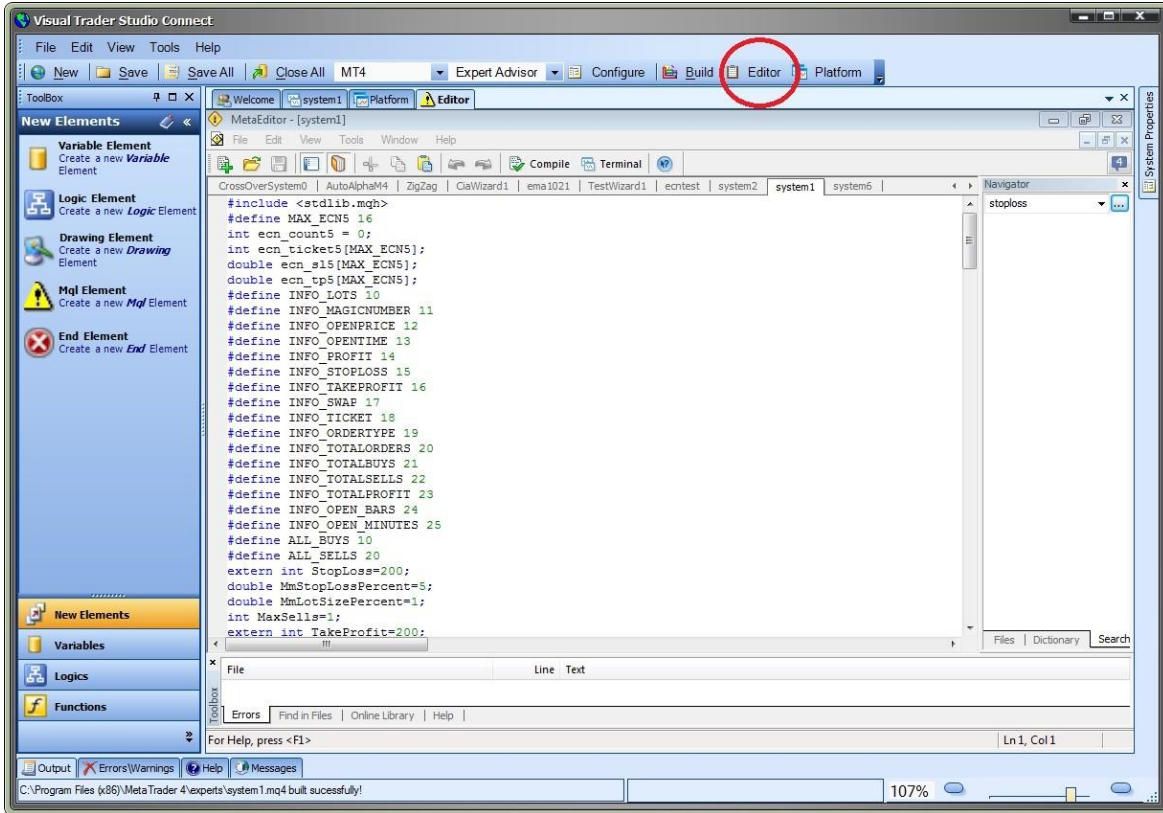
- The **Trade** Tab is found at the bottom of the [MetaTrader](#) Platform.
- The **Trade** Tab show the current status of all open orders.
- The *Comment* field is derived from the *Comment* parameter of the [fnOpenOrder](#) function
- NOTE: If the *Comment* column is not shown, right click the top menu of the **Trade** Tab and select *Comments*.

The screenshot shows the MetaTrader 4 interface. The Trade Tab is highlighted with a red circle at the bottom left. The main window displays a candlestick chart for EURUSD,H1. The Trade Tab table shows the following data:

Order /	Time	Type	Size	Symbol	Price	S / L	T / P	Price	Commi...	Swap	Profit	Comment
204626...	2012.12.05 21:30	buy	1.00	eurusd	1.30795	1.30595	1.30995	1.30780	0.00	0.00	-15.00	Visual Trading Studio
Balance: 4 999 688.56 Equity: 4 999 673.56 Margin: 2 615.90 Free margin: 4 997 057.66 Margin level: 191126.33%											-15.00	

The MetaEditor

The MetaEditor is used to create edit and view mq4 files. It can be launched inside of VTS by clicking the Editor button.



ECN Brokers

[ECN](#) brokers provide direct access to the Forex market without using a dealing desk.

When running an Expert Advisor, MetaTrader ECN brokers do not allow trades to be opened with non-zero limits. The takeprofit and the stoploss must be equal to zero or the order will fail to open.

All Expert Advisors created by VTS provide an ECN input parameter.

If the ECN input parameter is set to true:

- All trades are open with zero takeprofit and stoploss values.
- On the next tick, the trade is modified with the user-set takeprofit and stoploss values.

HowTos

- [Start New System](#)
- [Add Trailing Stop](#)

Start New System

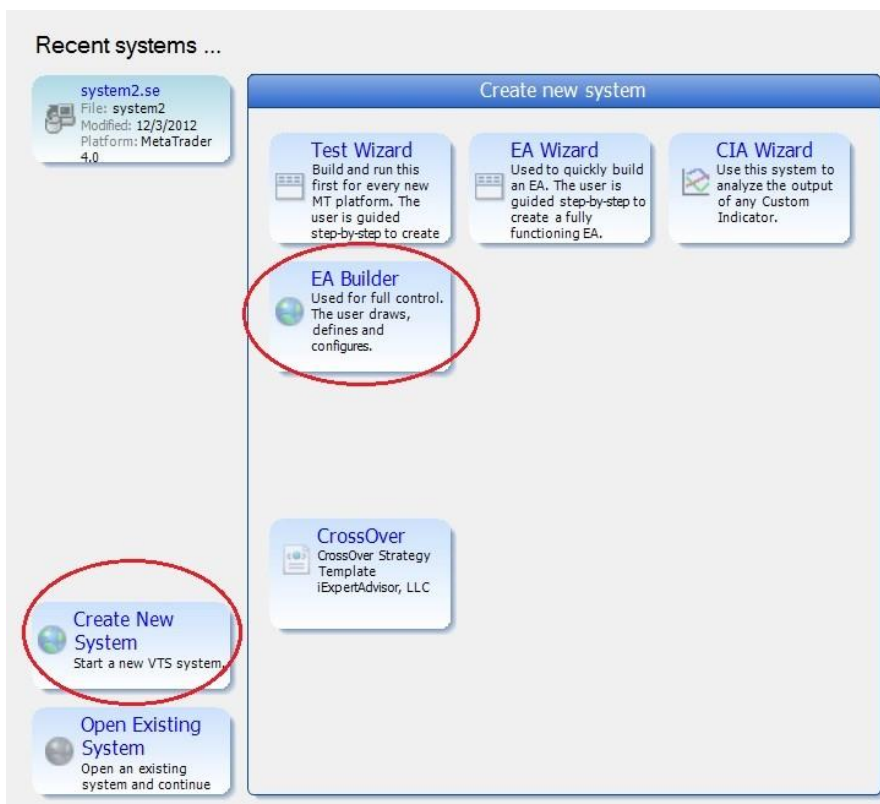
This **HowTo** provides step-by-step instructions for creating a new [VTSsystem](#) with two [Drawing Functions](#).

[start here](#)

Step 1

- **Step 1 (Start New System)**

From the [Welcome](#) screen, select **Create New System** and then select **EA Builder**.

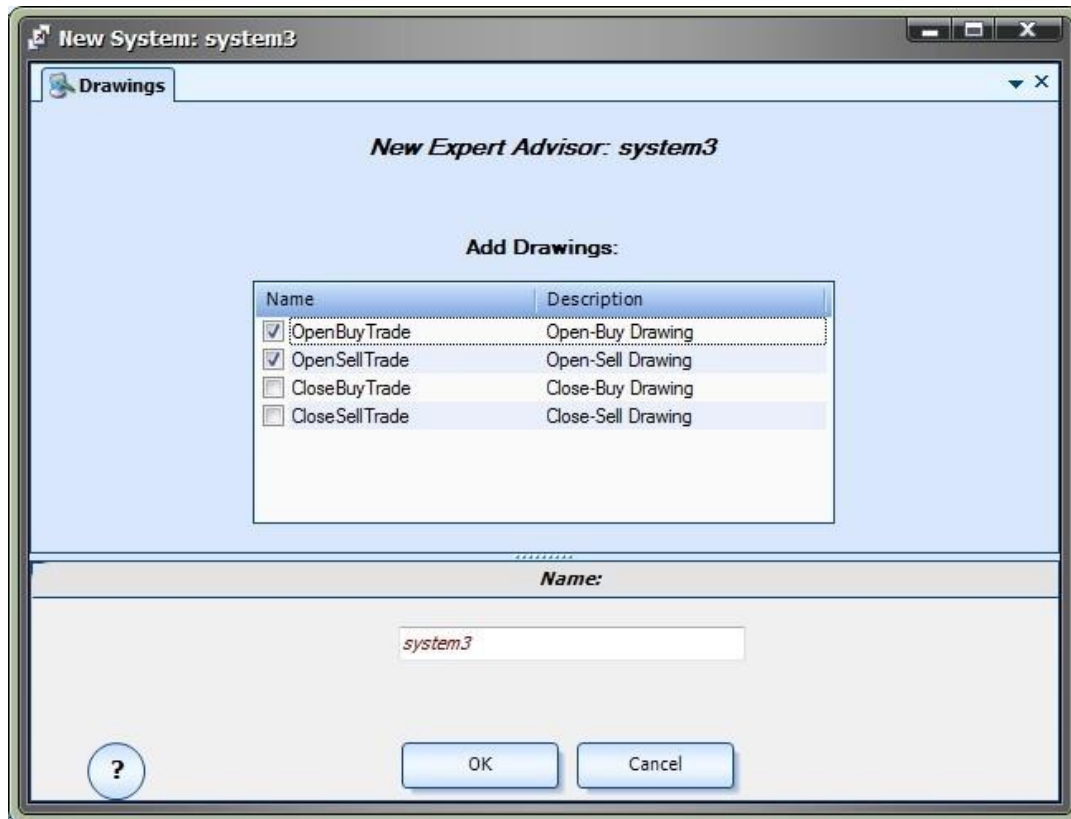


Step 2

- **Step 2 (Start New System)**

On the **New System** entry screen, uncheck *CloseBuyTrade* and *CloseSellTrade* and enter a name in the name field for the name of the [system](#) (This name will also be the name of the [Expert Advisor](#)).

Many EAs do not use explicit logic to close their trades - they allow the stoploss or takeprofit to close the trade. This EA will not define close functions, so we unchecked them.



Step 3

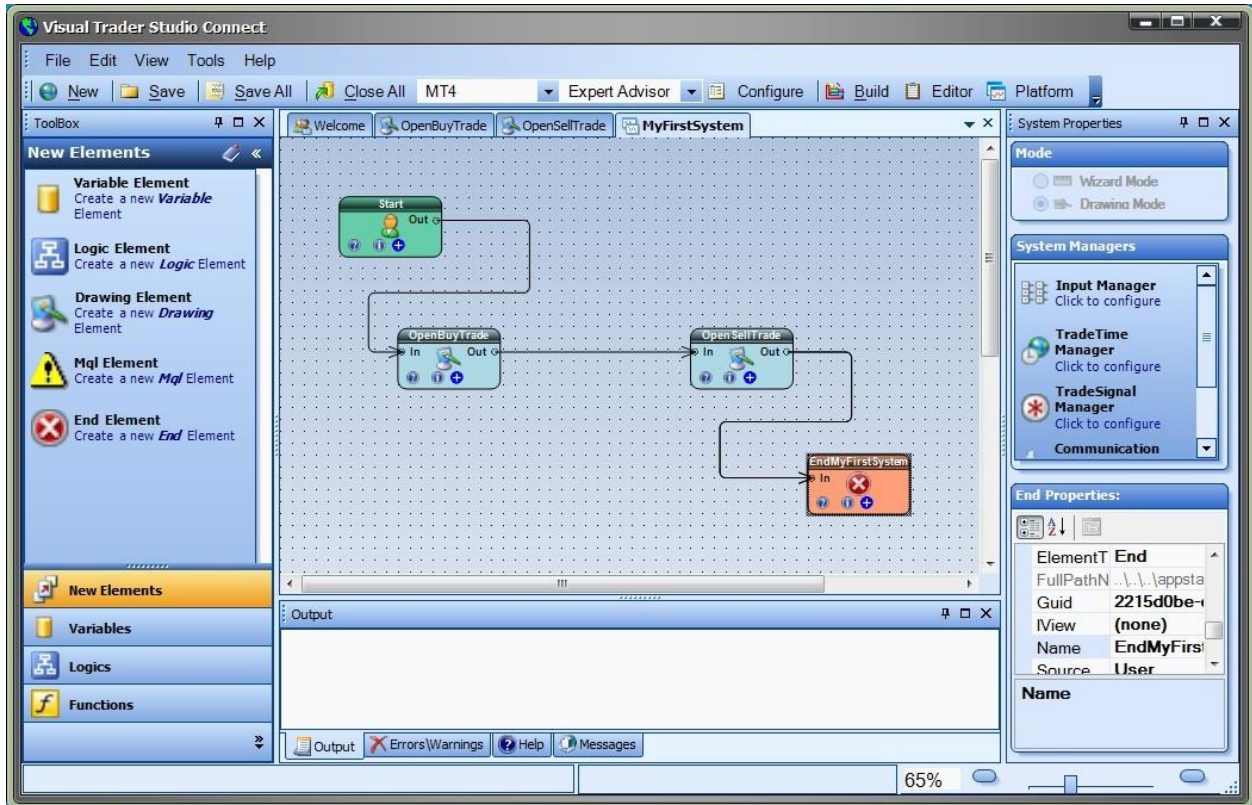
- **Step 3 (Start New System)**

After you select OK in the **New System** entry screen, [VTS](#) will build three [Drawing Tabs](#):

- **MyFirstSystem** (or System1, or whatever you named the system): This is the [main system](#) drawing.
- **OpenBuyTrade**: This is the [Drawing function](#) for opening a buy trade.
- **OpenSellTrade**: This is the [Drawing function](#) for opening a sell trade.

The [main system](#) drawing shows execution:

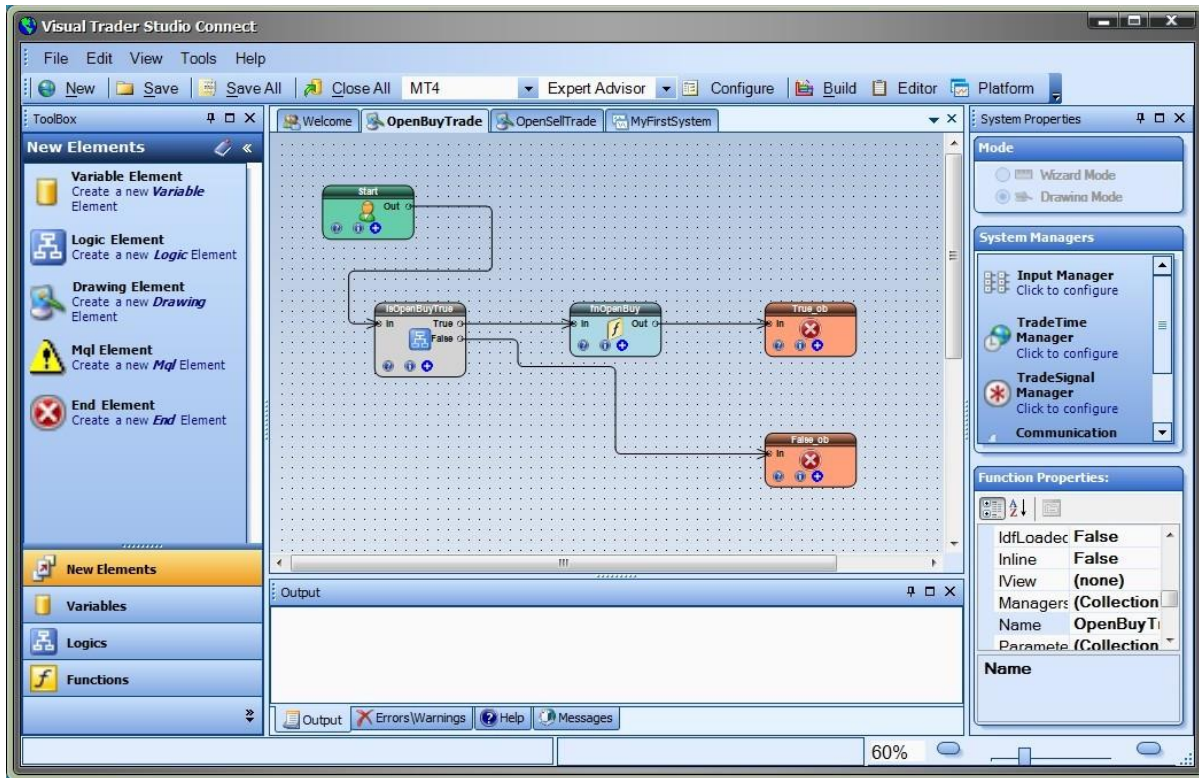
- beginning at the [StartElement](#),
- then traveling to the **OpenBuyTrade** function [Element](#),
- then traveling to the **OpenSellTrade** function [Element](#),
- finally terminating at the [EndElement](#).



The **OpenBuyTrade** drawing shows execution:

- beginning at the [StartElement](#),
- then traveling to the **IsOpenBuyTrueLogicElement**,
- if the Logic is True, traveling to the [fnOpenBuy](#) function
- finally terminating at the [EndElement](#). (either through the True or False of the Logic Element)

Note: **OpenSellTrade** is the same except the trade function is [fnOpenSell](#).



Step 4

- **Step 3 (Start New System)**

That's it! You've started a new trading system.

Now you are ready to:

- Define the [Logic](#) (this may required adding [platfrom functions](#) to the drawing)
- Selecting the [parameters](#) for the [fnOpenOrder](#) functions
- **Build** your EA (click the Build button the [main menu](#))
- **Run** your EA (click the Platform button the [main menu](#))

Add Trailing Stop

Add Trailing Stop

This *HowTo* provides step-by-step instructions for adding a trailing stop to an open order.

[start here](#)

Step 1

- **Step 1 (Add Trailing Stop)**

Drag an *fnOpenOrder* function from the [Functions Toolbox](#) under the **Trade** menu onto the Drawing Pad.



Step 2

- **Step 2 (Add Trailing Stop)**

Selecting the configuration (+) button of *fnOpenOrder* function to display the *fnOpenOrder Configuration Window*.

Check the [Use Trailing Stop](#) Checkbox and select a value.



FOREX DISCLAIMER: Risk Disclosure

- **FOREX Risk Disclosure**

Please read this carefully. If you don't understand any of the information provided in this disclosure or if you have any questions, please contact me. The National Futures Association (NFA) and CFTC (Commodity Futures Trading Commission), the regulatory agencies for the FOREX and futures market in the United States, require that customers be informed about potential risks in the FOREX market (see the information below).

The products and services referenced at this site are for educational purpose only, and are not intended to replace individual research or licensed investment advice. Unique experiences and past performances do not guarantee future results. Testimonials are not representative of all clients. Certain accounts may have worse performance than those indicated or alluded to. Trading currencies involves substantial risk, and there is always the potential for loss. Your trading results may vary. No representation is being made that these products or services, and any associated advice or training, will guarantee profits, or not result in losses from trading. Neither the products, nor any training services held in conjunction therewith, including, without limitation, through online seminars, alert services, in conjunction with our advertising and promotional campaigns, during our in-person seminars or otherwise, should be construed as providing a trade recommendation or the giving of investment advice. The purchase, sale or advice regarding a currency can only be performed by a licensed Broker/Dealer. Neither iExpertAdvisor, nor any of its affiliates, partners or associates involved in the production and maintenance of these products, services or this site, is a registered Broker/Dealer or Investment Advisor in any State or Federally-sanctioned jurisdiction. All purchasers of products and services referenced at this site are encouraged to consult with a licensed representative of their choice regarding any particular trade or trading strategy.

The information contained on this website represents the current view of iExpertAdvisor LLC on the issues discussed as of the date of publication. Because iExpertAdvisor must respond to changing market conditions, it should not be interpreted to be a commitment on the part of iExpertAdvisor, and iExpertAdvisor cannot guarantee the accuracy of any information presented after the date of publication.

iExpertAdvisor MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of iExpertAdvisor Corporation.

iExpertAdvisor may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from iExpertAdvisor, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

I am not a licensed financial advisor, registered investment adviser or registered broker-dealer. I do not provide investment or financial advice or make investment recommendations, nor am I in the business of transacting trades. Although the information, links and resources may teach or link you to the use of investing strategies, it is not investment, financial or other advice. Nothing contained in these communication files constitutes a solicitation, recommendation, promotion, endorsement or offer (buy or sell) by me or any of my provided resources, of any particular security, mutual fund, transaction or investment.

STOCK TRADING, OPTIONS TRADING, FUTURES TRADING, and FOREX TRADING all involve high risks, and while there is potential for gain, there is a high risk of loss, and you can potentially lose a lot of money. Past or simulated performance does not guarantee future results. Investing results will vary from individual to individual. Only you are liable for your investment and trading decisions and the

results there from, and you agree to hold me harmless and without liability from your investment decisions and the results there from. There are no implied or express warranties of merchantability or fitness for a particular purpose. Please use your due diligence and govern yourself accordingly.

- **Earnings & Income Disclaimer**

We make every effort to ensure that we accurately represent these products and services and their potential for income. Earning and Income statements made by our company and its customers are estimates of what we think you can possibly earn. There is no guarantee that you will make these levels of income and you accept the risk that the earnings and income statements differ by individual.

As with any business, your results may vary, and will be based on your individual capacity, business experience, expertise, and level

of desire. There are no guarantees concerning the level of success you may experience. The testimonials and examples used are exceptional results, which do not apply to the average purchaser, and are not intended to represent or guarantee that anyone will achieve the same or similar results. Each individual's success depends on his or her background, dedication, desire and motivation.

There is no assurance that examples of past earnings can be duplicated in the future. We cannot guarantee your future results and/or success. There are some unknown risks in business and on the Internet that we cannot foresee which can reduce results. We are not responsible for your actions.

The use of our information, products and services should be based on your own due diligence and you agree that our company is not liable for any success or failure of your business that is directly or indirectly related to the purchase and use of our information, products and services.

- **Risks Associated with FOREX and Trading**

Trading foreign currencies is a challenging and potentially profitable opportunity for educated and experienced investors. However, before deciding to participate in the FOREX market, you should carefully consider your investment objectives, level of experience and risk appetite. Most importantly, do not invest money you cannot afford to lose.

There is considerable exposure to risk in any foreign exchange transaction. Any transaction involving currencies involves risks including, but not limited to, the potential for changing political and/or economic conditions that may substantially affect the price or liquidity of a currency.

Moreover, the leveraged nature of FOREX trading means that any market movement will have an equally proportional effect on your deposited funds. This may work against you as well as for you. The possibility exists that you could sustain a total loss of initial margin funds and be required to deposit additional funds to maintain your position. If you fail to meet any margin call within the time prescribed, your position will be liquidated and you will be responsible for any resulting losses. Investors may lower their exposure to risk by employing risk-reducing strategies such as 'stop-loss' or 'limit' orders.

There are also risks associated with utilizing an internet-based deal execution software application including, but not limited, to the failure of hardware and software.

Trading foreign exchange on margin carries a high level of risk, and may not be suitable for all investors. The high degree of leverage can work against you as well as for you. Before deciding to invest in foreign exchange you should carefully consider your investment objectives, level of experience, and risk appetite. The possibility exists that you could sustain a loss of some or all of your initial investment and therefore you should not invest money that you cannot afford to lose. You should be aware of all the risks associated with foreign exchange trading, and seek advice from an independent financial advisor if you have any doubts.

- **Benefits and Risks of Leverage**

Even though the FOREX market offers traders the ability to use a high degree of leverage, trading with high leverage may increase the losses suffered. Please use caution when using leverage in trading or investing.

VTS Plug-ins

VTS Plug-ins provide enhanced functionality to the VTS Platform.

Function and Logic Power Plug-in

Requires VTS-Connect minimum version 4.0.0.32

The Power Plug-in provides advanced functionality to the Logic and Function Elements.

- [Logic Power Tab](#)
- [Function Power Tab](#)

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.

Enable the Power Plug-in

Enable the Function and Logic Power Plug-in

You must enter your License key to enable the **Function and Logic Power Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

- The **email** address is the email address used to purchase [VTS](#).
- The **License Key** is the key listed in the Members Area.
- The **Update** button is used to verify the email address and license key.
- The **edit key** button is used edit the key value.



Logic Power Tab

The **Logic PowerTab** offers extended functionality to the VTS [LogicElement](#).

- Generally speaking, when a [LogicElement](#) evaluates to **True**, the execution of an Expert Advisor follows a path to perform a specific action, such as open, close or modify a trade.
- The options available from the **Logic Power Tab** allow a trader to apply advanced rules to *how and when* the [LogicElement](#) evaluates to **True**. These rules are outside the typical *greater-than, less-than* evaluation of a [logical condition](#).
- The **Logic Power Tab** offers the following options:
 - **Logic Evaluation Frequency**
 - **Minimum required TRUE count**
 - **Maximum allowed TRUE count**
- **Note:** All times are determined by the MetaTrader Platform's clock, *not* the local clock of the running PC.

Logic Evaluation Frequency

This option defines how *often* a Logic is evaluated to be **True**.

On every new tick	The Logic is checked for True on every incoming tick . This is the default selection.
On every new bar	The Logic is checked for True only on the start of a new bar (or candle). The bar length is defined by the chart to which the EA is attached.
On every new hour	The Logic is checked for True only at the start of every hour.
On every new day	The Logic is checked for True only at the start of every day.

Minimum required TRUE count

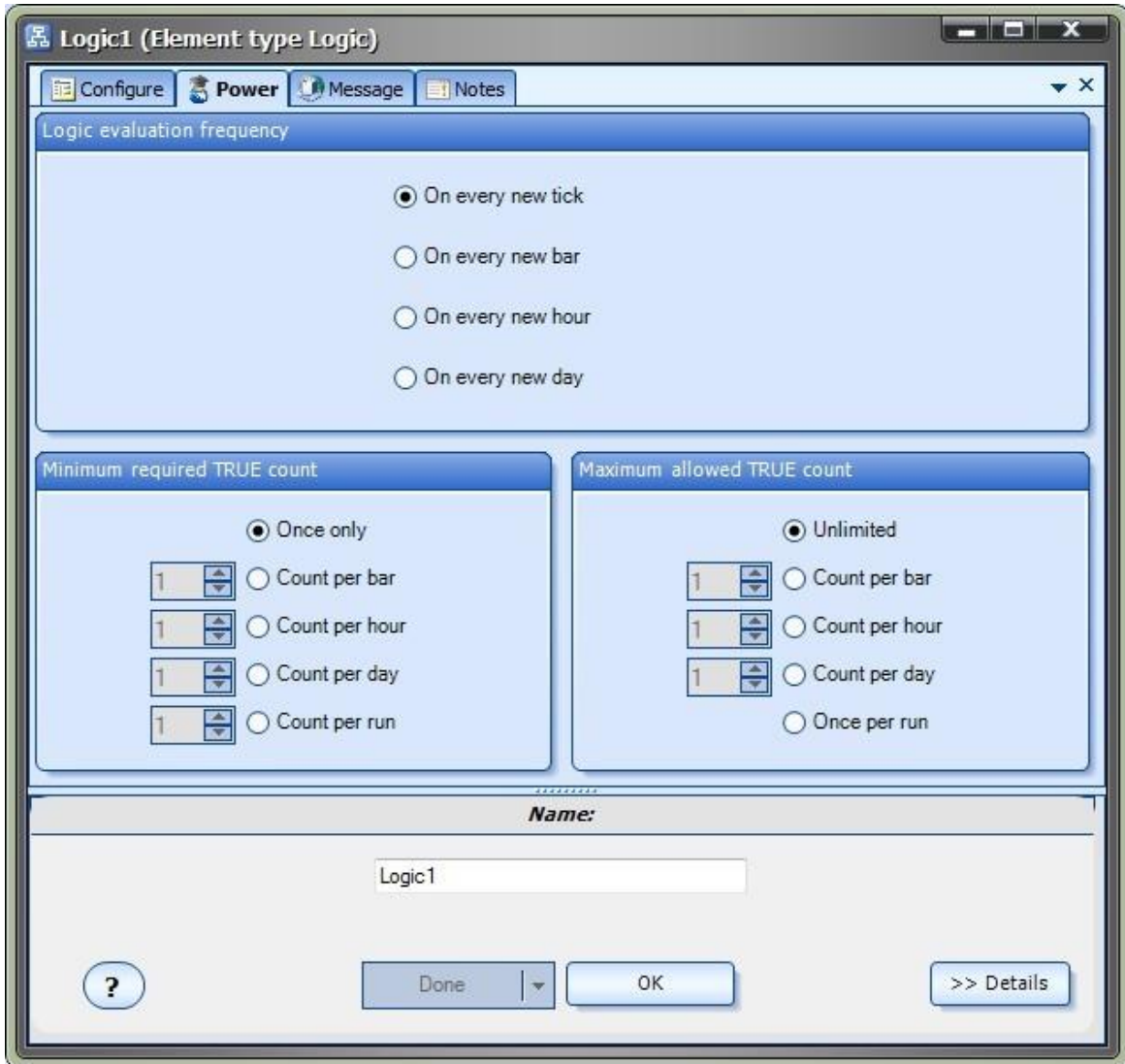
This option defines *how many times* a [Logic](#) must evaluate to **True** before the [Logic](#) returns a **True** value.

<i>Once only</i>	The Logic only needs to evaluate to True once. This is the default selection.
<i>Count per bar</i>	The Count value can be set to any integer above one. The Logic must evaluate to True Count times within a single Bar to return True . The bar length is defined by the chart to which the EA is attached. The running count is reset to zero at the start of a new Bar .
<i>Count per hour</i>	The Count value can be set to any integer above one. The Logic must evaluate to True Count times within an Hour to return True . The running count is reset to zero at the start of a new Hour .
<i>Count per day</i>	The Count value can be set to any integer above one. The Logic must evaluate to True Count times within a Day to return True . The running count is reset to zero at the start of a new Day .
<i>Count per run</i>	The Count value can be set to any integer above one. The Logic must evaluate to True Count times within one Run of the EA. The running count is reset to zero each time the EA is restarted.

Maximum allowed TRUE count

This option defines how many times a [Logic may evaluate](#) to **True**.

<i>Unlimited</i>	The Logic may return True and unlimited number of times. This is the default selection.
<i>Count per bar</i>	The Logic may return True once per Bar .
<i>Count per hour</i>	The Logic may return True once per Hour .
<i>Count per day</i>	The Logic may return True once per Day .
<i>Once per run</i>	The Logic may return True once per Run .



Function Power Tab

The **Function PowerTab** offers extended functionality to the VTS [Platform FunctionElement](#).

- The **Function Power Tab** offers the following options:
 - **Channel value**
 - **Trending value**
 - **Average value**

Channel value

- A channel is a collection of contiguous price bars (or candles).
- This option gets the **Highest** or **Lowest** value, within a specific channel, of the [platform function](#).
- **Channel type** is defined as **CHANNEL_HIGH** or **CHANNEL_LOW**.
- The **Start candle** is the first candle of the channel.
- The **End candle** is the last candle of the channel.

Note: Candles are numbered from right to left, starting at zero. The *currently forming candle* is candle number zero. See the [shift](#)

dialog for a diagram.

Trending value

- A trend is defined when a value moves in a continuous direction, either up or down.
- This option determines if a [platform function](#) is trending or not. A value of one is returned if there is a trend; a value of zero is returned if there is not a trend.
- **Trend type** is defined as **TREND_UP** or **TREND_DOWN**.
- The **Start candle** is the first candle of the channel to be tested for the trend
- The **End candle** is the last candle of the channel to be tested for the trend.
- The **minimum change** value defines the minimum change in value that must occur for a trend to be confirmed. The [default](#) value is 0.
- The **maximum outlier** value defines how many values within the channel can be against the trend and still confirm the trend. The [default](#) value is 0.

Average value

- This option calculates the moving average of the last number of bars of the [platform function](#).
- **Bars** defines the number of bars that are used to calculate the moving average.
- The selected **Method** defines the method used to calculate the moving average. The options are [SMA](#), [EMA](#), [SMMA](#), [LWMA](#).

The screenshot shows the configuration window for the iAC1 indicator. The window title is "iAC1* iAC(CHART,CHART,0)". The interface is organized into three main sections: "Channel value", "Trending value", and "Average value".

- Channel value:** Includes a checkbox for "Get channel value" (unchecked), a "Channel type" dropdown menu set to "CHANNEL_HIGH", and two input fields for "Start candle: 0" and "End candle: 12".
- Trending value:** Includes a checked checkbox for "Get trend value", a "Trend type" dropdown menu set to "TREND_UP", and four input fields: "Start candle: 0", "End candle: 12", "Minimum change: 0", and "Maximum outliers: 0".
- Average value:** Includes a checkbox for "Get average value" (unchecked), a "Method" dropdown menu set to "MODE_SMA", and an input field for "Bars: 12".

At the bottom of the dialog, there is a "Name:" field containing the text "iAC1". To the left of the "Done" button is a help icon "?". To the right of the "Done" button are "Cancel" and ">> Details" buttons.

Candlestick Library Plug-in

Requires VTS-Connect minimum version **4.0.0.37**

The Candlestick Library Plug-in provides candlestick recognition functions from the **Toolbox->Functions->Candlestick** menu.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.

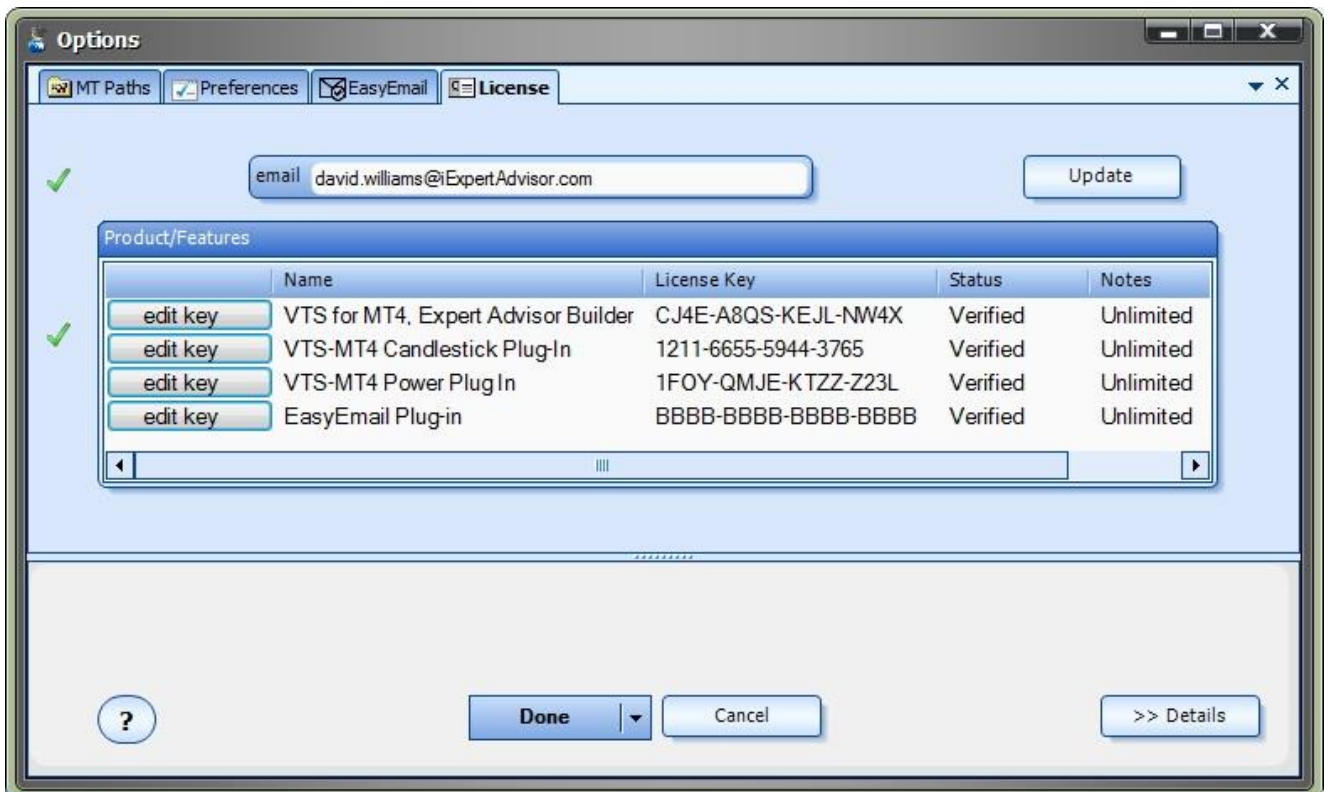


Enable the Candlestick Library Plug-in

You must enter your License key to enable the **Candlestick Library Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

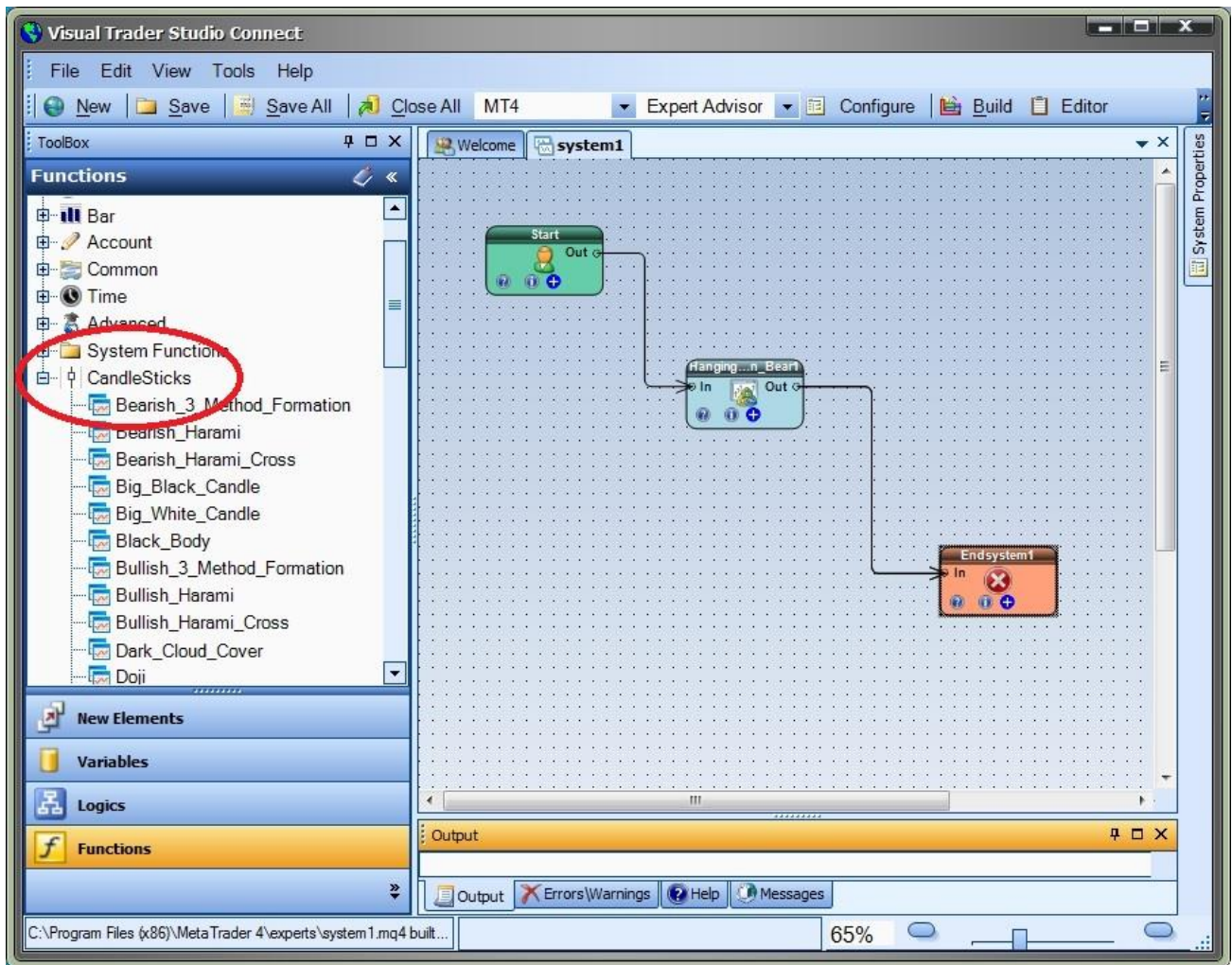
- The **email** address is the email address used to purchase [VTS](#).
- The **License Key** is the key listed in the Members Area.
- The **Update** button is used to verify the email address and license key.
- The **edit key** button is used edit the key value.



Candlestick Functions in the Toolbox

Once enabled, the Candlestick functions are available in the [Toolbox](#) Function tab under the Candlesticks menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.



Candlestick Function Parameters

After a Candlestick function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

All Candlestick functions have the parameters **Symbol, Timeframe and Shift**.

Parameter Name	Data type	Description
Symbol	String	The symbol of the currency to search for the pattern, such as EURUSD, USDJPY, etc.
Timeframe	Integer	The timeframe of the currency to search for the pattern. MetaTrader supports values from 1 minute (PERIOD_M1) to 1 month (PERIOD_MN1).
Shift	Integer	The candle index at which to start searching for the pattern. Zero starts at the currently forming candle, one starts at one candle to the left, etc.

Candlestick functions that use the terms "long" or "short" have the additional parameters **longbody and shortbody**.

Parameter Name	Data type	Description
longbody	double	The minimum value of a candle body to be determined as long.
shortbody	double	The maximum value of a candle body to be determined as short.

If the value of **longbody** is left as 0 (the default value), the candle function will automatically calculate the minimum **longbody** value as 2 times the 4-period ATR (average true range) for the given symbol and timeframe:

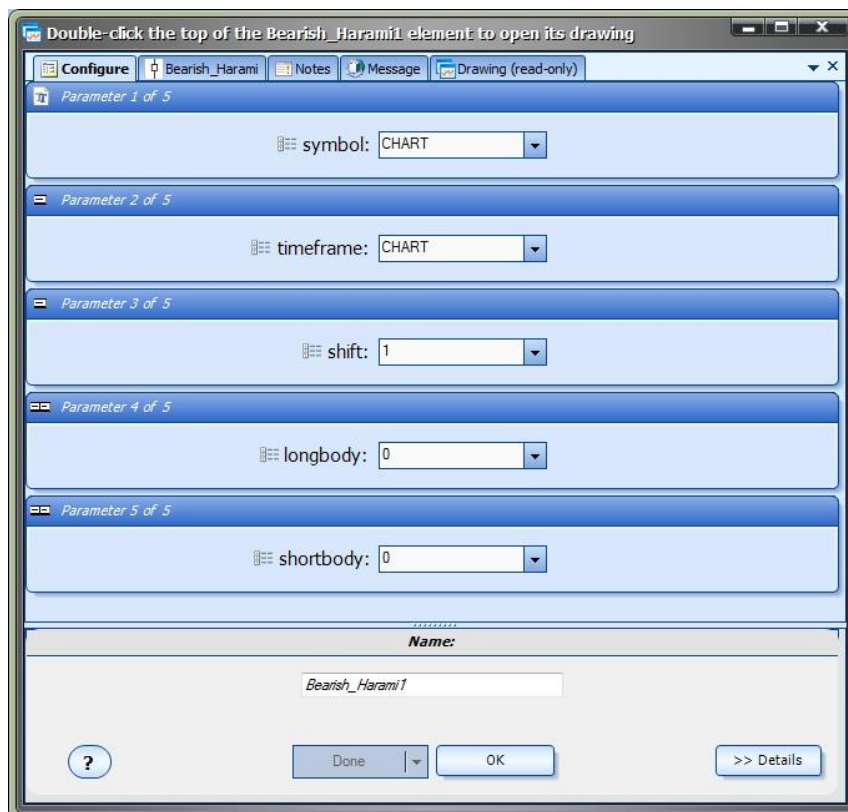
$$2 * iATR(symbol, timeframe, shift)$$

If the value of **shortbody** is left as 0 (the default value), the candle function will automatically calculate the maximum **shortbody** value as the 4-period ATR (average true range) for the given symbol and timeframe:

$$iATR(symbol, timeframe, shift)$$

The **longbody and shortbody** parameters can be set to any value using any combination of arithmetic and/or MetaTrader platform functions.

This is the candle function configuration window for the **Bearish Harami** candlestick pattern:



Candlestick Drawing

The Candlestick functions were created as [Drawing functions](#).

To open the drawing of any Candlestick function, drag the function from the [Toolbox](#) onto the [Drawing Pad](#) and double-click the [Element](#) in the caption area.

Each candlestick drawing contains the same Elements:

- One Start Element
- 3- 5 Variable Elements defined with a scope of [Parameter](#)
- 4-N MetaTrader price functions, such as iClose, iOpen, iHigh and iLow
- One [Logic](#) Element
- Two End Elements, one for True and one for False
- An image of the Candlestick pattern (for reference only, the image is not connected to any Elements)

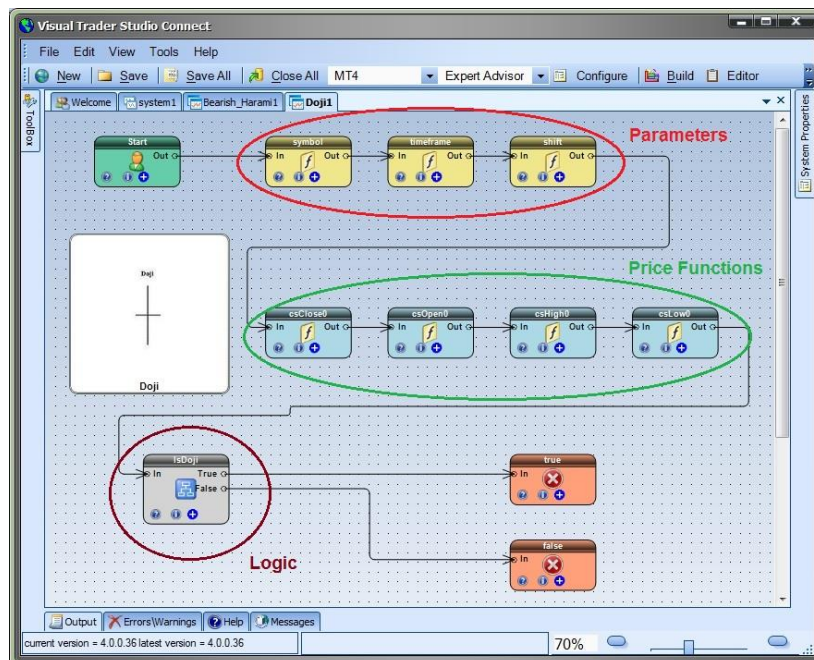
The first row of Elements contains the Parameter Variables. These are the values that appear when you select the (+) button on the Candlestick function Element. These values are referenced through the rest of the drawing. For example, the Symbol and Timeframe values are used in the Price functions.

The second row of Elements contains the price functions that are used in the Logic Element to identify the candlestick pattern. The number of price functions will vary depending on the pattern. In general, the open, close, high and low price values will be need for a number of different candles consecutive candles, where the index of the candle is determined by the [shift](#) value.

The Logic Element is connected after the price functions. The Logic Element will contain many logical conditions. These conditions represent the definition of the candle pattern. For example, a **Bear** candle is defined as a candle that closed lower than it opened. (A bear candle is usually colored black or red on a price chart). This definition is logically expressed as $Close < Open$ (The Close is less-than the open). The conditions with the logic will use price values that already considered the Symbol, Timeframe and Shift.

The drawing terminates with a True or False Element. If the Logic Element resolves to True, meaning the pattern has been recognized, then the drawing will terminate with a value of True. Otherwise the drawing will terminate with a value of False.

Note: Candlestick patterns that use the phrase long or short define two additional Parameters and two MQL Elements to calculate the values of long or short named **getlongbody** and **getshortbody**, respectively. The MQL Elements are found on the first line, connected after the parameter Elements.



Using a Candlestick Function in a Logic Element

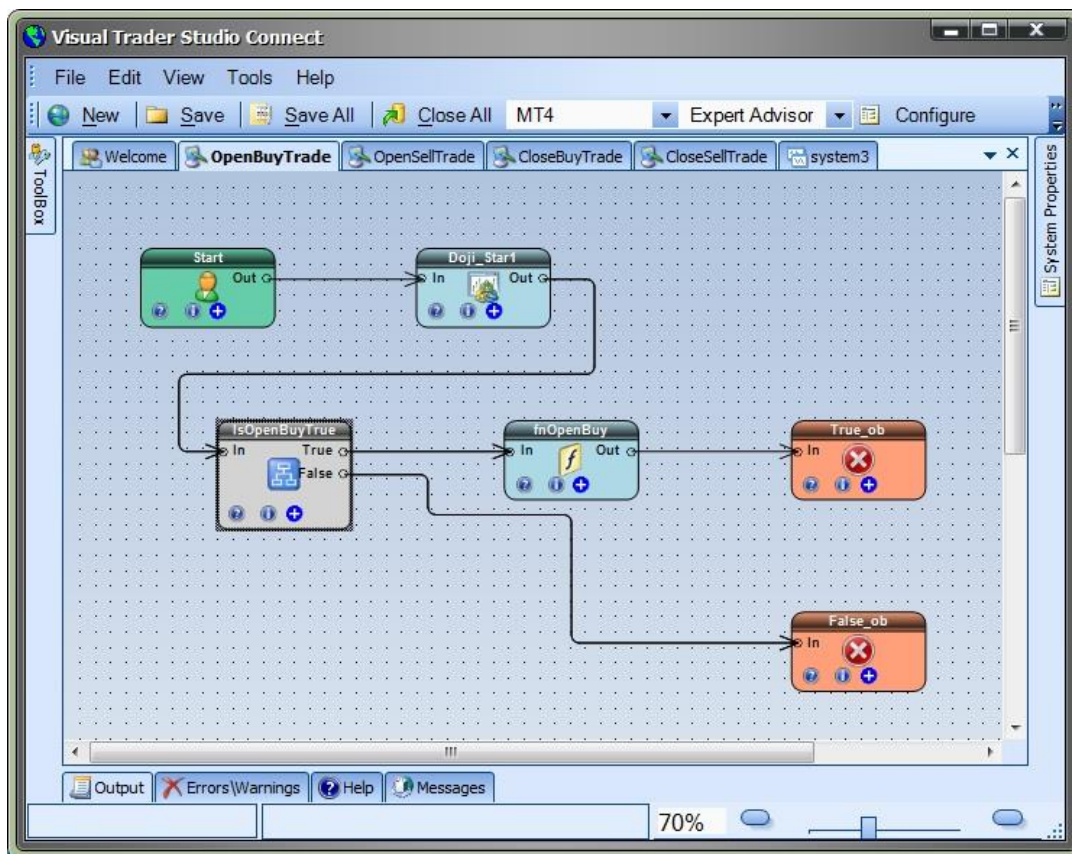
The purpose of a Candlestick function is to identify a formed pattern on a price chart and use the knowledge that the pattern occurred in the logic of an Expert Advisor.

Each Candlestick function returns a value of **True**, if the pattern is recognized, or **False**, if the pattern is not recognized.

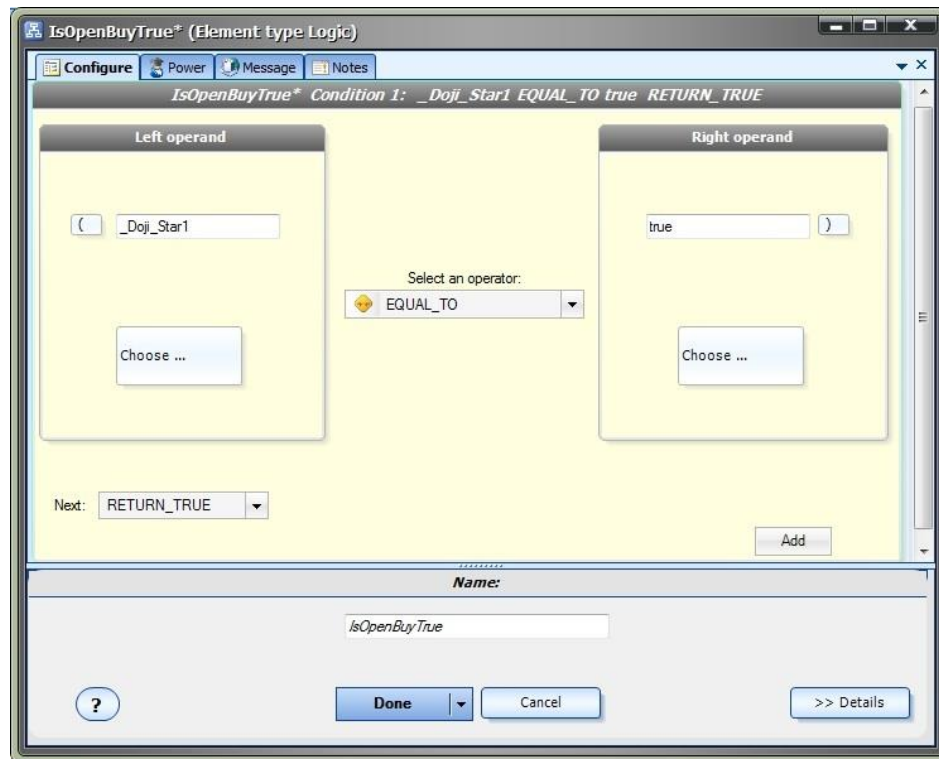
To utilize a Candlestick function:

- Drag and drop a Candlestick function onto the Drawing Pad.
- Connect the Candlestick Element before the Logic Element where it is referenced.
- Click the configuration (+) button on Candlestick Element to change any of the parameters (optional)
- Reference the Candlestick function in a [Logical Condition](#):
 - Set the Left Operand: Select the [Choose](#) button to find the Candlestick function by name (the name of the function with a preceding underscore)
 - Select an Operator, usually EQUAL_TO.
 - Set the Right Operand, usually **true**.

This drawing uses the candlestick function **Doji_Star** on the **OpenBuyTrade** drawing.



This is the Logical Condition of the **IsOpenBuyTrue** Logic Element.



Candlestick Pattern Definitions

Big Black Candle

Has an unusually long black body with a wide range between high and low. Prices open near the high and close near the low. Considered a bearish pattern.

Big White Candle

Has an unusually long white body with a wide range between high and low of the day. Prices open near the low and close near the high. Considered a bullish pattern.

Black Body

Formed when the opening price is higher than the closing price. Considered to be a bearish signal.

Doji

Formed when opening and closing prices are virtually the same. The lengths of shadows can vary.

Dragon Fly Doji

Formed when the opening and the closing prices are at the highest of the day. If it has a longer lower shadow it signals a more bullish trend. When appearing at market bottoms it is considered to be a reversal signal.

Gravestone Doji

Formed when the opening and closing prices are at the lowest of the day. If it has a longer upper shadow it signals a bearish trend. When it appears at market top it is considered a reversal signal.

Long Legged Doji

Consists of a Doji with very long upper and lower shadows. Indicates strong forces balanced in opposition.

Hanging Man

A black or a white candlestick that consists of a small body near the high with a little or no upper shadow and a long lower tail. The lower tail should be two or three times the height of the body. Considered a bearish pattern during an uptrend.

Hammer

A black or a white candlestick that consists of a small body near the high with a little or no upper shadow and a long lower tail. Considered a bullish pattern during a downtrend.

Inverted Black Hammer

A black body in an upside-down hammer position. Usually considered a bottom reversal signal.

Inverted Hammer

It consists of black or a white candlestick in an upside-down hammer position.

Long Lower Shadow

A black or a white candlestick is formed with a lower tail that has a length of 2/3 or more of the total range of the candlestick. Normally considered a bullish signal when it appears around price support levels.

Long Upper Shadow

A black or a white candlestick with an upper shadow that has a length of 2/3 or more of the total range of the candlestick. Normally considered a bearish signal when it appears around price resistance levels.

Marubozu

A long or a normal candlestick (black or white) with no shadow or tail. The high and the lows represent the opening and the closing prices. Considered a continuation pattern.

Shooting Star

A black or a white candlestick that has a small body, a long upper shadow and a little or no lower tail. Considered a bearish pattern in an uptrend.

Spinning Top

A black or a white candlestick with a small body. The size of shadows can vary. Interpreted as a neutral pattern but gains importance when it is part of other formations.

White Body Candle

Formed when the closing price is higher than the opening price and considered a bullish signal.

Shaven Bottom

A black or a white candlestick with no lower tail. [Compare with Inverted Hammer.]

Shaven Head

A black or a white candlestick with no upper shadow. [Compared with hammer.]

Bearish Harami

It consists of an unusually large white body followed by a small black body (contained within large white body). It is considered as a bearish pattern when preceded by an uptrend.

Bearish Harami Cross

It has a large white body followed by a Doji. It is considered as a reversal signal when it appears at the top.

Bearish_3_Method_Formation

It has a long black body followed by three small bodies (normally white) and a long black body. The three white bodies are contained within the range of first black body. This is considered as a bearish continuation pattern.

Bullish 3 Method Formation

It consists of a long white body followed by three small bodies (normally black) and a long white body. The three black bodies are contained within the range of first white body. This is considered as a bullish continuation pattern.

Bullish Harami

It consists of an unusually large black body followed by a small white body (contained within large black body). It is considered as a bullish pattern when preceded by a downtrend.

Bullish_Harami_Cross

It has a large black body followed by a Doji. It is considered as a reversal signal when it appears at the bottom.

Dark Cloud Cover

It consists of a long white candlestick followed by a black candlestick that opens above the high of the white candlestick and closes well into the body of the white candlestick. It is considered as a bearish reversal signal during an uptrend.

Engulfing Bearish Line

It consists of a small white body that is contained within the followed large black candlestick. When it appears at top it is considered as a major reversal signal.

Engulfing Bullish Line

It consists of a small black body that is contained within the followed large white candlestick. When it appears at bottom it is interpreted as a major reversal signal.

Evening Doji Star

It consists of three candlesticks. First is a large white body candlestick followed by a Doji that gaps above the white body. The third candlestick is a black body that closes well into the white body. When it appears at the top it is considered as a reversal signal. It signals more bearish trend than the evening star pattern because of the doji that has appeared between the two bodies.

Evening Star

It consists of a large white body candlestick followed by a small body candlestick (black or white) that gaps above the previous. The third is a black body candlestick that closes well within the large white body. It is considered as a reversal signal when it appears at top level.

Falling Window

A window (gap) is created when the high of the second candlestick is below the low of the preceding candlestick. It is considered that the window should be filled with a probable resistance.

Morning Doji Star

It consists of a large black body candlestick followed by a Doji that occurred below the preceding candlestick. On the following day, a third white body candlestick is formed that closed well into the black body candlestick which appeared before the Doji. It is considered as a major reversal signal that is more bullish than the regular morning star pattern because of the existence of the Doji.

Morning Star

It consists of a large black body candlestick followed by a small body (black or white) that occurred below the large black body candlestick. On the following day, a third white body candlestick is formed that closed well into the black body candlestick. It is considered as a major reversal signal when it appears at bottom.

On Neckline

In a downtrend, it consists of a black candlestick followed by a small body white candlestick with its close near the low of the preceding black candlestick. It is considered as a bearish pattern when the low of the white candlestick is penetrated.

Three Black Crows

It consists of three long black candlesticks with consecutively lower closes. The closing prices are near to or at their lows. When it appears at top it is considered as a top reversal signal.

Three White Soldiers

It consists of three long white candlesticks with consecutively higher closes. The closing prices are near to or at their highs. When it appears at bottom it is interpreted as a bottom reversal signal.

Tweezer Bottoms

It consists of two or more candlesticks with matching bottoms. The candlesticks may or may not be consecutive and the sizes or the colours can vary. It is considered as a minor reversal signal that becomes more important when the candlesticks form another pattern.

Tweezer Tops

It consists of two or more candlesticks with matching tops. The candlesticks may or may not be consecutive and the sizes or the colours can vary. It is considered as a minor reversal signal that becomes more important when the candlesticks form another pattern.

Doji Star

It consists of a black or a white candlestick followed by a Doji that gaps above or below these. It is considered as a reversal signal with confirmation during the next trading day.

Piercing Line

It consists of a black candlestick followed by a white candlestick that opens lower than the low of preceding but closes more than halfway into black body candlestick. It is considered as reversal signal when it appears at bottom.

Rising Window

A window (gap) is created when the low of the second candlestick is above the high of the preceding candlestick. It is considered that the window should provide support to the selling pressure.

Easy Email Plug-in

Requires VTS-Connect minimum version **4.0.0.38**

The **Easy Email Plug-in** allows any Expert Advisor to send emails with price-chart attachments using any email provider such Gmail, Yahoo, Hotmail, etc.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.

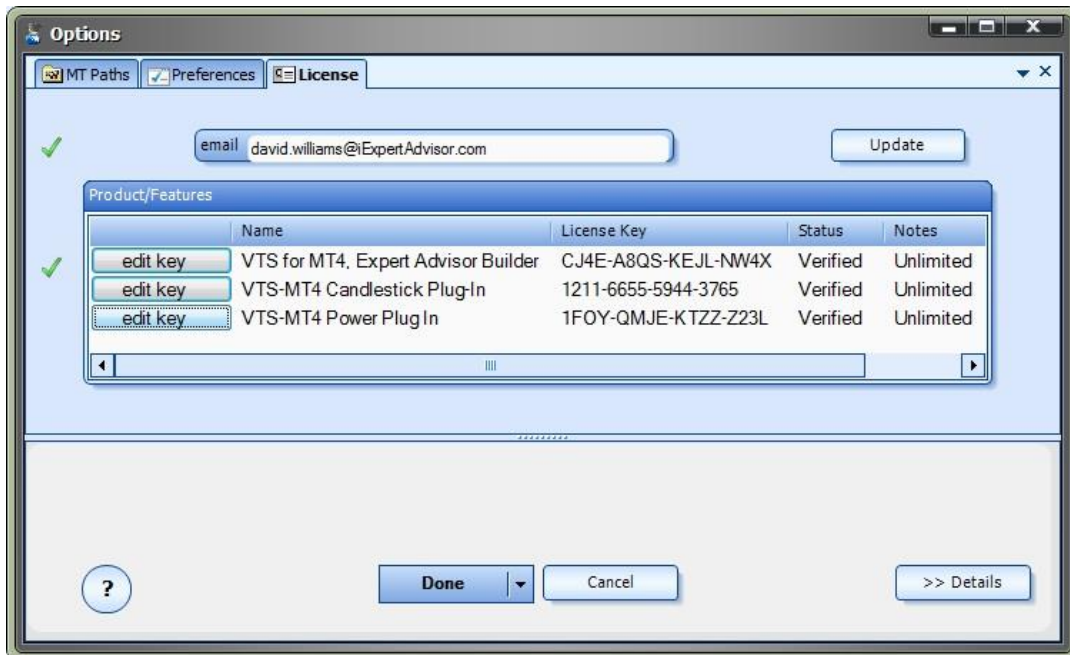


Enable the *Easy Email* Plug-in

You must enter your License key to enable the **Easy Email Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

- The **email** address is the email address used to purchase [VTS](#).
- The **License Key** is the key listed in the Members Area.
- The **Update** button is used to verify the email address and license key.
- The **edit key** button is used edit the key value.



Configure *Easy Email* Settings

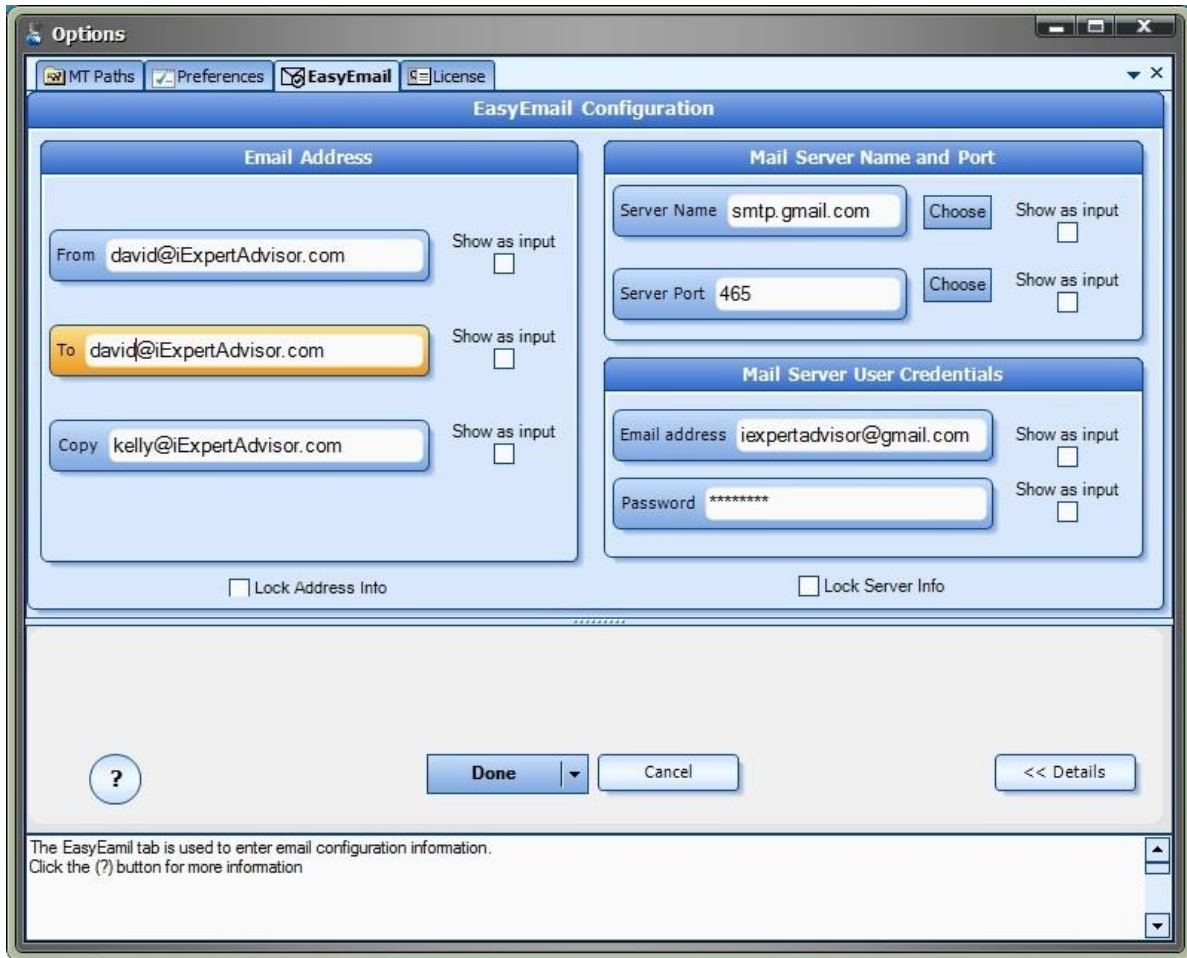
The settings for configuring *Easy Email* are found on the **Tools->Options->EasyEmail** tab.

NOTE: If you do not see the *EasyEmail* tab on the **Options** window, then the license has not been verified. Please check the **License** tab on the **Options** window.

Setting Name	Description	Example
From Address	The email address that your email will show that it was sent from. Normally you do not need to set this value when you compose an email within an email program, but this field is required to create a valid email. The From address can be the same as the To address.	david@iExpertAdvisor.com
To Address	The email address that your email will be sent to.	david@iExpertAdvisor.com
Copy Address	Optional email address to copy your email to.	kelly@iExpertAdvisor.com
Server Name	The smtp server name of your email provider. The Choose button lists the values for most popular email providers, however you can type directly in the box. The smtp server name can be found by (1) asking you email provider, or (2) a simple internet search.	smtp.gmail.com
Server Port	The port of your email providers smtp service. For secure connections, the value is usually 465. The Choose button lists the values for most popular email providers, however you can type directly in the box. The smtp port number can be found by (1) asking you email provider, or (2) a simple internet search.	465
Email Address	This is the email address that you have with your email provider. (Note: It can also be the To Address that emails are sent to).	iexpertadvisor@gmail.com
Password	The password used to connect with the Email Address with your email service provider. NOTE: This password is stored as clear text. Do not use the same password that you use for high security accounts.	hotdog

- The **Show as input** check boxes next to each setting allow the values to be changed when the Expert Advisor is attached to the chart.
- The **Lock** check boxes allow you to lock the information once it has been set. Usually this information only needs to be setup once.

This is screenshot of the *EasyEmail* Configuration window.



Using *Easy Email* from the Communication Manager

There are 3 methods available to send emails using the *Easy Email Plug-in*. Emails can be sent using:

- The [Communication Manager](#)
- Any [Element](#)
- The Toolbox function [fnSendEzEmail](#)

Note: Before any emails can be successfully sent, the [Easy Email Settings](#) must be configured.

This section describes how to use the **Communication Manager** to send emails.

The **Communication Manager** is used to send messages from an Expert Advisor based on the occurrence of an event. Complete information about the **Communication Manager** can be found [here](#).

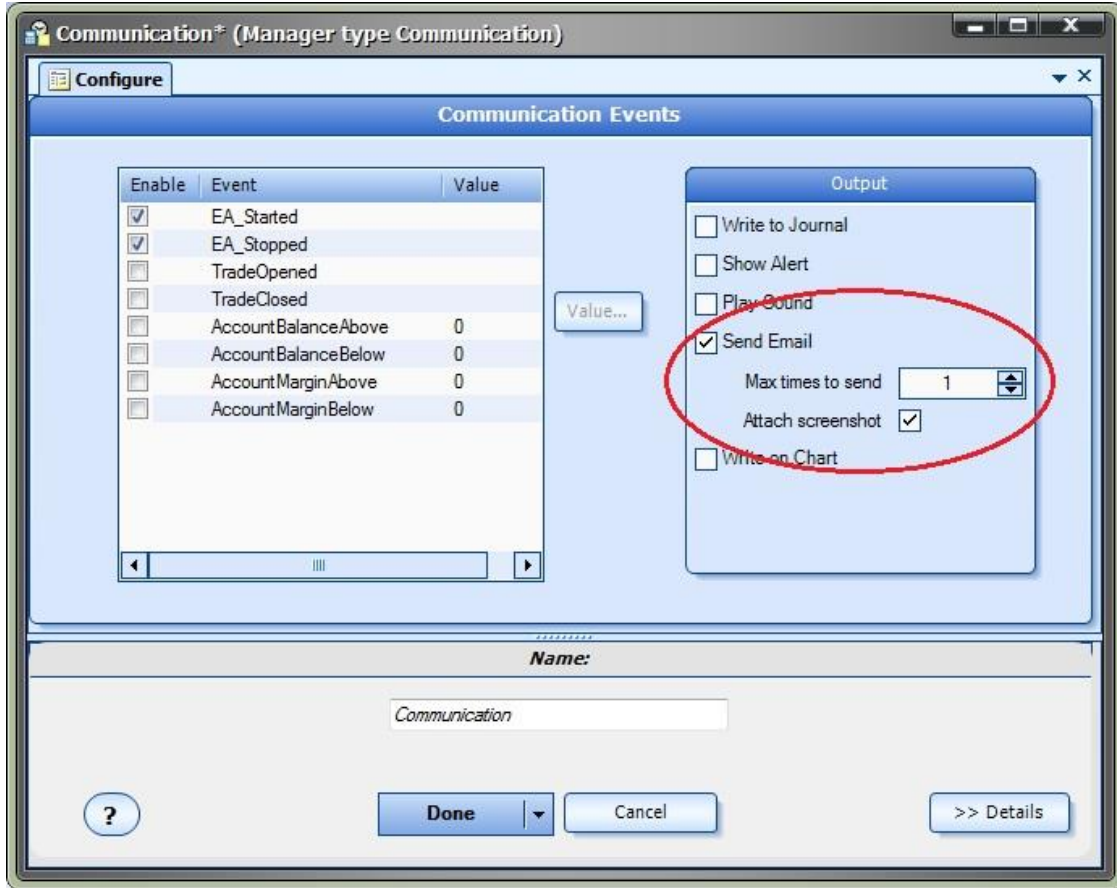
When the **Send Email** check box is checked, two additional options appear:

- **Max times to send**
- **Attach screenshot**

Max times to send controls how many times the same email will be sent. Normally an email only needs to be sent once. For example, if an email is sent when the account balance falls below \$1000, usually only one email message is required.

If there was no method to control *how many* emails were sent, the EA would continue to send emails as long as the condition was true (account below \$1000). This would quickly fill an inbox with the same email message.

When **Attach screenshot** is checked, a current screenshot of the chart that the EA is running on will be attached to the email.



Using *Easy Email* from any Element

There are 3 methods available to send emails using the **Easy Email Plug-in**. Emails can be sent using:

- The [Communication Manager](#)
- Any [Element](#)
- The Toolbox function [fnSendEzEmail](#)

Note: Before any emails can be successfully sent, the [Easy Email Settings](#) must be configured.

This section describes how to use the **Any Element** to send emails.

	<p>Selecting the configuration (+) button of an Element will display the Configuration Window.</p> <p>The Message Tab is found in the Configuration window.</p>
--	--

The **Message** Tab of an [Element](#) allows information to be sent from a running Expert Advisor. Complete information about the **Message Tab** can be found [here](#).

A message (in this case an email) is sent from an Element when the Expert Advisor's flow of execution follows a path that includes

the Element.

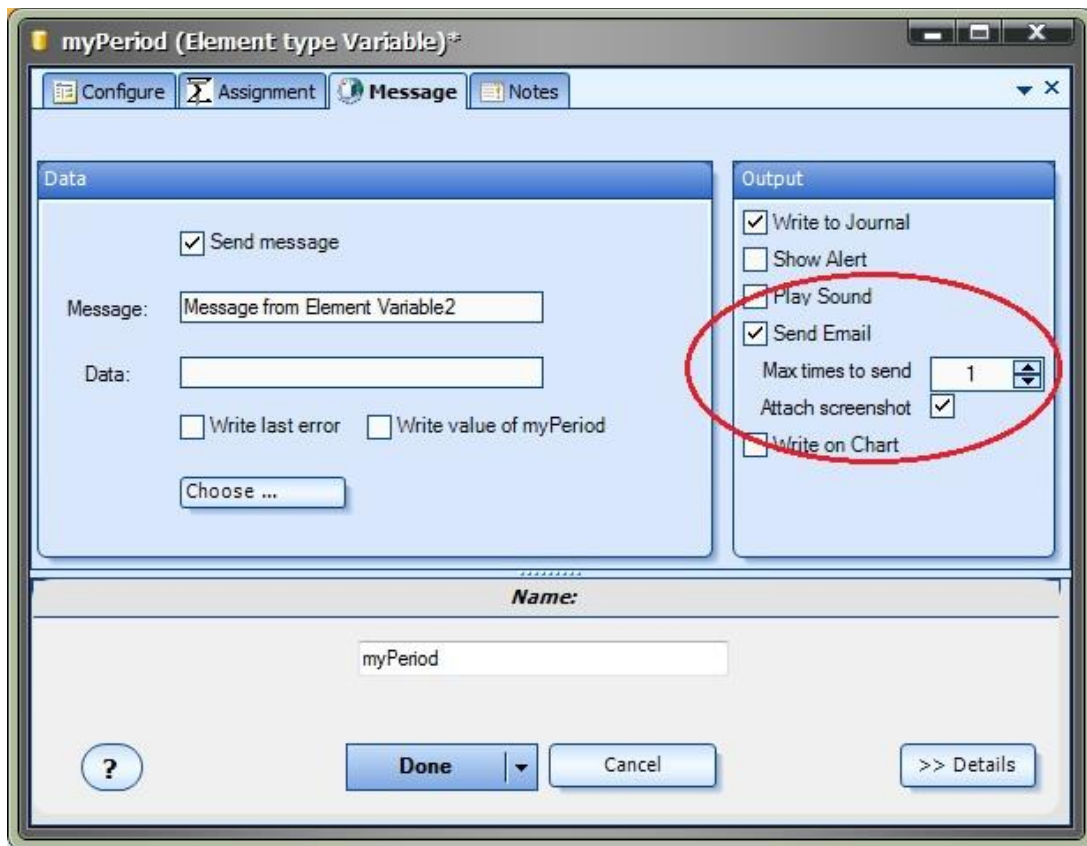
When the **Send Email** check box is checked, two additional options appear:

- **Max times to send**
- **Attach screenshot**

Max times to send controls how many times the same email will be sent. Normally an email only needs to be sent once. For example, if an email is sent when a Slow moving average has crossed up through a Fast moving average, usually only one email message is required.

If there was no method to control *how many* emails were sent, the EA would continue to send emails as long as the condition was true (the moving average cross). This would quickly fill an inbox with the same email message.

When **Attach screenshot** is checked, a current screenshot of the chart that the EA is running on will be attached to the email.



Using **Easy Email** with the `fnSendEzEmail` function

There are 3 methods available to send emails using the **Easy Email Plug-in**. Emails can be sent using:

- The [Communication Manager](#)
- Any [Element](#)
- The Toolbox function [fnSendEzEmail](#)

Note: Before any emails can be successfully sent, the [Easy Email Settings](#) must be configured.

This section describes how to use the `fnSendEzEmail` function to send emails.

The **Communication Manager** and the **Element Message Tab** are the easiest ways to use the Easy Email Plug-in to send emails.

This method is provided for the rare occasions when where drag and drop function control is required. Also, this technique serves as tutorial for creating and using [MQL functions](#).

Background

The MetaTrader platform supports sending emails using a simple built-in function named **SendMail**.

Unfortunately, the **SendMail** function only works with non-secure email servers, and today almost all email providers use secure connections for sending and receiving emails.

In order to establish a secure email connection, additional code must be created to manage an SMTP connection. SMTP stands for Simple Mail Transfer Protocol and is an internet standard for sending and receiving emails.

The SMTP connection code resides in a Dynamic Link Library (DLL). (In this case, the DLL is named *ezemail.dll*)

For an Expert Advisor to use the functionality of a DLL, the DLL must be described and referenced in the MQL code, and the DLL must reside in the *libraries* folder of the MetaTrader platform.

VTS automatically copies the DLL to the *libraries* folder and adds all required MQL code to use the **Easy Email** functions.

The function **fnSendEzEmail** is available from the [Toolbox](#) under the **Common** menu.

The **fnSendEzEmail** function is an [MQL function](#).

Because **fnSendEzEmail** is an MQL function, the MQL source code of this function is available from the [Toolbox](#) under the **Advanced->MQL** menu.

The name of the MQL source code function has been changed to **fnSendEzEmailMql** to create a distinction between the two functions.

When you create an MQL function, the name of the function within the MQL code will be the name of the function on the **Advanced->MQL** menu.

The name of the MQL Element in which the MQL code was entered can be named any available name, for example *MyMql*.

The function on the **Advanced->MQL** menu holds the MQL source code and can be edited at any time.

The function *MyMql* represents a call to the function on the **Advanced->MQL** menu. It does not contain the MQL code, only a call to the underlying function with the parameters set specifically.

For example, the function **fnSendEzEmail** can be dragged onto the [Drawing Pad](#), set the **To** parameter to a work email address and named *SendEmailToWork*.

Another copy can be dragged onto the Pad, the **To** parameter set to a home email address and name *SendEmailHome*.

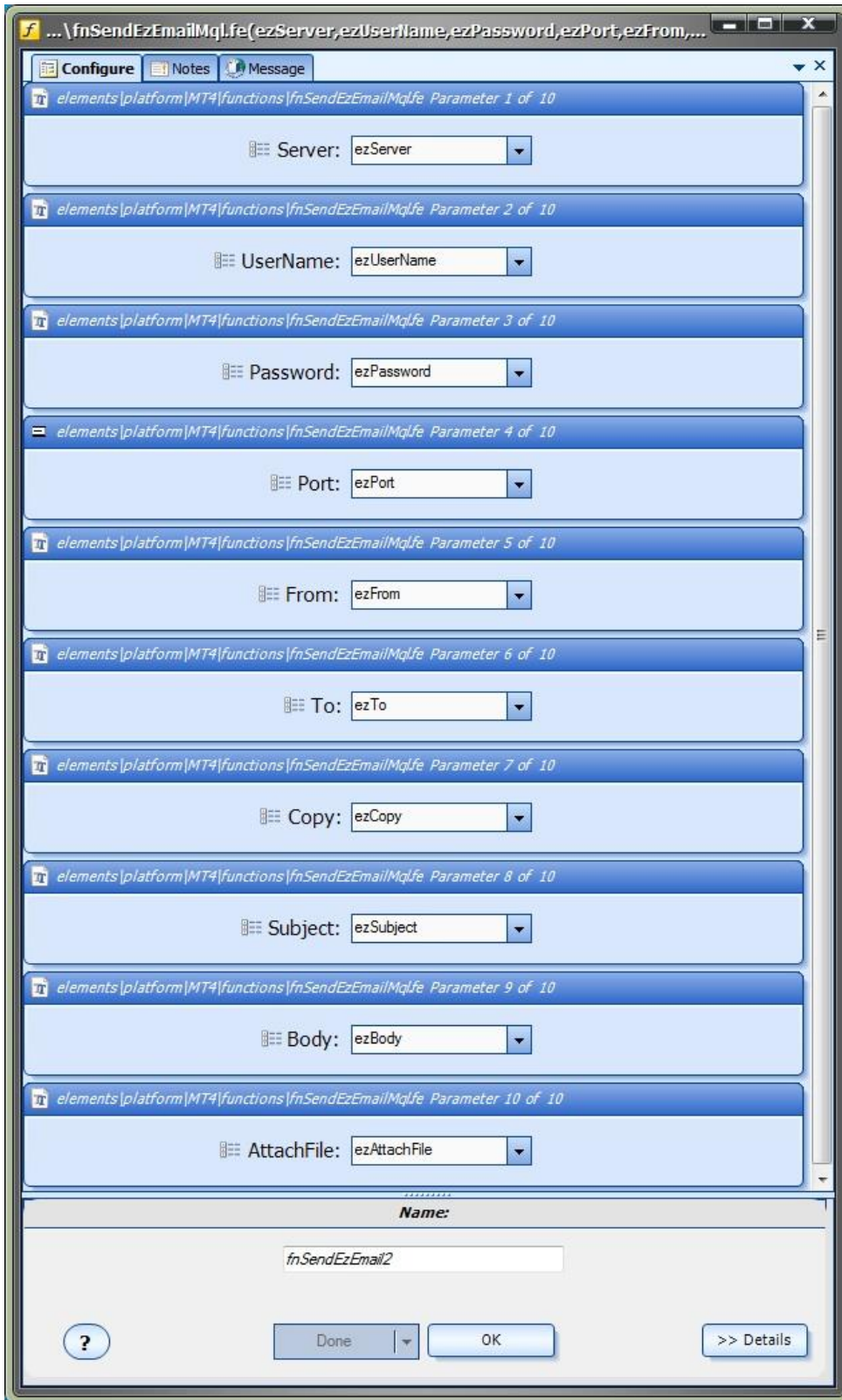
More information about MQL function is described [here](#).

The parameters of the **fnSendEzEmail** function are described below.

NOTE: There is no count available to control how many times an email is sent when using the **fnSendEzEmail** function. Care must be exercised to ensure a large number of emails are not sent.

Name	Data Type	Description
Server	string	The smtp server name of your email provider. The smtp server name can be found by (1) asking you email provider, or (2) a simple internet search.
User Name	string	This is the email address that you have with your email provider. (Note: It can also be the To Address that emails are sent to).
Password	string	The password used to connect with the Email Address with your email service provider. NOTE: This password is stored as clear text. Do not use the same password that you use for high security accounts.
Port	integer	The port of your email providers smtp service. For secure connections, the value is usually 465. The smtp port number can be found by (1) asking you email provider, or (2) a simple internet search.
From	string	The email address that your email will show that it was sent from. Normally you do not need to set this value when you compose an email within an email program, but this field is required to create a valid email. The From address can be the same as the To address.
To	string	The email address that your email will be sent to.
Copy	string	Optional email address to copy your email to.
Sub	string	The subject that will appear in the subject line of the email.

j e c t	i n g	
B o d y	s t r i n g	The body of the email (the text message).
A t t a c h F i l e	s t r i n g	The full path of a file to be attached to the email.



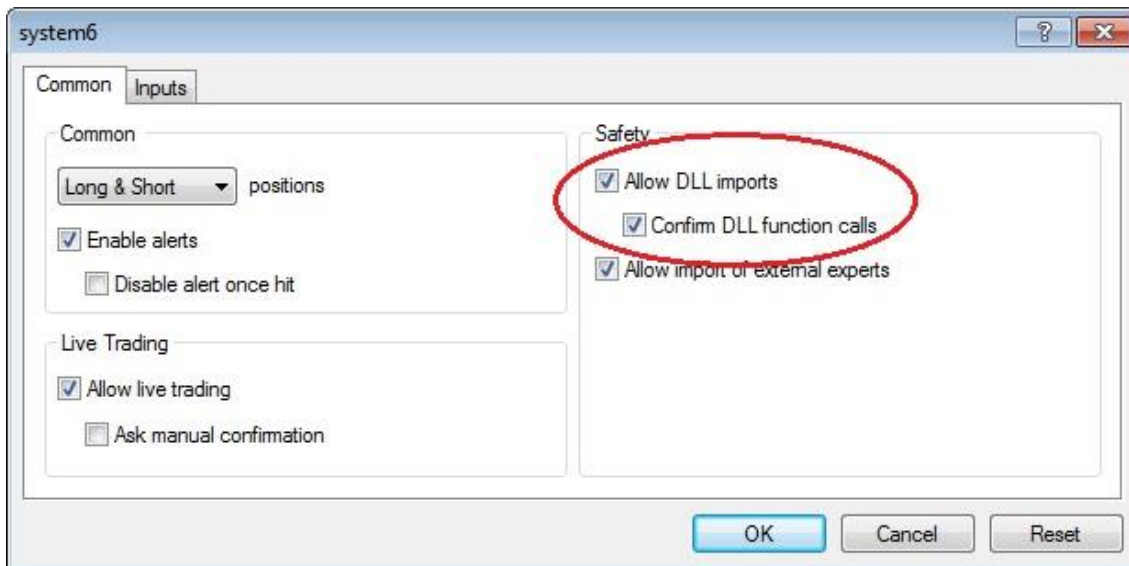
Testing the *Easy Email* Connection

After *Easy Email* has been [Enabled](#) and [Configured](#) it should be tested on a Demo account.

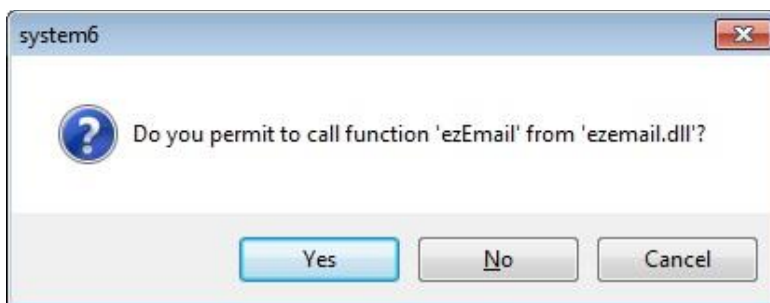
- From the [Welcome](#) screen, click the **EA Builder** button to create a new system.
- Uncheck all of the **Add Drawing** check boxes so that no Trading Drawings are created.
- This will create a single system drawing with a Start and an End [Element](#).
- Use the [Communication Manager](#) method to send an email:
 - Check **EA_Started**
 - Check **Send Email**
 - Leave **Max Times to send** as 1
 - Check **Attach screenshot**
- Click **Done** to save the changes and close the window.
- Click **Build** to build the Expert Advisor.
- Open your MetaTrader platform and attach the EA to a price chart.

When the EA is attached to a chart, in the **Common** tab, in the **Safety** section, there is a checkbox to **Allow DLL imports** and another to **Confirm DLL function calls**.

- **Allow DLL imports** must be checked
- **Confirm DLL function calls** should be unchecked



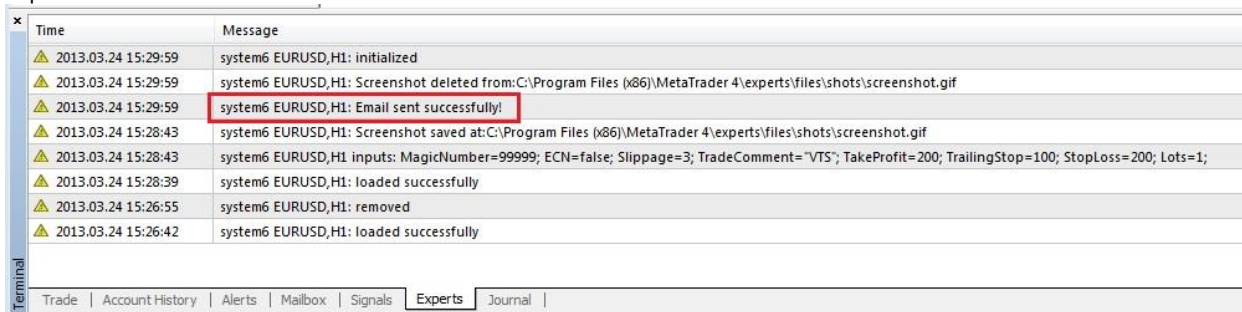
If the **Confirm DLL function calls** is unchecked, each call to the DLL function **ezemail** will need to be manually permitted.



- Select the [Experts Tab](#) in the Terminal window of the platform.

Shortly after the EA has been attached to a price chart an email should be sent from the platform to the specified **To** address.

If the email is sent successfully, the message **Email send successfully!** is written to the Experts tab:



If the email is not sent successfully, the message **Email error. See log file at: experts\files\ezemail.log** is written to the Experts tab:

The full path of the **experts\files\ezemail.log** file is found under the MetaTrader platform installation folder. For example:

C:\Program Files (x86)\MetaTrader 4\experts\files\ezemail.log

- The file ezemail.log is a plain text file and can be opened with any text editor including notepad.
- If the email was unsuccessfully, the file ezemail.log will contain details about the error.

Sample errors using Gmail

Error	Error Message
Incorrect UserName	Error with user authentication;Server says: 535-5.7.1 Username and Password not accepted. Learn more at 535 5.7.1 http://support.google.com/mail/bin/answer.py?answer=14257_dz9sm16749589vdc.4 - gsmt
Incorrect Password	Error with user authentication;Server says: 535-5.7.1 Username and Password not accepted. Learn more at 535 5.7.1 http://support.google.com/mail/bin/answer.py?answer=14257_hi4sm13011409jgc.6 - gsmt
Incorrect Server Name (removed the .com)	Error with connecting server;current server: smtp.gmail ;getaddrinfo failed for smtp.gmail 465, error: 0x00002af9
Incorrect Port	Error with connecting server;current server: smtp.gmail.com;connect failed, error: 0x0000274c

FX TrendLine Plug-in

Requires VTS-Connect minimum version **4.0.0.40**

The **FX TrendLine Plug-in** allows any Expert Advisor to detect if one or more manual or automatically drawn trend lines have been broken.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the *FX TrendLine* Plug-in

You must enter your License key to enable the ***FX TrendLine Plug-in***. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

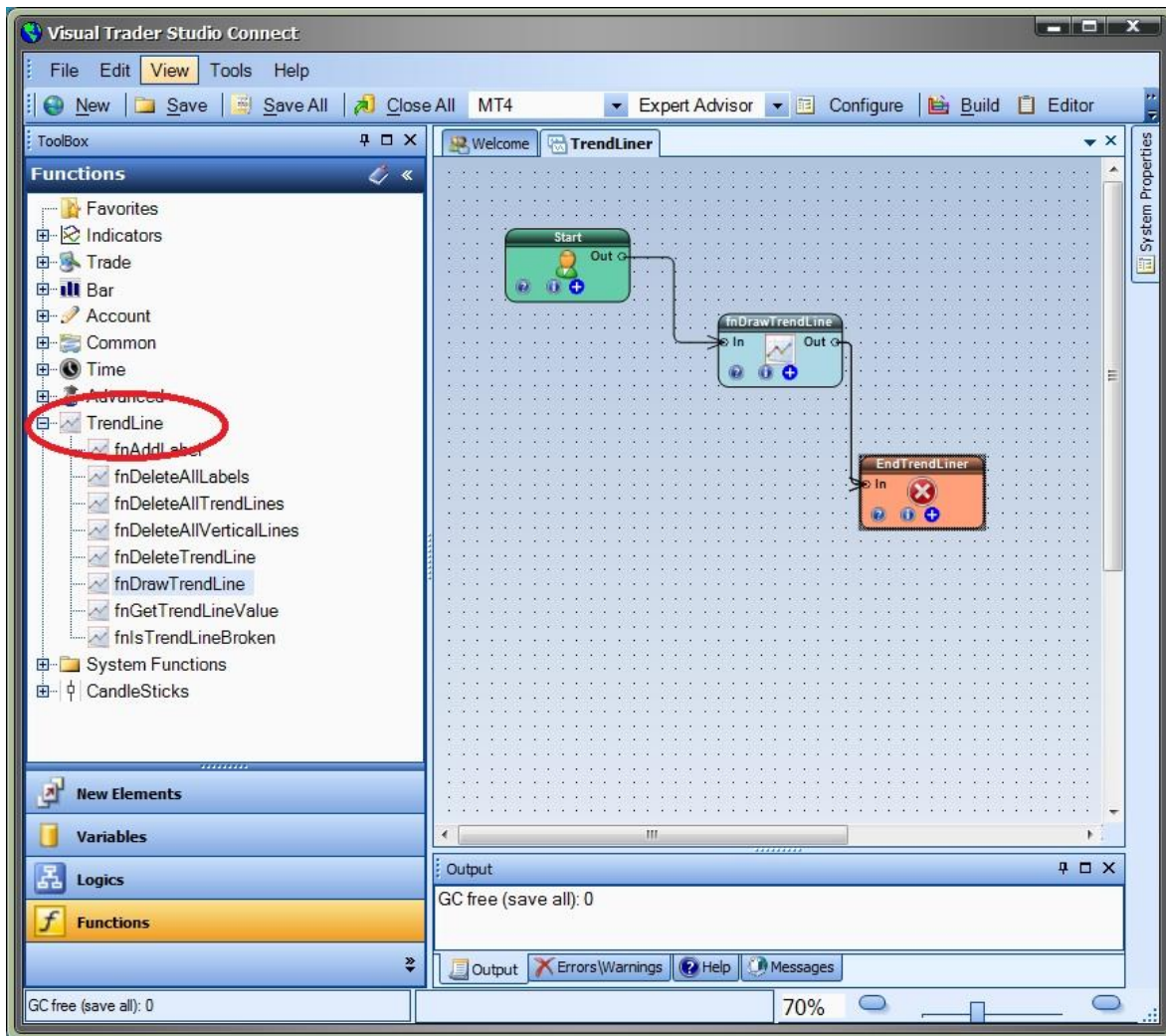
- The **email** address is the email address used to purchase [VTS](#).
- The **License Key** is the key that is sent to the email address.
- The **Verify** button is used to verify the email address and license key.
- The **Add** button is used to add a key.
- The **Remove** button is used to remove a key.
- Double-click a key to edit its value.



TrendLines Functions in the Toolbox

Once enabled, the TrendLine functions are available in the [Toolbox](#) Function tab under the TrendLines menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.



TrendLine Functions

TrendLines Functions

The TrendLine function library include these functions:

- [fnDrawTrendLine](#)
 - Used to programatically draw a trend line on a price chart.
- [fnIsTrendLineBroken](#)
 - Used to determine if a price value has broken through a trend line.
- [fnGetTrendLineValue](#)
 - Used to get the value of a trend line at any point on the line.
- [fnAddLabel](#)
 - Used to add a label to a price chart.
- [fnDeleteTrendLine](#)
 - Used to programatically delete a trend line from a price chart.

- [fnDeleteAllTrendLines](#)
 - Used to programatically delete all trend lines from a price chart.
- [fnDeleteAllLabels](#)
 - Used to programatically delete all labels from a price chart.
- [fnDeleteAllVerticalLines](#)
 - Used to programatically delete all vertical lines from a price chart.

fnDrawTrendLine

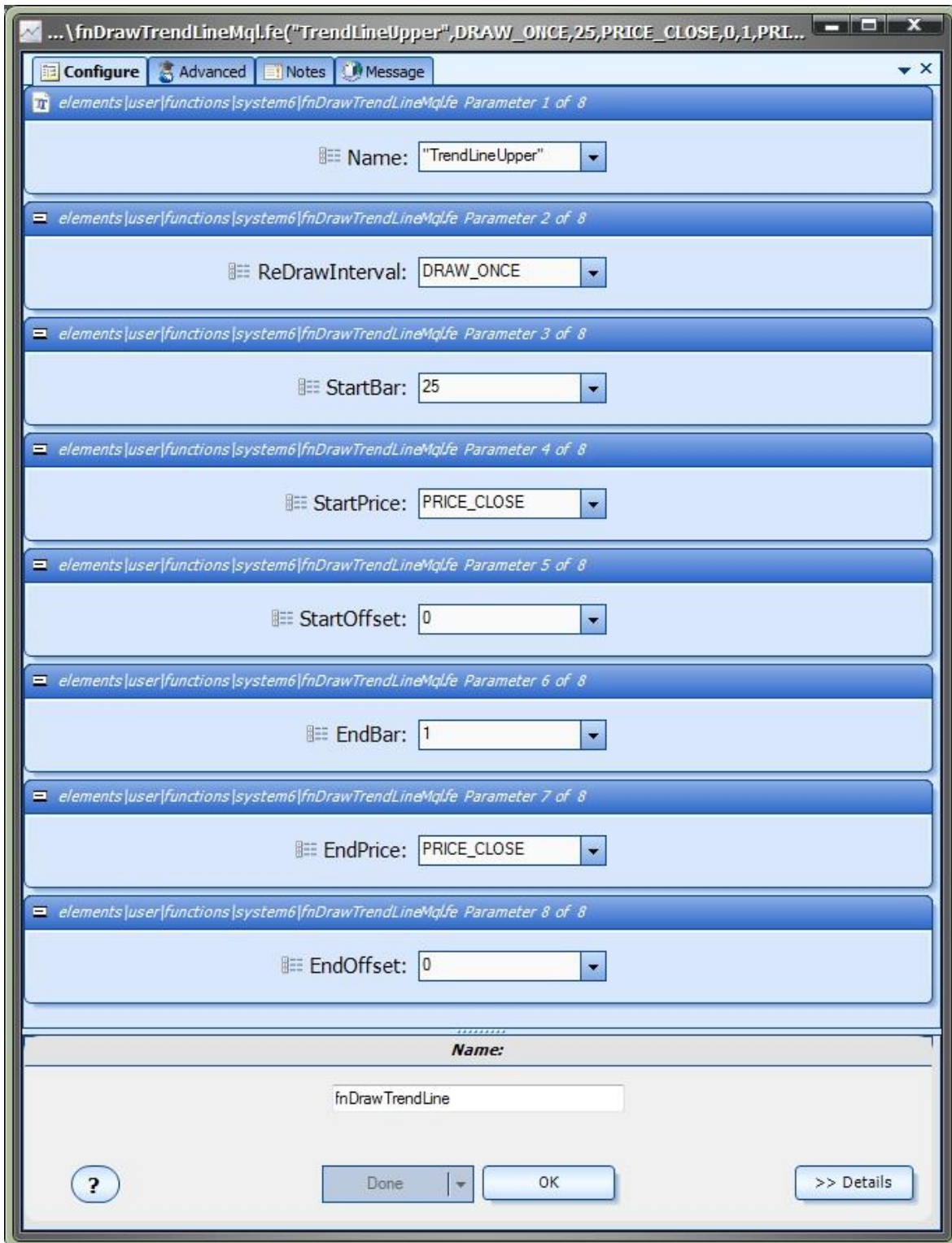
The trend line function **fnDrawTrendLine** is used to programatically draw a trend line on a price chart.

After the **fnDrawTrendLine** function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
Name	string	The name of the trend line. All trend lines are identified by name. The name should be unique. The pull-down menu provides sample names, but the text can be overwritten to any allowable name. The name must be surrounded by double quotes, for example "myName". Avoid using blank spaces in the name: It may cause unpredictable behaviour.
ReDrawInterval	DRAW type (integer)	The interval at which the trend line is redrawn. The pull-down menu offers these choices: RD_ONCE : draw the line one time only RD_DAILY : redraw the line on each new day RD_HOURLY : redraw the line on each new hour RD_NEWBAR : redraw the line on each new bar RD_NEWTICK : redraw the line on each new tick
StartBar	integer	The horizontal location (bar) on the price chart where the trend line begins. Lines are drawn left to right, so the StartBar is to the left of the EndBar on a price chart. The StartBar is a larger number than the EndBar.
StartPrice	double	The vertical location (price) where the trend line begins. The StartPrice is the price value of the StartBar. The pull-down menu offers these choices: PRICE_CLOSE PRICE_OPEN PRICE_HIGH PRICE_LOW PRICE_MEDIAN PRICE_TYPICAL PRICE_WEIGHTED
StartOffset	integer	A positive or negative offset of the StartPrice, where 1 increment equals 1 point. This value moves the start point of the line up or down vertically on the price chart.
EndBar	integer	The horizontal location (bar) on the price chart where the trend line ends. Lines are drawn left to right, so the EndBar is to the right of the StartBar on a price chart. The EndBar is a smaller number than the StartBar.
EndPrice	double	The vertical location (price) where the trend line ends. The EndPrice is the price value of the EndBar. The pull-down menu offers these choices: PRICE_CLOSE PRICE_OPEN PRICE_HIGH PRICE_LOW PRICE_MEDIAN PRICE_TYPICAL PRICE_WEIGHTED
EndOffset	integer	A positive or negative offset of the EndPrice, where 1 increment equals 1 point. This value moves the end point of the line up or down vertically on the price chart.
LineColor*	Color	The color of the trend line.
DrawLabel*	boolean	Draws a label on the chart that contains the trend line's name.
DrawMarks*	boolean	Draws vertical lines on the chart on the Start and End bar locations.

* These parameters are found on the [Advanced](#) tab.



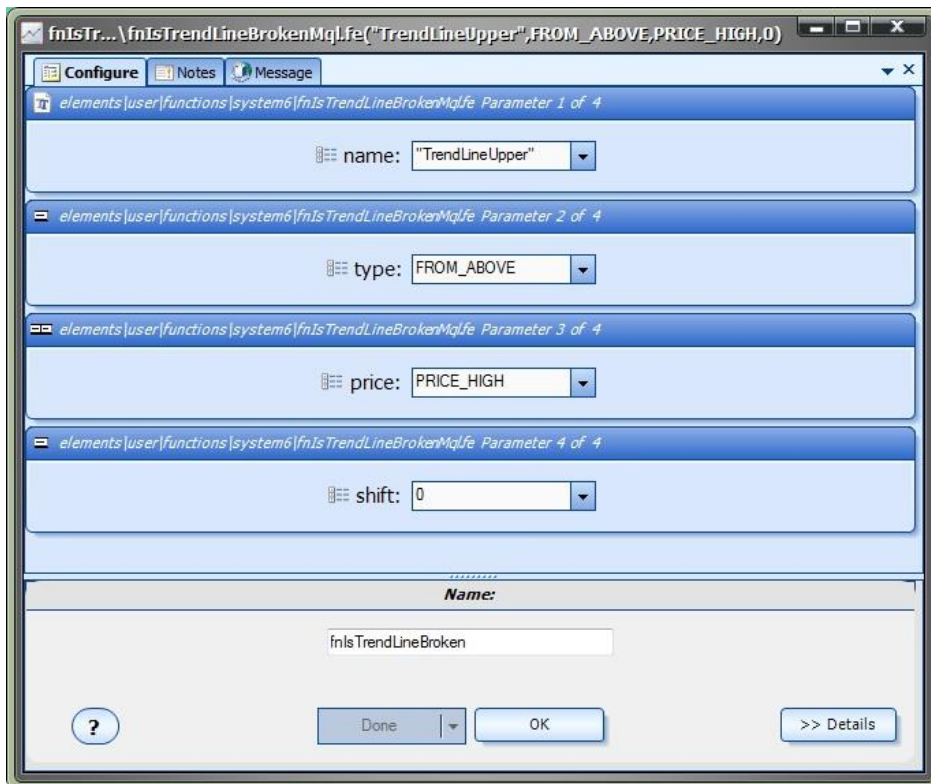
fnIsTrendLineBroken

The *fnIsTrendLineBroken* function is used to determine if a price value has broken through a trend line.

After the *fnIsTrendLineBroken* function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
name	string	The name of the trend line. All trend lines are identified by name. The name should be unique. The pull-down menu provides sample names, but the text can be overwritten to any allowable name. The name must be surrounded by double quotes, for example "myName". Avoid using blank spaces in the name: It may cause unpredictable behaviour.
type	FROM value (integer)	The direction from which the price value was broken. The pull-down menu offers these choices: FROM_ABOVE : The price value broke the trend line from above the line. FROM_BELOW : The price value broke the trend line from below the line.
price	double	The price value to use to determine if the price has broken through the trend line. The pull-down menu offers these choices: Ask Bid PRICE_CLOSE PRICE_OPEN PRICE_HIGH PRICE_LOW PRICE_MEDIAN PRICE_TYPICAL PRICE_WEIGHTED
shift	integer	The candle index on the price chart of where to test if the trend line has been broken. Zero is the currently forming candle, one is one candle to the left, etc.



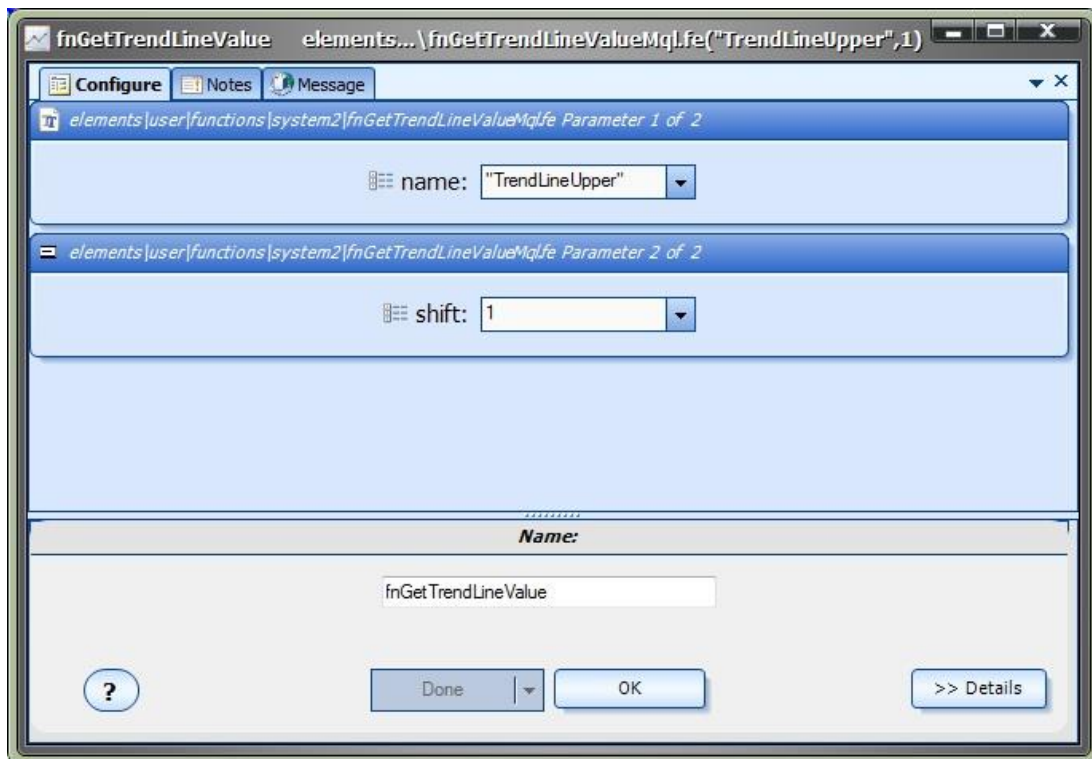
fnGetTrendLineValue

The **fnGetTrendLineValue** function is used to get the value of a trend line at any point on the line.

After the **fnGetTrendLineValue** function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
name	string	The name of the trend line. All trend lines are identified by name. The name should be unique. The pull-down menu provides sample names, but the text can be overwritten to any allowable name. The name must be surrounded by double quotes, for example "myName". Avoid using blank spaces in the name: It may cause unpredictable behaviour.
shift	integer	The candle index on the price chart of where to capture the trend line's value. Zero is the currently forming candle, one is one candle to the left, etc.



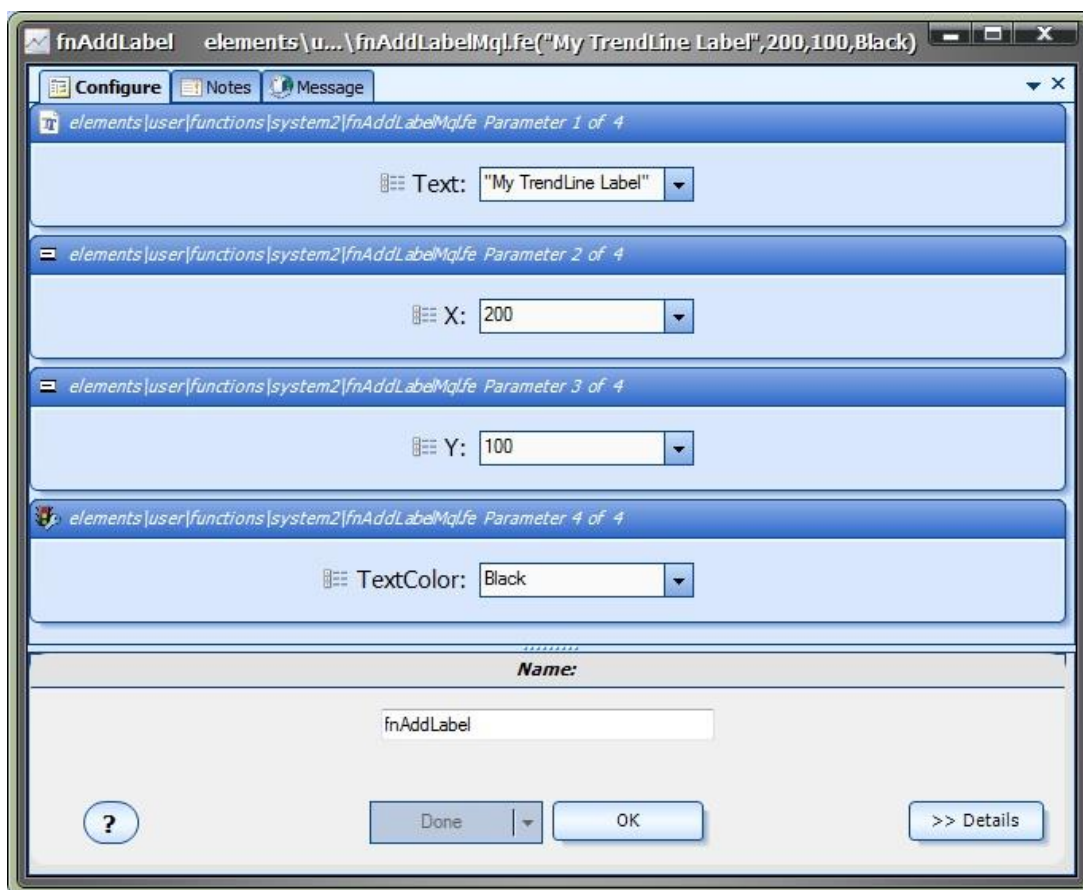
fnAddLabel

The **fnAddLabel** is used to add a label to a price chart.

After the **fnGetTrendLineValue** function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
Text	string	The text of the label to add to the price chart. The text must be surrounded by double quotes, for example "myName".
X	integer	The horizontal offset from the top left corner of the chart. The X value is zero at the far left corner of the chart and increases as you move to the right. The default value is 200.
Y	integer	The vertical offset from the top left corner of the chart. The Y value is zero at the top of the chart and increases as you move down. The default value is 100.
TextColor	Color	The color of the text.



fnDeleteTrendLine

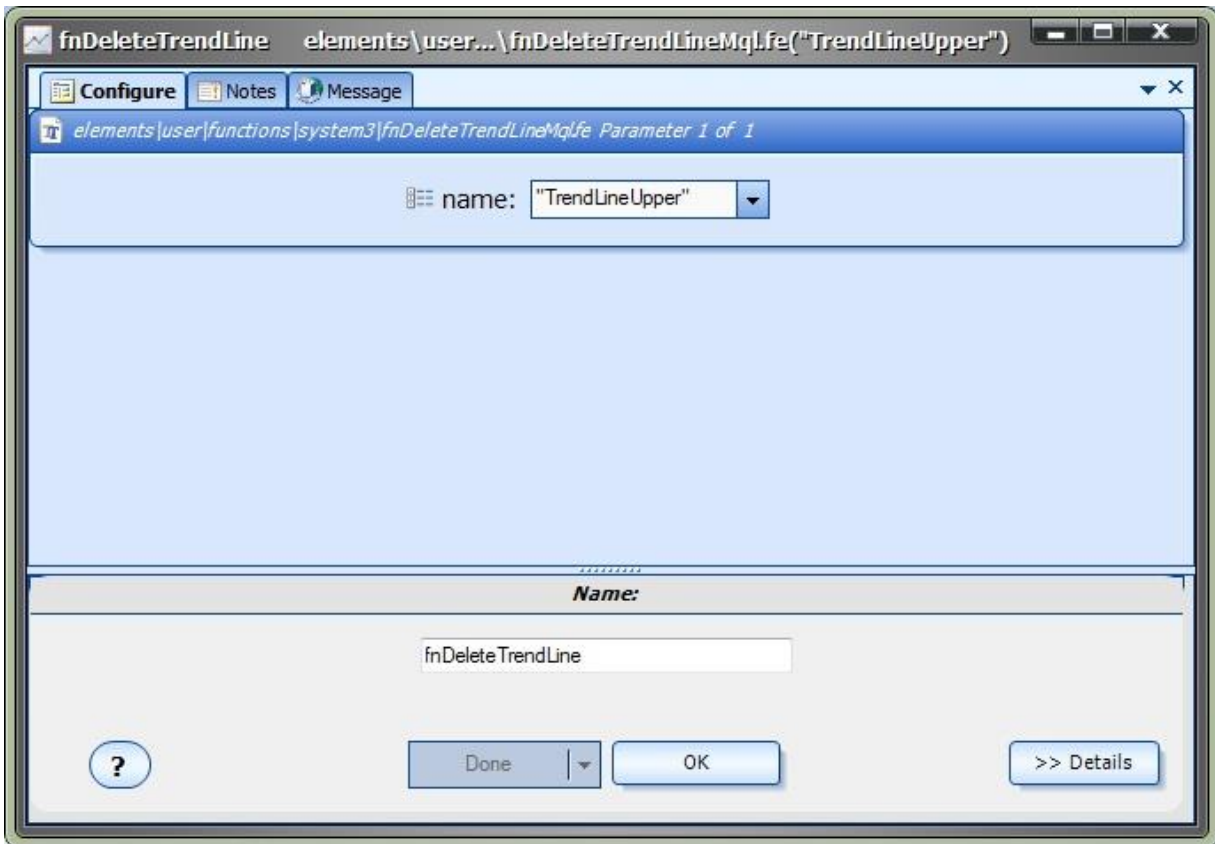
The **fnDeleteTrendLine** function is used to programatically delete a trend line from a price chart.

After the **fnDeleteTrendLine** function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
name	string	The name of the trend line. All trend lines are identified by name. The name should be unique. The pull-down menu provides sample names, but the text can be overwritten to any allowable name. The name must be surrounded by double quotes, for example "myName". Avoid using blank spaces in the name: It may cause unpredictable behaviour.

Note: Any and all trend lines that match the name are deleted from the price chart. This includes trend lines created both manually and programatically.



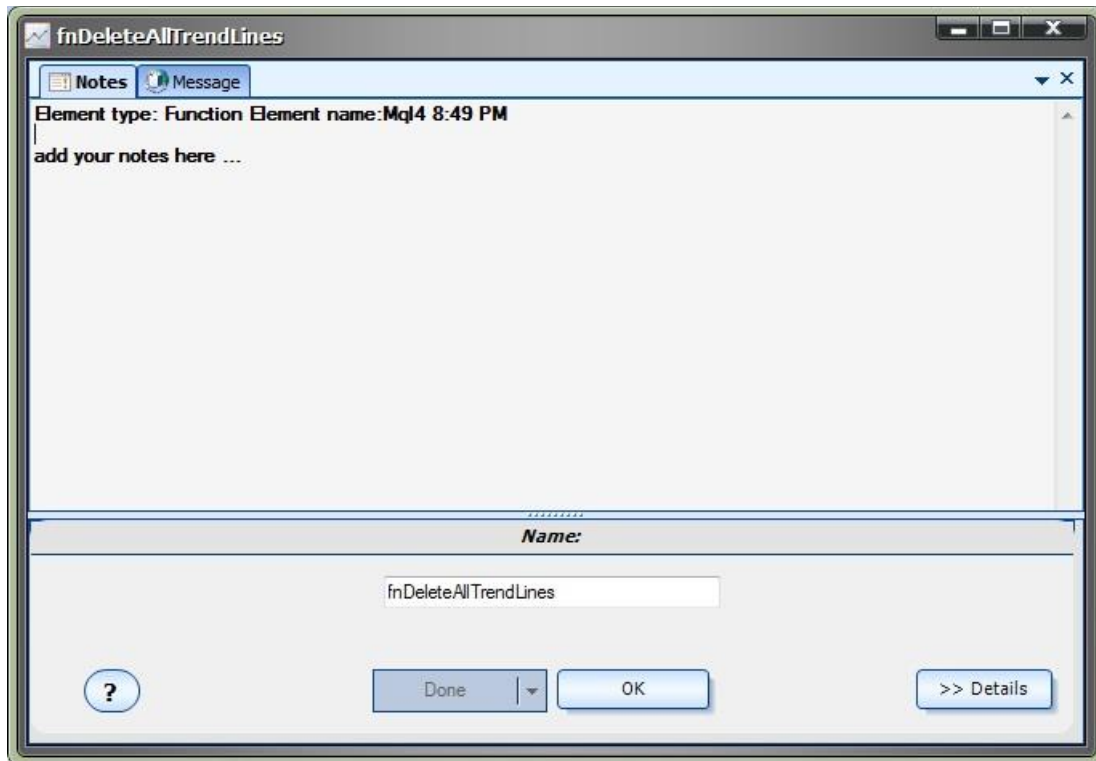
fnDeleteAllTrendLines

The **fnDeleteAllTrendLines** function is used to programmatically delete all trend lines from a price chart.

After the **fnDeleteAllTrendLines** function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The **fnDeleteAllTrendLines** function does not require any parameters.

Note: Any and all trend lines are deleted from the price chart. This includes trend lines created both manually and programmatically.



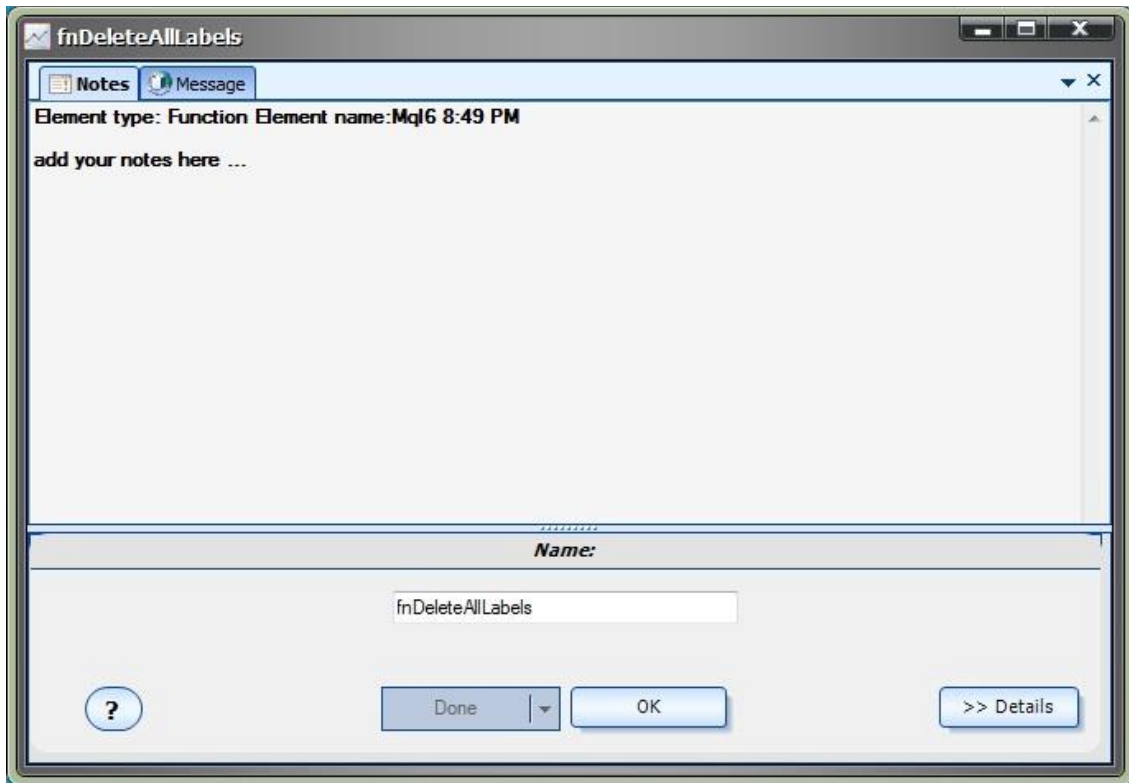
fnDeleteAllLabels

The **fnDeleteAllLabels** is used to programmatically delete all labels from a price chart.

After the **fnDeleteAllLabels** function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The **fnDeleteAllLabels** function does not require any parameters.

Note: Any and all labels are deleted from the price chart. This includes labels created both manually and programmatically.



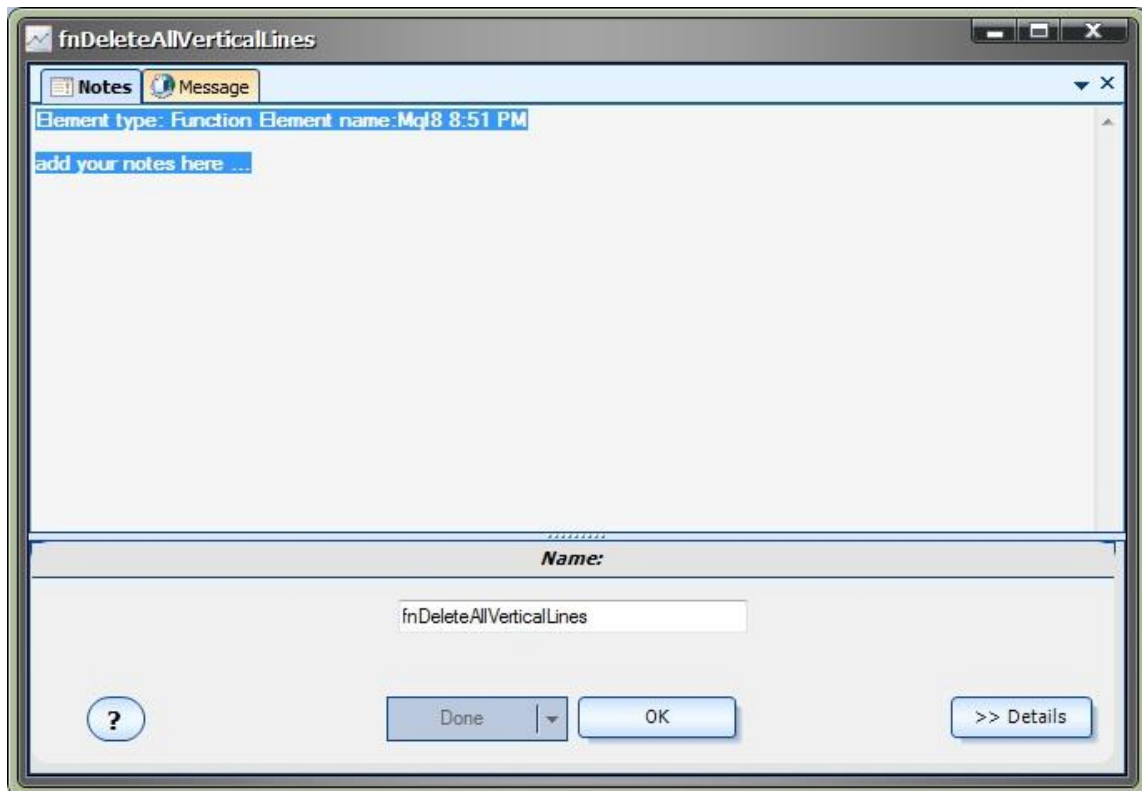
fnDeleteAllVerticalLines

The **fnDeleteAllVerticalLines** function is used to programmatically delete all vertical lines from a price chart.

After the **fnDeleteAllVerticalLines** function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The **fnDeleteAllVerticalLines** function does not require any parameters.

Note: Any and all vertical lines are deleted from the price chart. This includes vertical lines created both manually and programmatically.

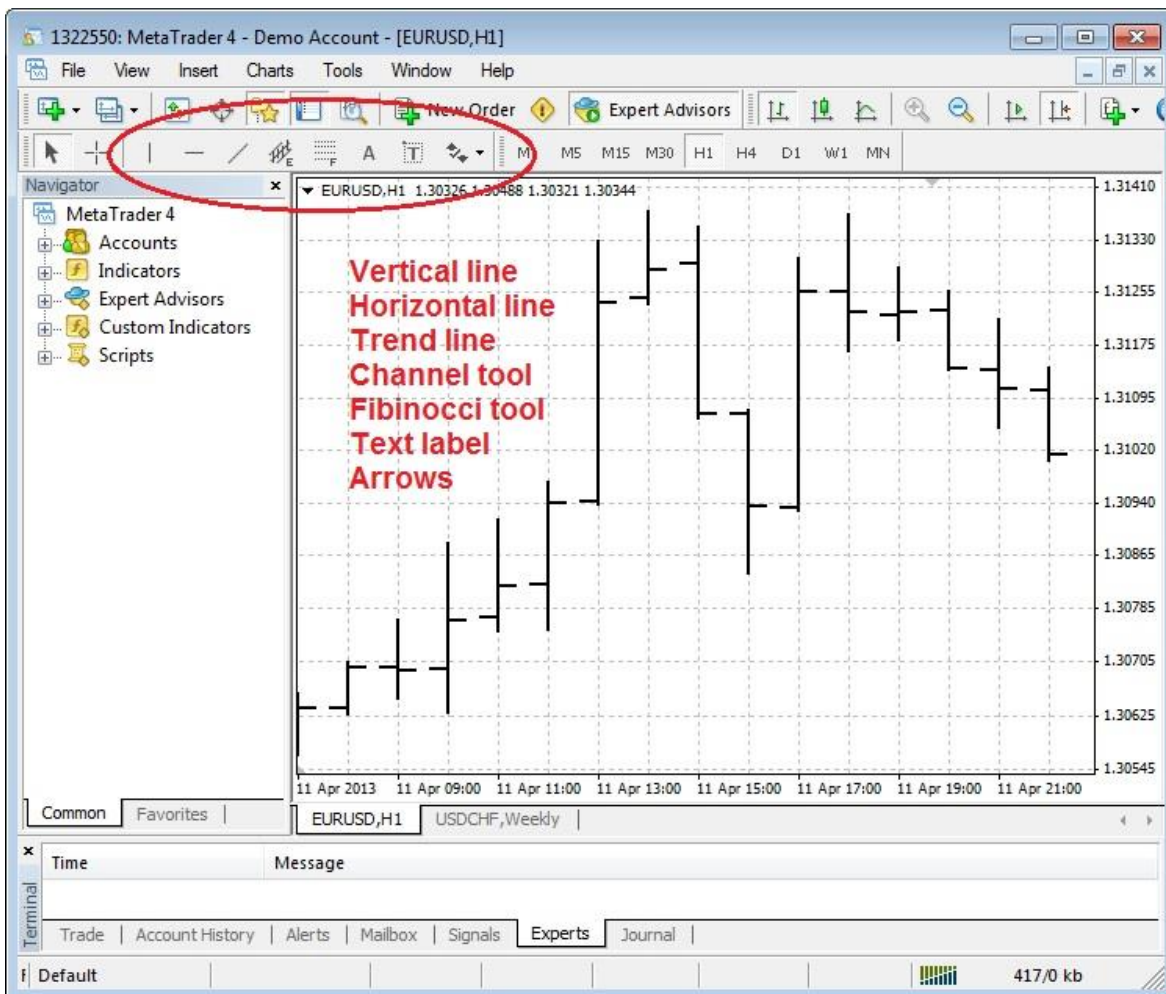


Using FX TrendLine: Objects on the MetaTrader platform

Every line, label and arrow that you see on a MetaTrader price chart is referred to as an Object. The MetaTrader platform provides a set of MQL functions for programmatically creating and deleting Objects. The Object functions are available from the VTS [Function Toolbox](#) under the Advanced->Object menu. The Object functions are on the Advanced menu because they require advanced MQL knowledge and can be difficult to use.

Objects can be manually created using the toolbar on the top of the MetaTrader platform. The objects that can be created from the MT tool bar are:

- Vertical line
- Horizontal line
- Trend line
- Channels line
- Fibonacci lines
- Text
- Arrows



Manually drawing a trend line

To manually draw a trend line on a MetaTrader price chart, click the trend line button, then click the start location on the chart, hold the mouse down while moving to the end location and release the mouse.

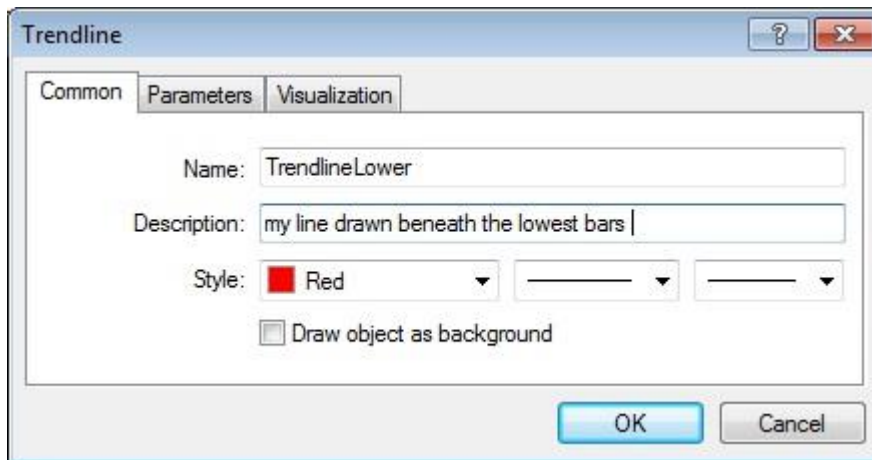
Once drawn, the location and angle of the trend line can be adjusted by double-clicking the line to select it and then moving it with the mouse.

When a trend line is manually created, it is given a generated name by the MetaTrader platform. For example, it may be named "Trendline 29693".

The **FX TrendLine Plug-in** identifies trend lines by their name. To change the name of a manually drawn trend line, double-click the trend line to select it, right-click the mouse and select "Trendline properties ...".

This will allow changing the trend line properties including the name.

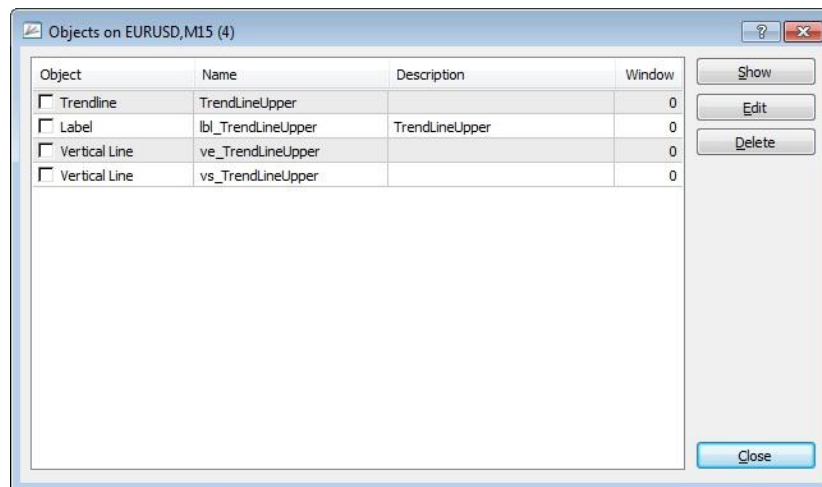
Note: The name of each trend line should be unique. Do not place more than one trend line on the same chart with the same name. However, it's permissible to have the same name trend line on different charts.



Removing a trend line (or any other Object)

To manually remove a single trend line from a MetaTrader price chart, double-click the trend line to select it, and press the delete key on the keyboard.

Objects on a chart can be managed by going to **Charts->Objects->Object List**. This is the best method to delete many objects at once.



Building an Expert Advisor to manage a manually drawn trend line

The **FX TrendLine Plug-in** can monitor any trend line on a chart. It does this by searching for a trend line by name.

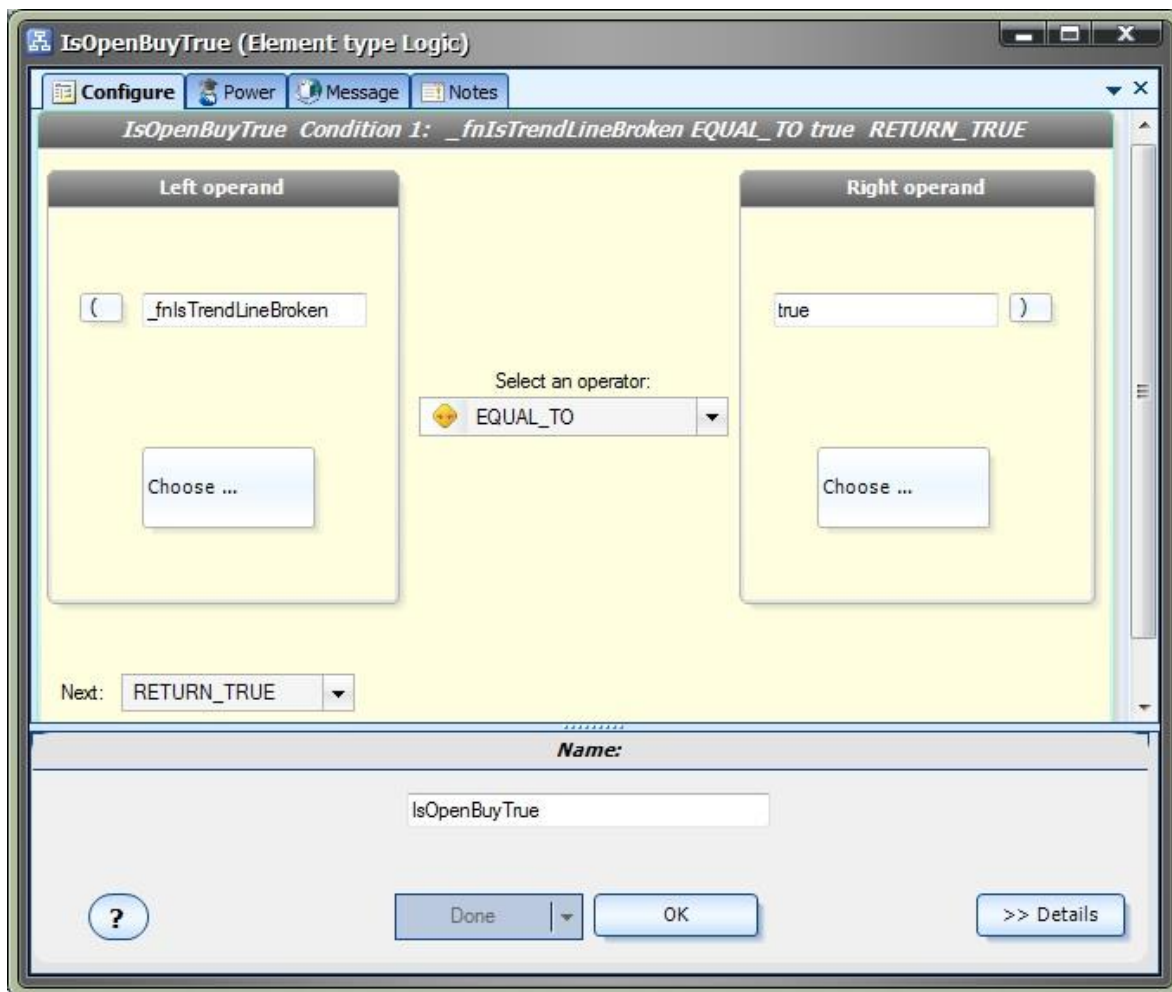
After you manually [draw your trend line](#) on your price chart, record the exact name of the trend line. This name will be referenced in your Expert Advisor by the [fnIsTrendLineBroken](#).

To build an Expert Advisor that opens a BUY trade when a trend line has been broken, drag and drop the TrendLine function **fnIsTrendLineBroken** on to the **OpenBuyOrder** drawing pad and connect it before the [Logic](#) element **IsOpenBuyTrue**.

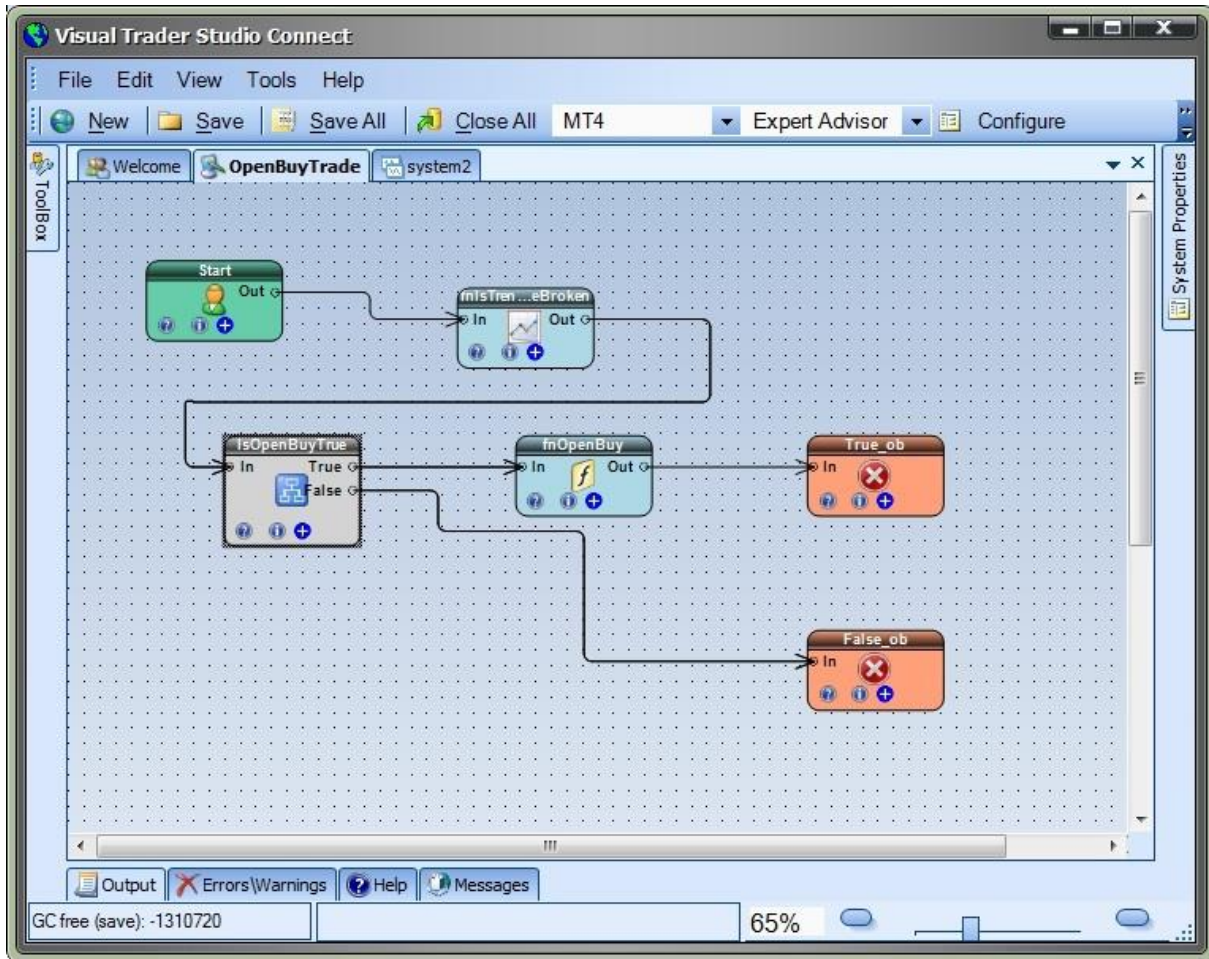
Set the [parameters](#) of the **fnIsTrendLineBroken** function:

- **name**: set the value of the **name** parameter to the exact name of the trend line. For example "mytrendline".
- **type**: set the value of the **type** parameter to the direction to test for the trend line break, either FROM_ABOVE or FROM_BELOW.
- **price**: set the value of the **price** parameter to one of the [price constants](#), or **Bid** or **Ask**.
- **shift**: set the value of the **shift** parameter to the index value of the candle (or bar) to test for the break.
 - Normally, the **shift** value is set to 0 and the price value is set to **Bid** or **Ask**. This will test the latest price against the trend line on the far right edge of the chart.
 - Alternatively, any candle on the chart can be tested for a break.

After connecting and configuring the **fnIsTrendLineBroken** function, configure the **IsOpenBuyTrue** Logic element to test for the break. The function **IsOpenBuyTrue** returns a value of **true** if/when the line is broken.



The full **OpenBuyOrder** drawing is shown below.



Building an Expert Advisor to draw and manage a trend line

The **FX TrendLine Plug-in** can programatically draw a trend line on a price chart using the **fnDrawTrendLine** function.

To build an Expert Advisor that opens a BUY trade when a programatically drawn trend line has been broken, drag and drop the TrendLine function **fnDrawTrendLine** on to the **OpenBuyOrder** drawing pad and connect it before the [Logic](#) element **IsOpenBuyTrue**.

Set the [parameters](#) of the **fnDrawTrendLine** function:

- **Name:** set the value of the **name** parameter to the name of the trend line. For example "mytrendline". This name will be referenced later in the **fnIsTrendLineBroken** function.
- **ReDrawInterval:** set the value of the **ReDrawInterval** parameter to desired redraw value:
 - RD_ONCE : draw the line one time only
 - RD_DAILY : redraw the line on each new day
 - RD_HOURLY : redraw the line on each new hour
 - RD_NEWBAR : redraw the line on each new bar
 - RD_NEWTICK : redraw the line on each new tick

Note: A trend line is projected to the right *indefinitely*. It is not necessary to redraw the trend line unless the drawing coordinates change.

Four coordinates are required to draw a line in two dimensional space. For example, on a typical graph, using the [Cartesian](#) coordinate system, the coordinates are usually defined as (x1, y1) and (x2,y2).

The primary coordinates of the trend line are defined as the **StartBar**, the **StartPrice**, the **EndBar** and the **EndPrice**.

- **StartBar:** set the value of the **StartBar** parameter to the bar (or candle) where the trend line should begin.
 - **NOTE:** *On a MetaTrader price chart, the currently forming candle is defined as candle 0 and the candle numbers increase to the left. Therefore, the StartBar is a larger number than the EndBar!*
- **StartPrice:** set the value of the **StartPrice** parameter to a [price value constant](#). The price is the vertical location where the line begins. The **StartPrice** is the *price value of the StartBar*.
- **StartOffset:** set the value of the **StartOffset** parameter to a positive or negative integer value (for example:10, 20, ...). The **StartOffset** value moves the vertical start location up or down relative to the **StartPrice**.
- **EndBar:** set the value of the **EndBar** parameter to the bar (or candle) where the trend line should end.
 - **NOTE:** A trend line is projected to the right *indefinitely* regardless of where the EndBar is located. The EndBar simply defines the angle of the line, not its actual end point.
- **EndPrice:** set the value of the **EndPrice** parameter to a [price value constant](#). The price is the vertical location where the trend line ends. The **EndPrice** is the *price value of the EndPrice*.
 - **NOTE:** If the **EndPrice** is set as the Close, High, or Low of bar number 0, or to Ask or Bid, then the **EndPrice** may change value on each tick.
- **EndOffset:** set the value of the **EndOffset** parameter to a positive or negative integer value (for example:10, 20, ...). The **EndOffset** value moves the vertical end location up or down relative to the **EndPrice**.
- **LineColor:** optionally set the value of the **LineColor** parameter to a Color. The default is Black.
- **DrawLabel:** optionally set the value of the **DrawLabel** parameter to a true. (The default is false.) This will draw a label with the name of the trend line on the price chart. The label can be removed programatically using [fnDeleteAllLabels](#), or manually from from the [MetaTrader tool bar](#).
- **DrawMarks:** optionally set the value of the **DrawMarks** parameter to a true. (The default is false.) This will draw a vertical lines through the start and end bars. The vertical lines can be removed programatically using [fnDeleteAllVerticalLines](#), or manually from from the [MetaTrader tool bar](#).



FX PowerGrid Plug-in

Requires VTS-Connect minimum version **4.0.0.43**

The **FX PowerGrid Plug-in** allows you to easily build a trading grid that opens multiple positions as the market moves. A trading grid is ideal in ranging market conditions.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor Builder* is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.

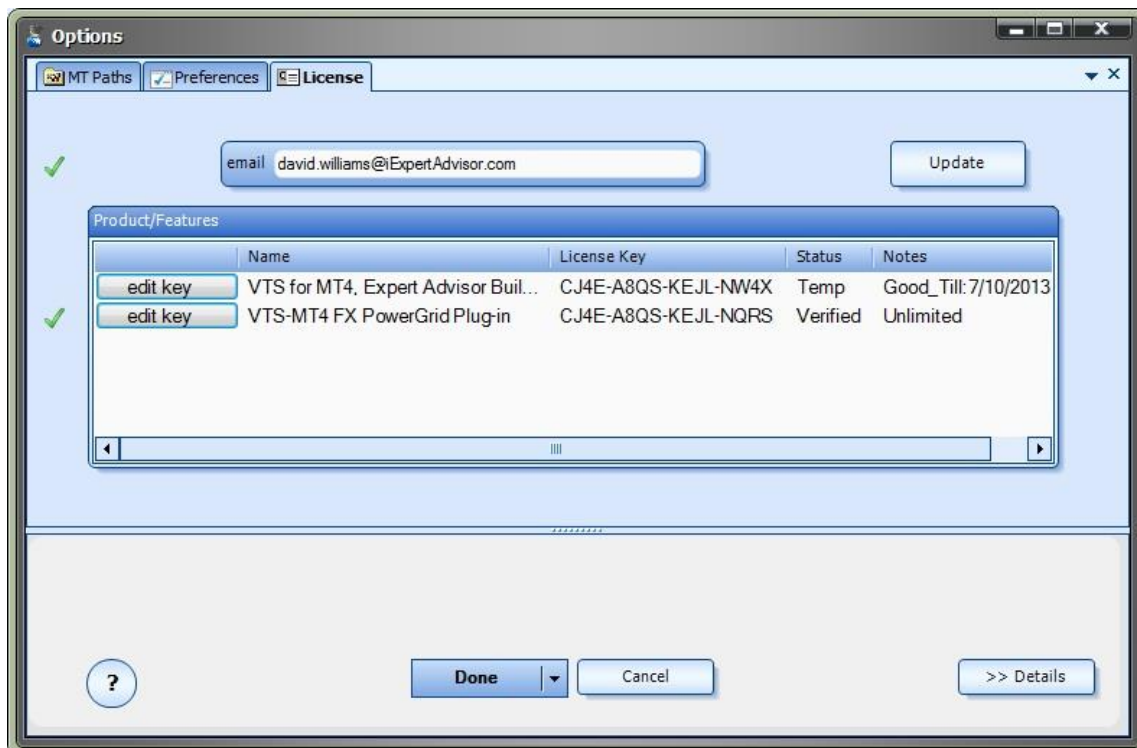


Enable the *FX PowerGrid* Plug-in

You must enter your License key to enable the ***FX PowerGrid Plug-in***. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

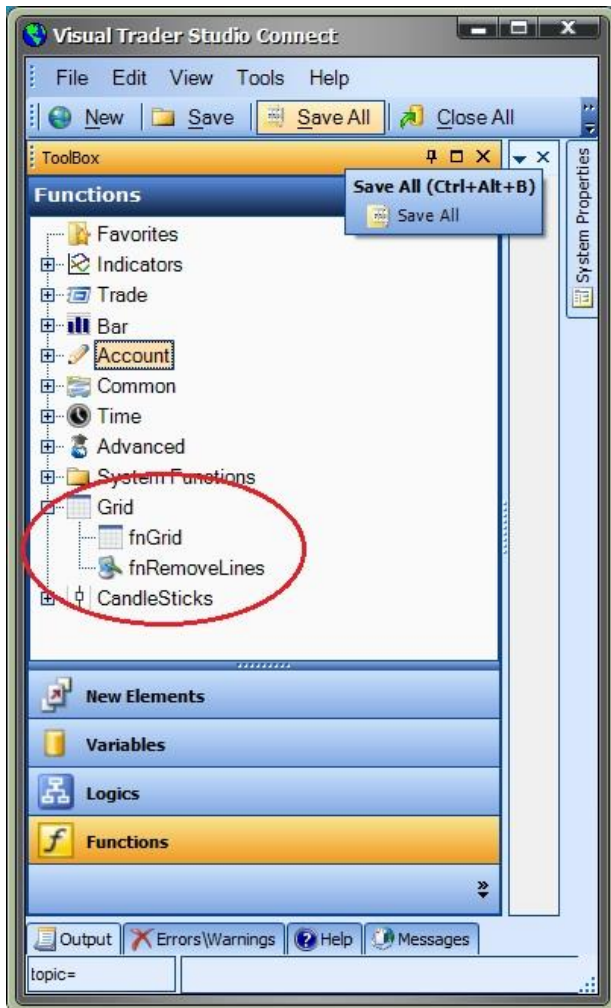
- The **email** address is the email address used to purchase [VTS](#).
- The **License Key** is the key listed in the Members Area.
- The **Update** button is used to verify the email address and license key.
- The **edit key** button is used edit the key value.



FX PowerGrid Functions in the Toolbox

Once enabled, the PowerGrid functions are available in the [Toolbox](#) Function tab under the Grid menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.



FX PowerGrid Functions

PowerGrid Functions

The PowerGrid function library include these functions:

- [fnGrid](#)
 - Used to build a trading grid Expert Advisor.
- [fnRemoveLines](#)
 - Used to remove vertical and horizontal lines from a price chart.

fnGrid

The **FX PowerGrid** function *fnGrid* is used to build a fully functional trading grid.

After the *fnGrid* function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [fnGridFunction Configuration](#) window displays three main tabs:

- **FX Power Grid** The main tab for configuring the grid.
- **Notes** The [Note](#) tab is used to add useful comments to the [Element](#).
- **Message** The [Message](#) tab is used to send messages from the [Element](#).

The **FX Power Grid** main tab displays 3 sub-tabs:

- [Grid Main Settings](#) Used to set the main settings of the trading grid
- [Grid Up Levels](#) Used to configure and add [Up Levels](#) to the trading grid
- [Grid Down Levels](#) Used to configure and add [Down Levels](#) to the trading grid

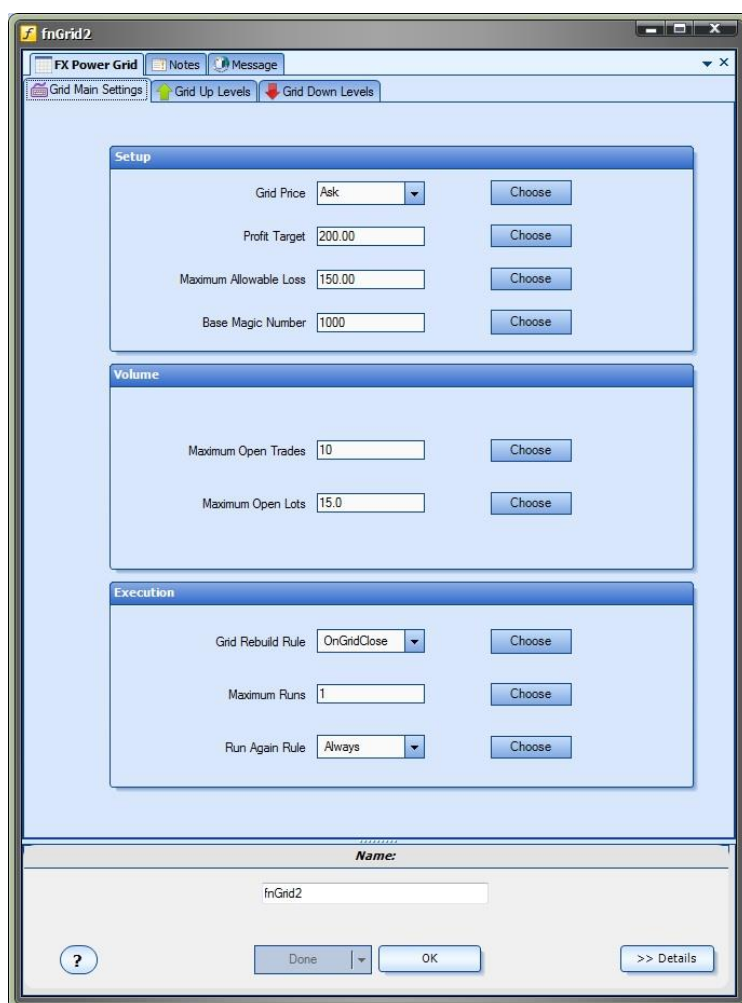
Grid Main Settings

Grid Main Settings

The **Grid Main Settings** tab is used to set global settings for the trading grid. The following table describes each setting.

Parameter Name	Data type	Description
Grid Price (<i>Grid Start Price</i>)	double	This is the price used for determining price action throughout the grid. For example, if the Grid Price is set to Bid , trades will be opened based on the value of the Bid price with respect to the Grid levels. Any price value can be used. Most trading grids use Ask, Bid or (Ask+Bid)/2. These values are all available from the pull-down menu as well as the Choose button. This value also defines the Grid Start Price . The start price is set the moment the Grid is first build and is used as an anchor price for the life of the grid.
Profit Target	double	This is the profit value in dollars (or local currency) that when reached will cause the grid to be closed for a profit. The profit is calculated using only the trades opened by the grid, not all trades open on the account.
Maximum Allowable Loss	double	This is the loss value in dollars (or local currency) that when reached will cause the grid to be closed for a loss. The loss is calculated using only the trades opened by the grid, not all trades open on the account.
Base Magic Number	integer	A Magic Number is used to identify an open trade. The Base Magic Number is used to define magic numbers for each level of the grid. For example, if the Base Magic Number is 1000: <ul style="list-style-type: none"> • A trade opened in Up Level 1 will have a magic number of 1001. • A trade opened in Up Level 2 will have a magic number of 1002. • Etc. • A trade opened in Down Level 1 will have a magic number of 999. • A trade opened in Down Level 2 will have a magic number of 999. • Etc. NOTE: The trader must take care to not allow the magic numbers to collide with other running Expert Advisors that use magic numbers.
Maximum Open Trades	integer	The maximum number of trades that the grid will open. Any attempt by the Grid to open a trade after this number has been reached will be denied.
Maximum Open Lots	double	The maximum total number of lots that the grid will open. Any attempt by the Grid to open additional lots after this number has been reached will be denied.
Grid Rebuild Rule	integer	This rule defines when the Grid will be rebuilt. The allowable values are: <ul style="list-style-type: none"> • Never: The Grid is never rebuilt. • OnGridClose: The Grid is rebuilt when the Grid has been close for a profit or loss. <i>This is the behavior of most grids.</i> • OnDay: The Grid is rebuilt at the start of each day.

		<ul style="list-style-type: none"> ● OnHour: The Grid is rebuilt at the start of each hour. ● OnNewBar: The Grid is rebuilt at the start of each new bar. ● OnTick: The Grid is rebuilt on each new tick. Note: This setting only makes sense in very specific configurations, for example when the grid levels are defined as prices (such as a Moving Average) that may change with each tick.
Maximum Runs	integer	<p>Defines how many times the Grid may run. A run is complete when the Grid has closed for either a Profit or a Loss.</p> <p>Note: A value of 0 allows unlimited runs.</p>
Run Again Rule	integer	<p>Defines when the Grid may run again. The allowable values are:</p> <ul style="list-style-type: none"> ● Always: Always runs again, until Maximum Runs is reached. ● OnceOnly: Runs one time only. ● OnLoss: Runs again only after a loss. ● OnWin: Runs again only after a win.



Grid Up Levels

The **Grid Up Levels** tab is used to define an unlimited number of levels for a trading grid. The following table describes each setting.

Parameter Name	Data type	Description
Grid Price Level	double	<p>The price level which represents the start of this level. For example, if the grid is started when the Bid price is 1.3450, a reasonable level for the first level would be 1.3470. If the Bid price is ever at or above 1.3470, a trade will be opened as defined by the settings for the level.</p> <p>This value may be an offset or an absolute price level.</p> <ul style="list-style-type: none"> An example of an offset is 25. This will define that the level begins at the Grid Start Price plus 25 pips. An example of an absolute price is the 12-period moving Average, or any technical indicator that yields a price value. <p>Note: offsets are cumulative: If level 1 has an offset of 20 and level 2 has an offset of 30, then the start of level 2 is at: Grid Start Price + 20 + 30</p>
Trade Type	integer	<p>Defines the type of trade opened when the Grid Price is in this level. The choices from the pull-down menu are:</p> <p>OP_BUY OP_SELL OP_BUYLIMIT OP_SELLLIMIT OP_BUYSTOP OP_SELLSTOP</p> <p>Note: Pending orders are not opened until the Grid Price enters the level.</p>
Lot Size	double	<p>The Lot size of any trade opened in this level. The value may be multiple or a value.</p> <ul style="list-style-type: none"> An example of a multiple is 1.5. This will define the lot size as 1.5 multiplied by the level number and then multiplied by the global <i>Lots</i> extern variable. Level 1: (1.5 x 1 x Lots) Level 2: (1.5 x 2 x Lots) An example of a value is any valid lot number, such as 1.0, or any variable, such as the extern variable <i>Lots</i>. <p>Note: The global <i>Lots</i> extern variable is automatically created by VTS.</p>
Stoploss	double	<p>The stoploss value of any trade opened in this level. The value may be offset or a absolute value.</p> <ul style="list-style-type: none"> The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].
Takeprofit	double	<p>The takeprofit value of any trade opened in this level. The value may be offset or a absolute value.</p> <ul style="list-style-type: none"> The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].

Add Level

The **Add Level** button on the bottom right corner of the window is used to add levels to the grid. In theory, an unlimited number of levels may be added, however the number will be bounded by the memory and processor of your PC.

Remove Level

The **Add Level** button offers a pull-down to **Remove Last Level**. Note: only the last level added may be removed and a minimum of one level must always exist.

The screenshot displays the 'fnGrid2' application window. At the top, there are tabs for 'FX Power Grid', 'Notes', and 'Message'. Below these are buttons for 'Grid Main Settings', 'Grid Up Levels', and 'Grid Down Levels'. The main area shows three levels: 'Level 1', 'Level 2', and 'Level 3'. The 'Level 1 Price Level' section includes a 'value type' dropdown, radio buttons for 'Offset value' (selected) and 'Absolute value (price)', a text input field containing '500.0', and a 'Choose' button. The 'Level 1 Trade Parameters' section contains four sub-sections: 'Trade Type' with a dropdown set to 'OP_BUY' and a 'Choose' button; 'Lot Size' with a 'value type' dropdown, radio buttons for 'Multiple' and 'Value' (selected), a 'Lots' text input field, and a 'Choose' button; 'Stoploss' with a 'value type' dropdown, radio buttons for 'Offset value' (selected) and 'Absolute value (price)', a 'StopLoss' text input field, and a 'Choose' button; and 'Takeprofit' with a 'value type' dropdown, radio buttons for 'Offset value' (selected) and 'Absolute value (price)', a 'TakeProfit' text input field, and a 'Choose' button. At the bottom right of the main area is an 'Add Level' button with a pull-down arrow. Below the main area is a 'Name:' label and a text input field containing 'fnGrid2'. At the very bottom are a help icon (?), a 'Done' button with a pull-down arrow, an 'OK' button, and a '>> Details' button.

Grid Down Levels

The **Grid Down Levels** tab is used to define an unlimited number of levels for a trading grid. The following table describes each setting.

Parameter Name	Data type	Description
Grid Price Level	double	<p>The price level which represents the start of this level. For example, if the grid is started when the Bid price is 1.3450, a reasonable level for the first level would be 1.3430. If the Bid price is ever at or below 1.3430, a trade will be opened as defined by the settings for the level.</p> <p>This value may be an offset or an absolute price level.</p> <ul style="list-style-type: none"> An example of an offset is 25. This will define that the level begins at the Grid Start Price minus 25 pips. An example of an absolute price is the 12-period moving Average, or any technical indicator that yields a price value. <p>Note: offsets are cumulative: If level 1 has an offset of 20 and level 2 has an offset of 30, then the start of level 2 is at: Grid Start Price - 20 - 30</p>
Trade Type	integer	<p>Defines the type of trade opened when the Grid Price is in this level. The choices from the pull-down menu are:</p> <p>OP_BUY OP_SELL OP_BUYLIMIT OP_SELLLIMIT OP_BUYSTOP OP_SELLSTOP</p> <p>Note: Pending orders are not opened until the Grid Price enters the level.</p>
Lot Size	double	<p>The Lot size of any trade opened in this level. The value may be multiple or a value.</p> <ul style="list-style-type: none"> An example of a multiple is 1.5. This will define the lot size as 1.5 multiplied by the level number and then multiplied by the global <i>Lots</i> extern variable. Level 1: (1.5 x 1 x Lots) Level 2: (1.5 x 2 x Lots) An example of a value is any valid lot number, such as 1.0, or any variable, such as the extern variable <i>Lots</i>. <p>Note: The global <i>Lots</i> extern variable is automatically created by VTS.</p>
Stoploss	double	<p>The stoploss value of any trade opened in this level. The value may be offset or a absolute value.</p> <ul style="list-style-type: none"> The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].
Takeprofit	double	<p>The takeprofit value of any trade opened in this level. The value may be offset or a absolute value.</p> <ul style="list-style-type: none"> The Absolute value option is useful for using technical indicator values or price values such as Low[1] or High[1].

Add Level

The **Add Level** button on the bottom right corner of the window is used to add levels to the grid. In theory, an unlimited number of levels may be added, however the number will be bounded by the memory and processor of your PC.

Remove Level

The **Add Level** button offers a pull-down to **Remove Last Level**. Note: only the last level added may be removed and a minimum of one level must always exist.

The screenshot displays the 'fnGrid2' application window. At the top, there are tabs for 'FX Power Grid', 'Notes', and 'Message'. Below these are buttons for 'Grid Main Settings', 'Grid Up Levels', and 'Grid Down Levels'. The main area shows three levels: 'Level 1', 'Level 2', and 'Level 3', each with a red downward arrow. The 'Level 1' configuration is expanded, showing the following settings:

- Level 1 Price Level:** A 'value type' dropdown is set to 'Offset value'. The 'Grid Price Level' is '50.0'. There are radio buttons for 'Offset value' (selected) and 'Absolute value (price)'. A 'Choose' button is present.
- Level 1 Trade Parameters:**
 - Trade Type:** A dropdown menu is set to 'OP_SELL'. A 'Choose' button is present.
 - Lot Size:** A 'value type' dropdown is set to 'Value'. There are radio buttons for 'Multiple' and 'Value' (selected). A 'Lots' input field and a 'Choose' button are present.
 - Stoploss:** A 'value type' dropdown is set to 'Offset value'. The 'StopLoss' input field contains 'StopLoss'. There are radio buttons for 'Offset value' (selected) and 'Absolute value (price)'. A 'Choose' button is present.
 - Takeprofit:** A 'value type' dropdown is set to 'Offset value'. The 'TakeProfit' input field contains 'TakeProfit'. There are radio buttons for 'Offset value' (selected) and 'Absolute value (price)'. A 'Choose' button is present.

At the bottom right of the configuration area is an 'Add Level' button with a red downward arrow. Below this is a 'Name:' label and an input field containing 'fnGrid2'. At the very bottom, there is a '?' icon, a 'Done' button with a dropdown arrow, an 'OK' button, and a '>> Details' button.

fnRemoveLines

The **FX PowerGrid** function *fnRemoveLines* is used to remove all vertical and horizontal lines from a price chart.

The *fnGrid* function draws lines for the grid start price, start time and each grid level. The *fnRemoveLines* function can be used to programmatically remove all of these lines.

One convenient method is to create a [deinit](#) drawing function and add *fnRemoveLines* to the drawing. This will cause all of the lines to be removed each time the Expert Advisor is removed from the chart.

- The *fnRemoveLines* does not required any parameters.
- The *fnRemoveLines* is a [Drawing function](#). Its drawing can be opened and/or modified by double-clicking the caption of the function.

Using FX PowerGrid

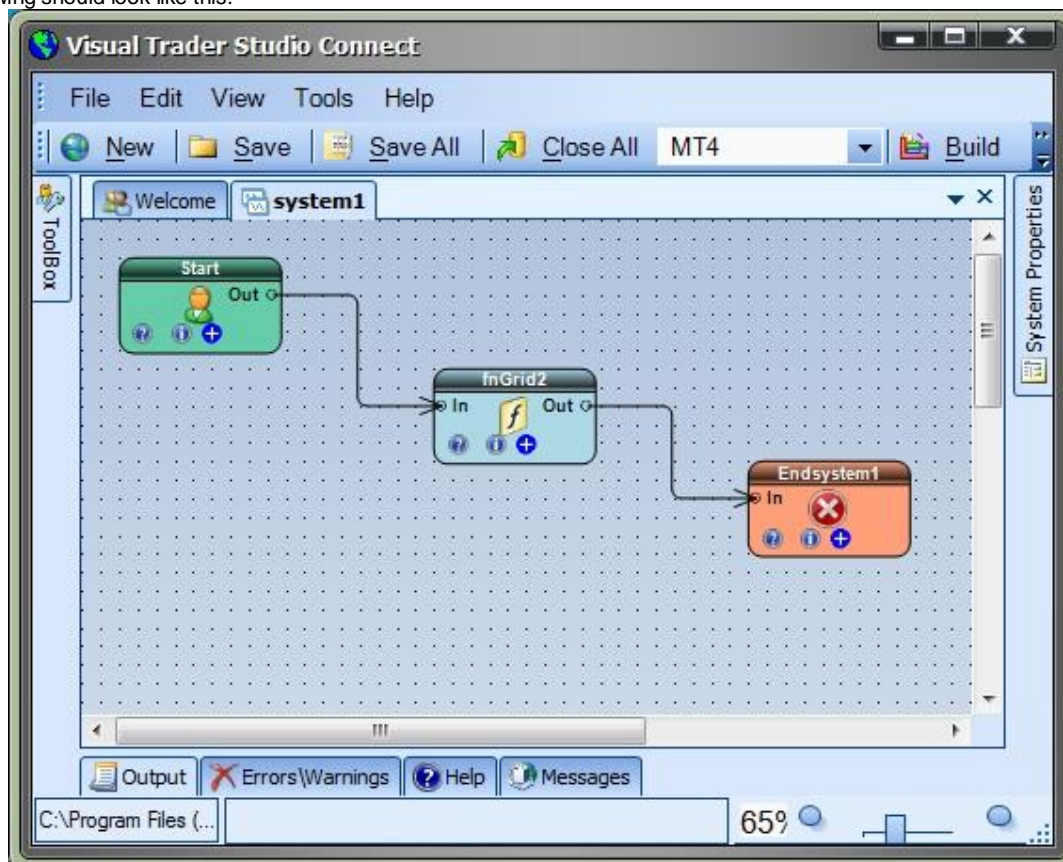
Using FX PowerGrid

FX PowerGrid is very easy to use.

Follow these 5 steps to build your first [trading grid](#).

1. Create a new EA Builder system
2. Uncheck all Drawings so that only a main-system drawing is created
3. Drag an [fnGrid](#) Function Element from the [ToolBox](#) and connect it between the [Start](#) and the [EndElement](#).

The drawing should look like this:

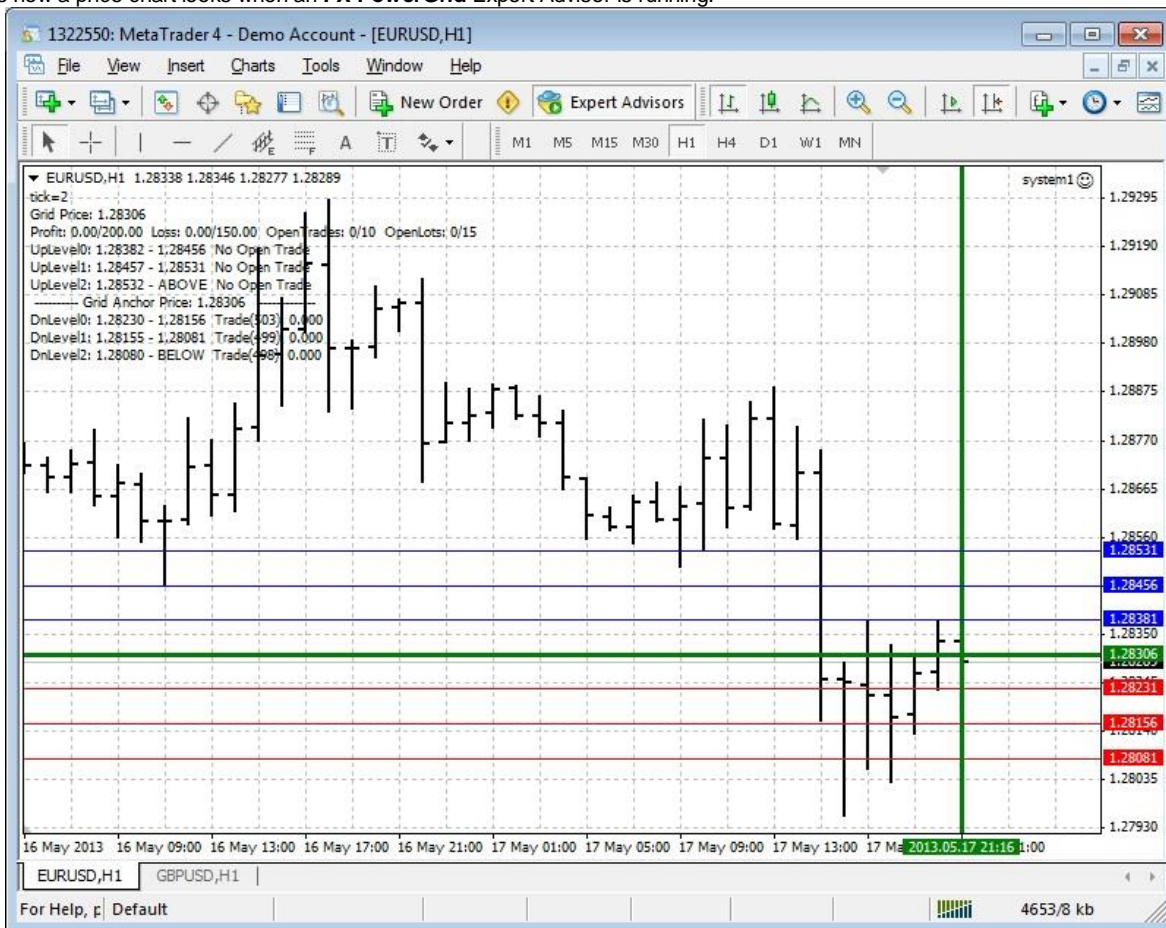


4. Now [configure](#) your grid by clicking the (+) button in the bottom right corner of the fnGrid Element.
5. Click the [Build](#) button to create your Expert Advisor [Trading Grid](#).

Notes

- If any indicators are used within the grid, they should be connected before the grid Element.
- The grid Element may be attached to [Logic](#) Elements, however care should be taken to ensure the flow of Logic allows the Grid to be executed when desired.
- More than one grid can be added to a drawing. The limitation will be with the memory and/or processor power of the running PC. Also, be careful to define [base Magic Numbers](#) that do not interfere with each other.

This how a price chart looks when an **FX PowerGrid** Expert Advisor is running.



- The Grid Start Time is the thick green horizontal line.
- The Grid Start Price (anchor price) is the thick vertical line.
- The start of each Up Level begins with a blue line.
- The start of each Down Level begins with a red line.
- The top left corner of the chart shows:
 - The current Grid Price.
 - The current Total Profit, Total Loss, Open Trades and Open Lots
 - The range of each level, and if a trade is open in that level the profit of the trade.

Also, the comment field of each trade opened by the **FX PowerGrid** contains the EA name and the magic number of the trade.

Script & Multi-Platform Plug-in

Requires VTS-Connect minimum version **4.0.0.45**

The **Script & Multi-Platform Plug-in** allows you to build MetaTrader [Scripts](#) and build an Expert Advisor in multiple locations.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.

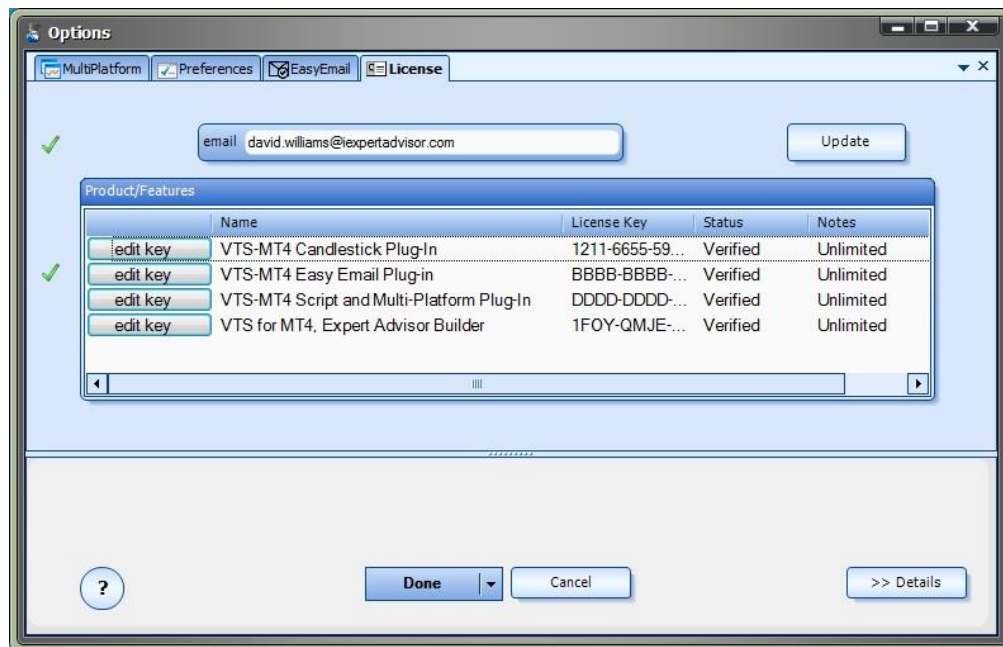


Enable the Script & Multi-Platform Plug-in

You must enter your License key to enable the **Script & Multi-Platform Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

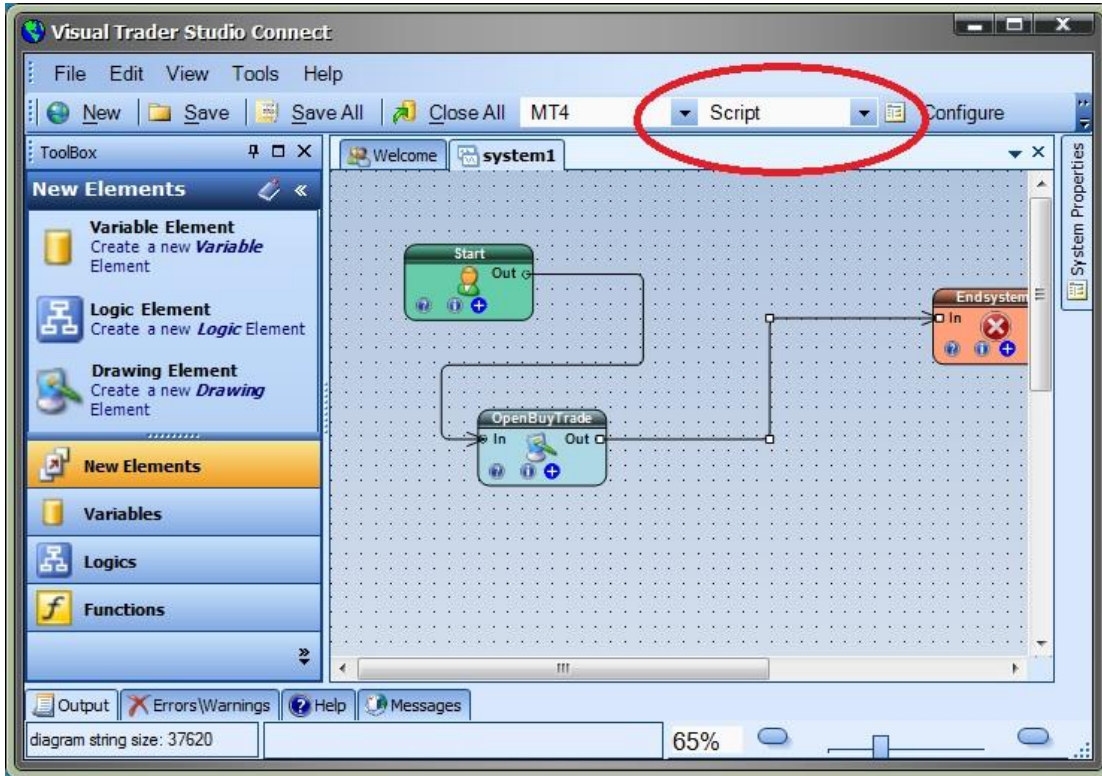
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

- The **email** address is the email address used to purchase [VTS](#).
- The **License Key** is the key listed in the Members Area.
- The **Update** button is used to verify the email address and license key.
- The **edit key** button is used edit the key value.



Building a MetaTrader Script

To build a MetaTrader [script](#), change the build target from *Expert Advisor* to *Script* on the VTS [Main Menu](#) and click the Build button.

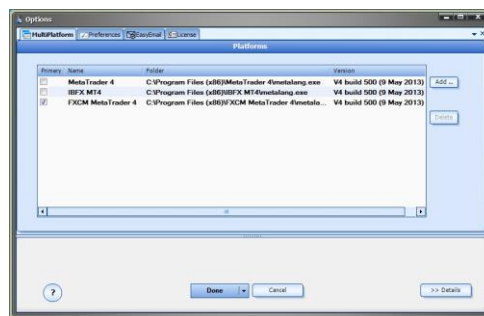


Using the Multi-Platform Feature

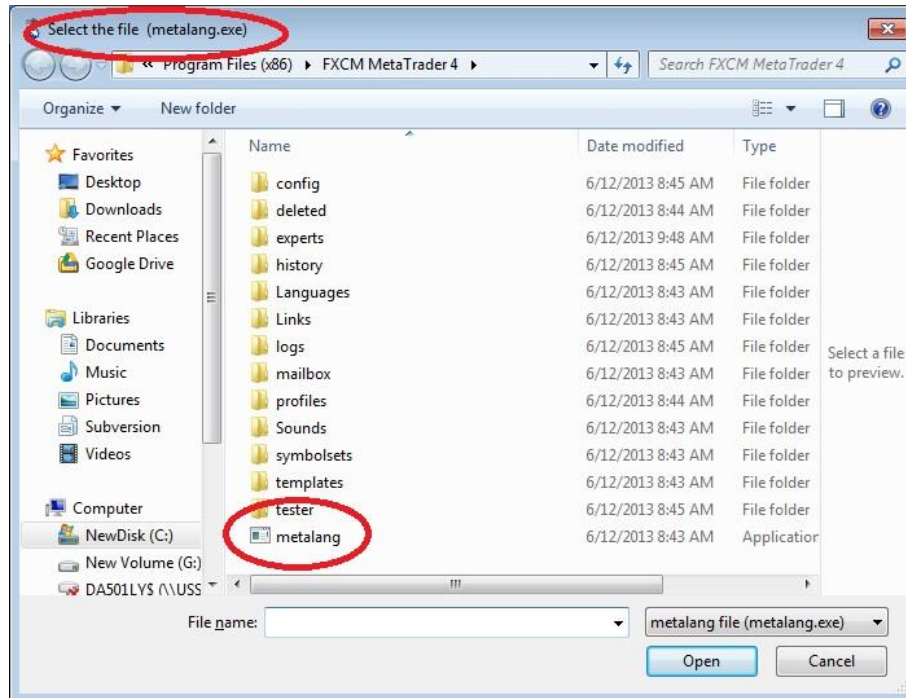
The *Multi-Platform* feature allows VTS to build an Expert Advisor or Script in multiple MetaTrader platforms.

A MetaTrader platform is added to VTS using the Multi-Platform dialog screen.

The *Multi-Platform* dialog screen is found at *Tools->Options-> MultiPlatform*



The *Add* button is used to add MetaTrader platform folder. Navigate to and select the file *metalang.exe* to add a MetaTrader platform.



The *Delete* button is used to remove a MetaTrader platform from the list. Note, a row of the *MultiPlatform* list must be selected to enable the *Delete* button.

The following columns are found in the *MultiPlatform* list:

- Primary* This checkbox defines the MetaTrader platform as the primary platform
- Name The name extracted from the MetaTrader platform path
- Folder The folder to the *metalang.exe* file for the MetaTrader platform
- Version The version of the MetaTrader platform

* Only one MetaTrader platform may be designated as *primary*. The primary MetaTrader platform holds the only copy of the MQL source. If any other copies of the MQL source exist in a non-primary MetaTrader platform's folder they will be deleted. Only a single copy of the MQL code is saved to avoid multiple copies of the same MQL file that may contain different code.

Note: Ensure that any Custom Indicators used in the VTS system exist in all MetaTrader platforms. If a Custom Indicator is not found, a *run-time* error will occur; no error will be generated during *build-time*.

Profit Exits Plug-in

Requires VTS-Connect minimum version 4.0.0.46

The *Profit Exits Plug-in* allows you to easily add advanced trade exits to your Expert Advisor.

What is a Plug-in?

VTS stands for *Visual Traders Studio*.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A *VTS Plug-in* allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the Profit Exits Plug-in

You must enter your License key to enable the *Profit Exit Plug-in*. Your license key for all of your VTS products can be found in the [Members Area](#).

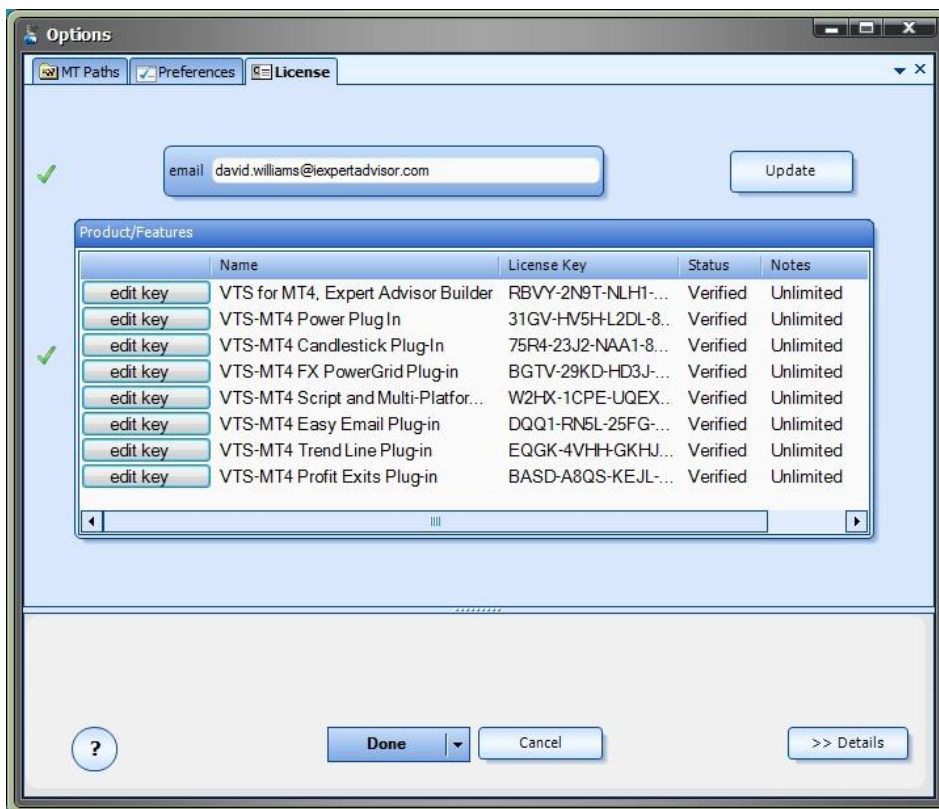
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The email address is the email address used to purchase [VTS](#).

The License Key is the key listed in the Members Area.

The Update button is used to verify the email address and license key.

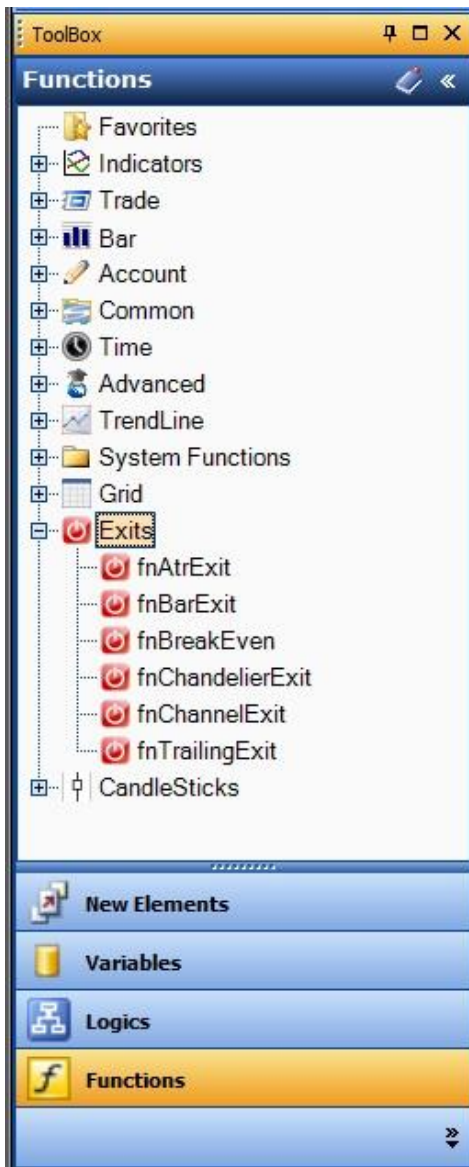
The edit key button is used edit the key value.



Profit Exit Functions in the Toolbox

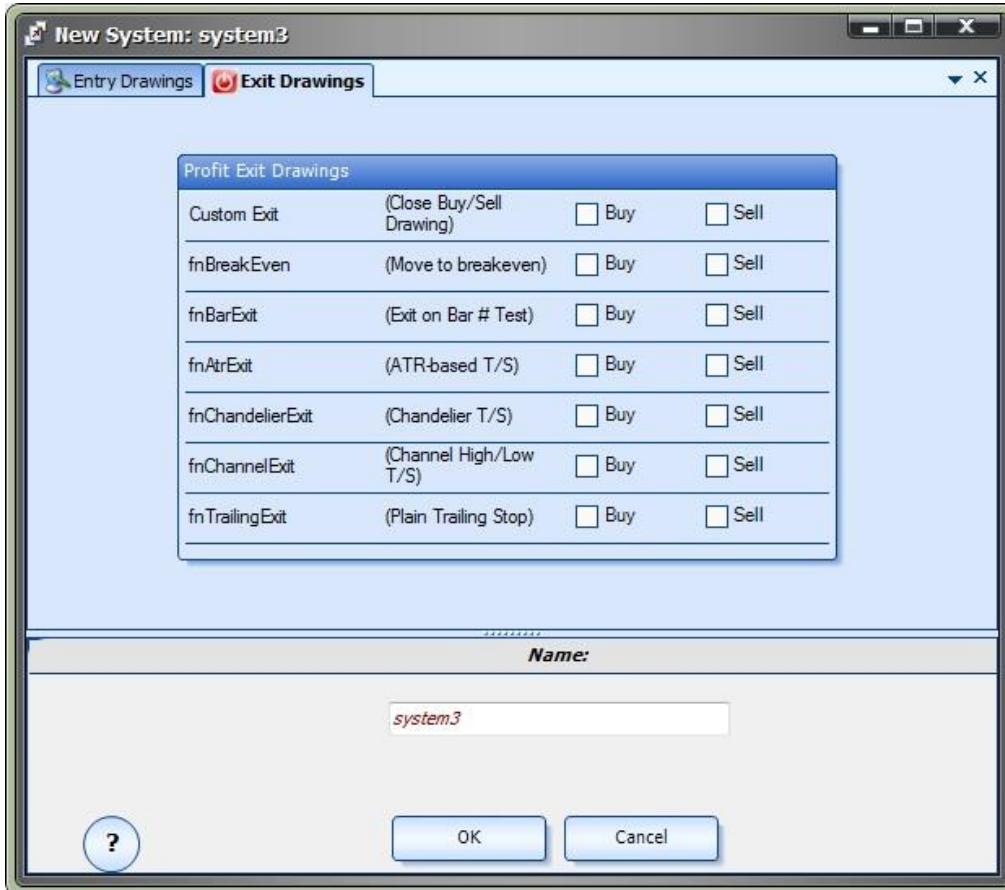
Once enabled, the Profit Exit functions are available in the [Toolbox](#) Function tab under the Grid menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.



Profit Exit Functions in the New System Window

Once enabled, the Profit Exit functions are available from the *New System* Window that is shown when creating a new system.



In addition to the *Entry Drawings* tab, a new tab named *Exit Drawings* is available.

Exit Name	Description	Buy	Sell
Custom Exit	A custom exit will create a drawing with a non-configured Logic element. This is the same drawing that is created when the Profit Exit plug-in is not installed. Note: Selecting this Exit will disable all other Exits choices.	Check to create <i>CloseBuyTrade</i> drawing.	Check to create <i>CloseSellTrade</i> drawing.
fnBreakEven	Adds the <i>fnBreakEven</i> function to the drawing.	Check to add the <i>fnBreakEven</i> function to the <i>CloseBuyTrade</i> drawing.	Check to add the <i>fnBreakEven</i> function to the <i>CloseSellTrade</i> drawing.
fnBarExit	Adds the <i>fnBarExit</i> function to the drawing.	Check to add the <i>fnBarExit</i> function to the <i>CloseBuyTrade</i> drawing.	Check to add the <i>fnBarExit</i> function to the <i>CloseSellTrade</i> drawing.
fnAtrExit	Adds the <i>fnAtrExit</i> function to the drawing.	Check to add the <i>fnAtrExit</i> function to the <i>CloseBuyTrade</i> drawing.	Check to add the <i>fnAtrExit</i> function to the <i>CloseSellTrade</i> drawing.
fnChandelierExit	Adds the <i>fnChandelierExit</i> function to the drawing.	Check to add the <i>fnChandelierExit</i> function to the <i>CloseBuyTrade</i> drawing.	Check to add the <i>fnChandelierExit</i> function to the <i>CloseSellTrade</i> drawing.
fnChannelExit	Adds the <i>fnChannelExit</i> function to the drawing.	Check to add the <i>fnChannelExit</i> function to the	Check to add the <i>fnChannelExit</i> function to the

		<i>CloseBuyTrade</i> drawing.	<i>CloseBuyTrade</i> drawing.
fnTrailingExit	Adds the <i>fnTrailingExit</i> function to the drawing.	Check to add the <i>fnTrailingExit</i> function to the <i>CloseBuyTrade</i> drawing.	Check to add the <i>fnTrailingExit</i> function to the <i>CloseBuyTrade</i> drawing.

Note: The *Profit Exit* functions are available from the *New System* menu as a convenience.

All of the *Profit Exit* functions are available from the [Toolbox](#) and can be manually added to any [drawing](#) at any time.

Profit Exit Functions

The Profit Exit function library include these functions:

[fnAtrExit](#)

Executes a trailing stop using the Average True Range (ATR) indicator.

[fnBarExit](#)

Closes an open order after a number of bars.

[fnBreakEven](#)

Moves an order's stoploss to the order's open price (break even).

[fnChandelierExit](#)

Exits on a high or low since the trade was opened.

[fnChannelExit](#)

Exits on a high or low seen in the last N bars (channel).

[fnTrailingExit](#)

Executes a trailing stop.

fnAtrExit

The Profit Exit function *fnAtrExit* executes a trailing stop using the Average True Range (ATR) indicator.

The open order is trailed using the value of the ATR calculated at the close of the last bar.

For example, if the value of the ATR is 15, the order is trailed by 15 points (or [PIPs](#)).

The period parameter is used to define the period used to calculate the ATR.

The count parameter is used as a multiple of the ATR value.

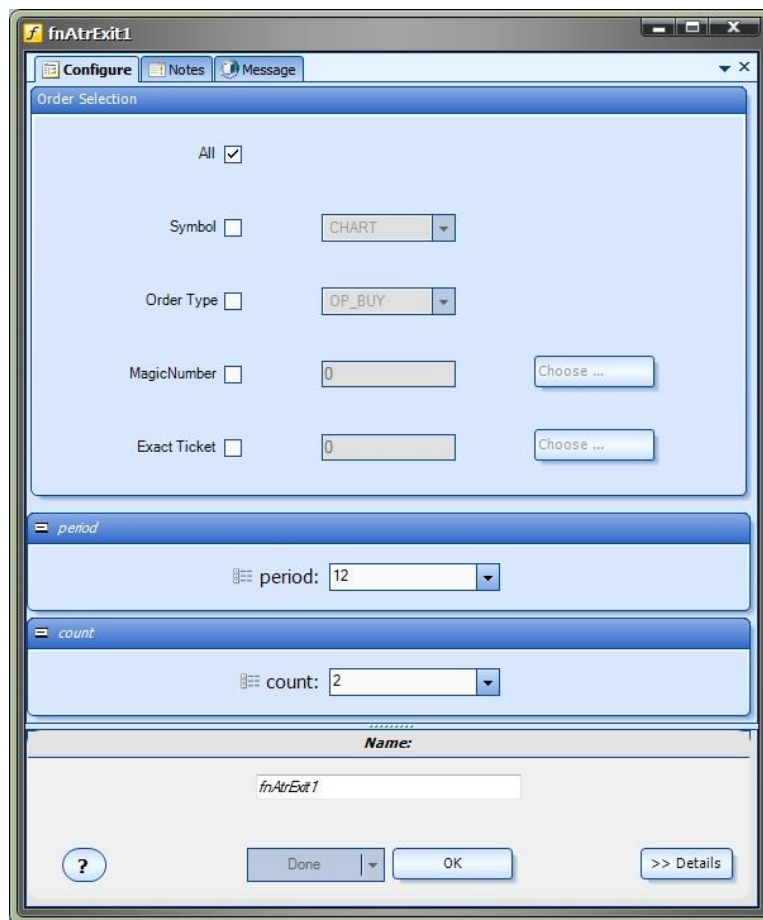
For example, if the value of the ATR is 15 and the value of the count parameter is 2, then the order is trailed by 30 points (or [PIPs](#))

After the *fnAtrExit* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

The following table provides information about each [parameter](#) of the *fnAtrExit* function.

Order Selection	<p>Determines what orders of the Account are monitored.</p> <p>All: all open orders on the account.</p> <p>Symbol: only open orders matching the selected symbol.</p> <p>Order Type: only open orders of the selected order type.</p> <p>MagicNumber: only open orders with a matching magicnumber.</p> <p>Exact Ticket: only open orders with the matching order ticket.</p> <p>Note: Selecting <i>Exact Ticket</i> disables all other selection criteria.</p>
period	The period used to calculate the ATR value.
count	A multiplier applied to the ATR value.



fnBarExit

The Profit Exit function *fnBarExit* closes an open order after a number of bars.

The *fnBarExit* counts the number of bars (or candles) that have occurred since the order was opened.

When the trade has been open for the defined number of bars, it is closed.

Note: The *fnBarExit* is normally used for testing only!

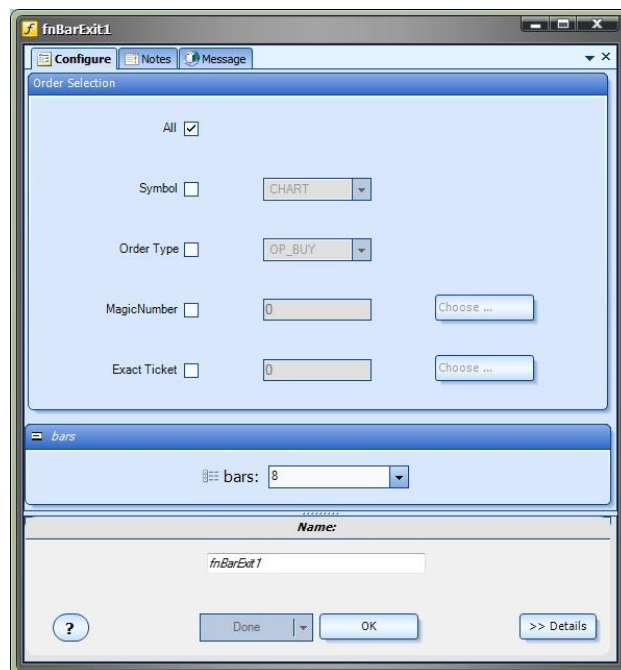
Specifically, *fnBarExit* is used to test the quality of a system's entry logic.

After the *fnBarExit* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

The following table provides information about each [parameter](#) of the *fnBarExit* function.

Order Selection	Determines what orders of the Account are monitored. All: all open orders on the account. Symbol: only open orders matching the selected symbol. Order Type: only open orders of the selected order type . MagicNumber: only open orders with a matching magicnumber . Exact Ticket: only open orders with the matching order ticket . Note: Selecting <i>Exact Ticket</i> disables all other selection criteria.
bars	The number of bars allowed to elapse before closing the trade.



fnBreakEven

The Profit Exit function *fnBreakEven* moves an open order's stoploss to the order's open price (break even).

When the profit of an open order reaches a defined value, the *fnBreakEven* function moves the stoploss to order's open price value (break even).

The points parameter is used to define the number of points (or [PIPs](#)) of profit before moving the stoploss to break-even.

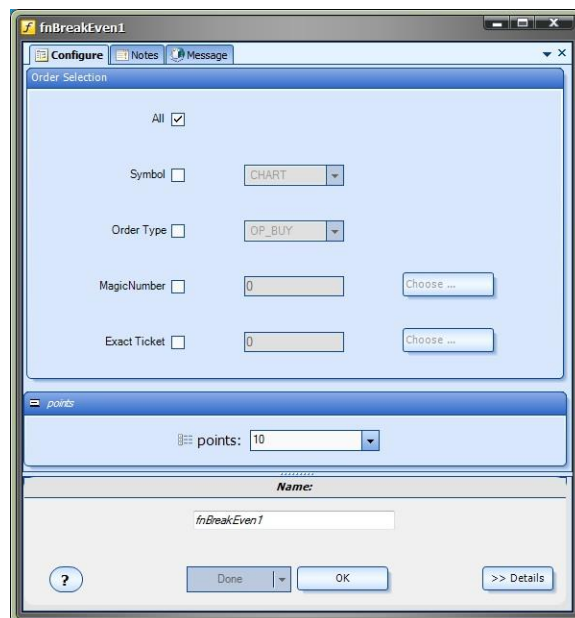
The points parameter is an integer value. For example, if points is 10, the order is moved to break-even when the trade has 10 points of profit.

After the *fnAtrExit* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

The following table provides information about each [parameter](#) of the *fnAtrExit* function.

Order Selection	<p>Determines what orders of the Account are monitored.</p> <p>All: all open orders on the account.</p> <p>Symbol: only open orders matching the selected symbol.</p> <p>Order Type: only open orders of the selected order type.</p> <p>MagicNumber: only open orders with a matching magicnumber.</p> <p>Exact Ticket: only open orders with the matching order ticket.</p> <p>Note: Selecting <i>Exact Ticket</i> disables all other selection criteria.</p>
points	Number of points, as an integer, of profit before the order is moved to break-even.



fnChandelierExit

The Profit Exit function *fnChandelierExit* exits on a high or low since the trade was opened.

The *fnChandelierExit* determines its stoploss based on the highest value the market has reached since the trade has been open.

The stoploss is placed somewhat beneath the highest high – in terms of points (or [PIPs](#)).

The term chandelier is used because the stoploss hangs like a chandelier from the market high (or low).

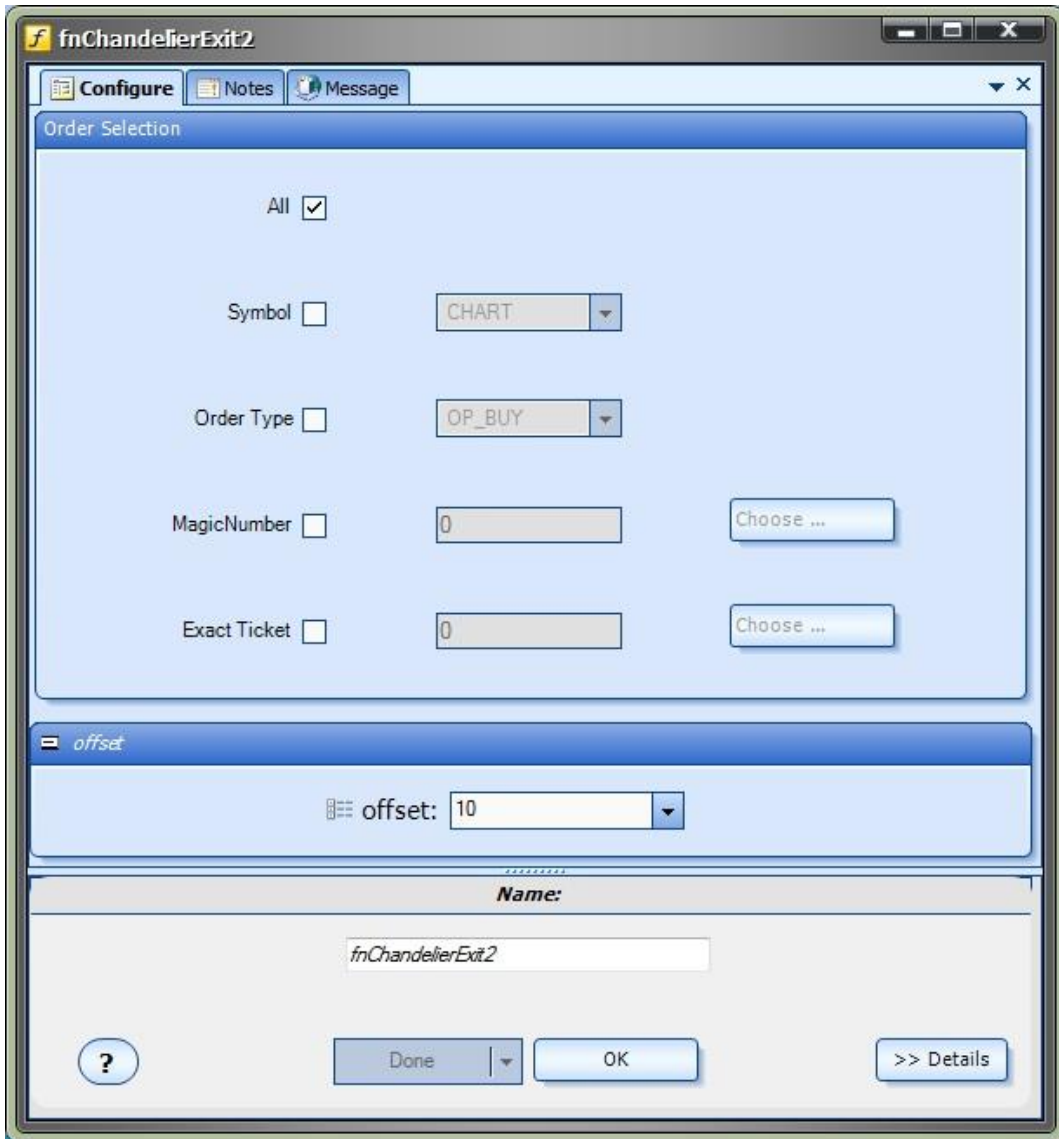
The offset parameter is used to “hang” from the highest high seen during the trade’s lifetime (or the lowest low).

After the *fnChandelierExit* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

The following table provides information about each [parameter](#) of the *fnChandelierExit* function.

Order Selection	Determines what orders of the Account are monitored. All: all open orders on the account. Symbol: only open orders matching the selected symbol. Order Type: only open orders of the selected order type . MagicNumber: only open orders with a matching magicnumber . Exact Ticket: only open orders with the matching order ticket . Note: Selecting <i>Exact Ticket</i> disables all other selection criteria.
offset	The number of points away from the highest high or lowest low to place the trailing stop.



fnChannelExit

The **Profit Exit** function *fnChannelExit* exits on a high or low in the last N bars.

The *fnChannelExit* function is based on the lowest (or highest) price seen in the last N periods, or in the channel period. For instance, if a system based on 1-hour charts uses a channel period of 24, the lowest (or highest) price seen in the last 24 periods (or bars, or hours) is set as the stoploss.

After the *fnChannelExit* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

The following table provides information about each [parameter](#) of the *fnChannelExit* function.

Order Selection	<p>Determines what orders of the Account are monitored.</p> <ul style="list-style-type: none"> ● All: all open orders on the account. ● Symbol: only open orders matching the selected symbol. ● Order Type: only open orders of the selected order type. ● MagicNumber: only open orders with a matching magicnumber. ● Exact Ticket: only open orders with the matching order ticket. <p>Note: Selecting Exact Ticket disables all other selection criteria.</p>
startBar	The bar at which the channel begins
endBar	The bar at which the channel ends
channelType	The type of value to use: HIGH or LOW

The screenshot shows the configuration window for the *fnChannelExit1* function. The 'Order Selection' section includes checkboxes for 'All' (checked), 'Symbol', 'Order Type', 'MagicNumber', and 'Exact Ticket'. Each of these has a corresponding input field or dropdown menu. The 'startBar' parameter is set to 0, 'endBar' to 10, and 'channelType' to MODE_LOW. The 'Name' field at the bottom contains the text 'fnChannelExit1'. Navigation buttons include a help icon, 'Done', 'OK', and '>> Details'.

fnTrailingExit

The **Profit Exit** function *fnTrailingExit* executes a simple trailing stop.

The *fnTrailingExit* function is the same as the function [fnTrailingExit](#) found in the Trade menu of the [Toolbox](#).

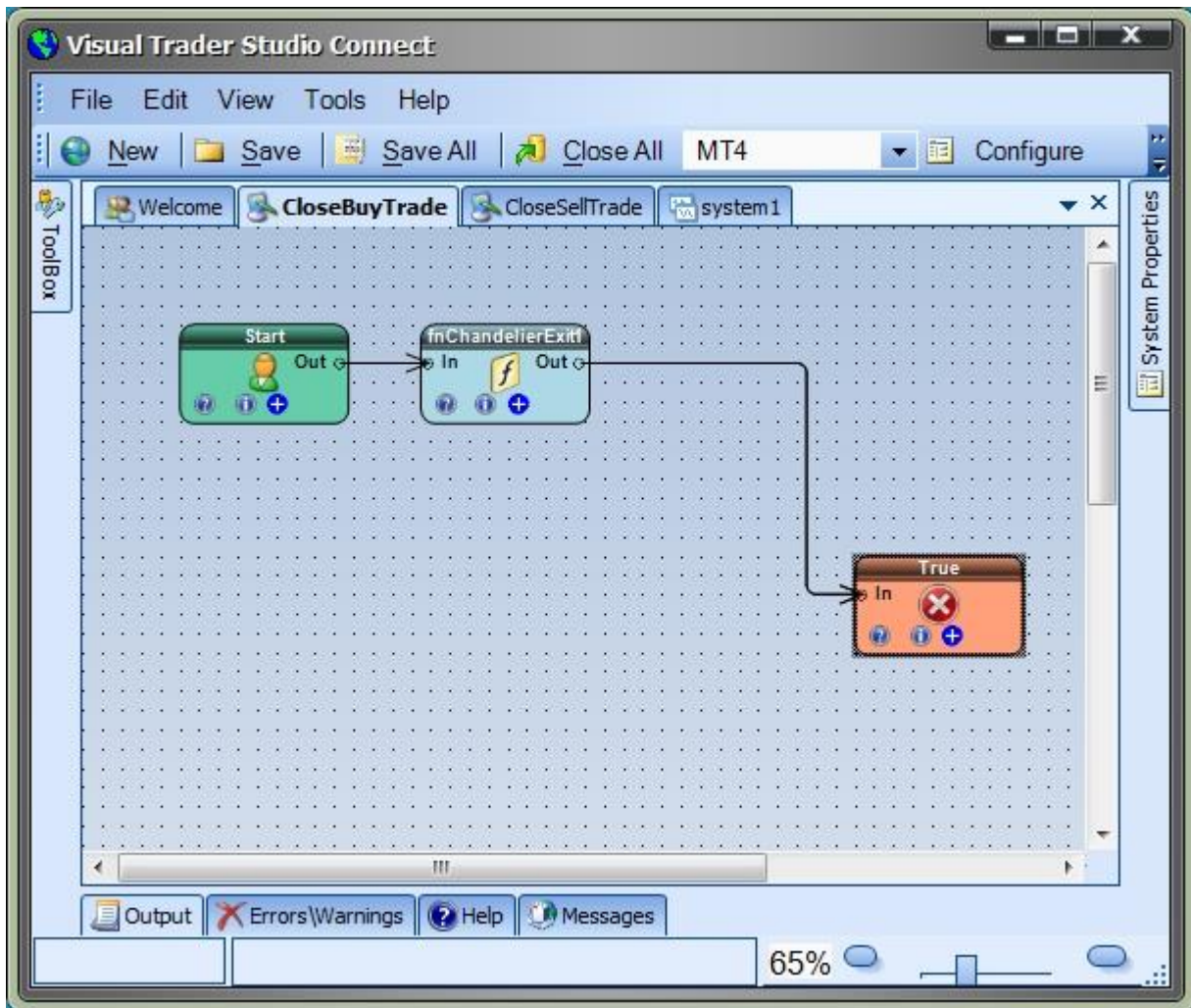
fnTrailingExit is included in the **Profit Exit** plug-in for completeness.

The following table provides information about each [parameter](#) of the *fnTrailingExit* function.

Order Selection	<p>Determines what orders of the Account are monitored.</p> <ul style="list-style-type: none"> • All: all open orders on the account. • Symbol: only open orders matching the selected symbol. • Order Type: only open orders of the selected order type. • MagicNumber: only open orders with a matching magicnumber. • Exact Ticket: only open orders with the matching order ticket. <p>Note: Selecting Exact Ticket disables all other selection criteria.</p>
stoploss	The value at which to trail the open position.

Using the Profit Exit Plug-in

The functions of the Profit Exit plug-in are used like any other functions in VTS. They can be dragged, dropped and connected in any way.



Logic can be implemented to employ different exits based on a characteristic of the trade.

For example, a [Channel Exit](#) may be used to start a trade, but after a period of time, or a certain amount of profit a, the exit may be switched to a [Chandelier Exit](#) or a simple [Trailing stop](#).

IMPORTANT NOTE ABOUT INVALID STOPS (MQL Error 130)

The MQL error 130 occurs when an attempt is made to modify an order with an invalid stoploss or takeprofit value.

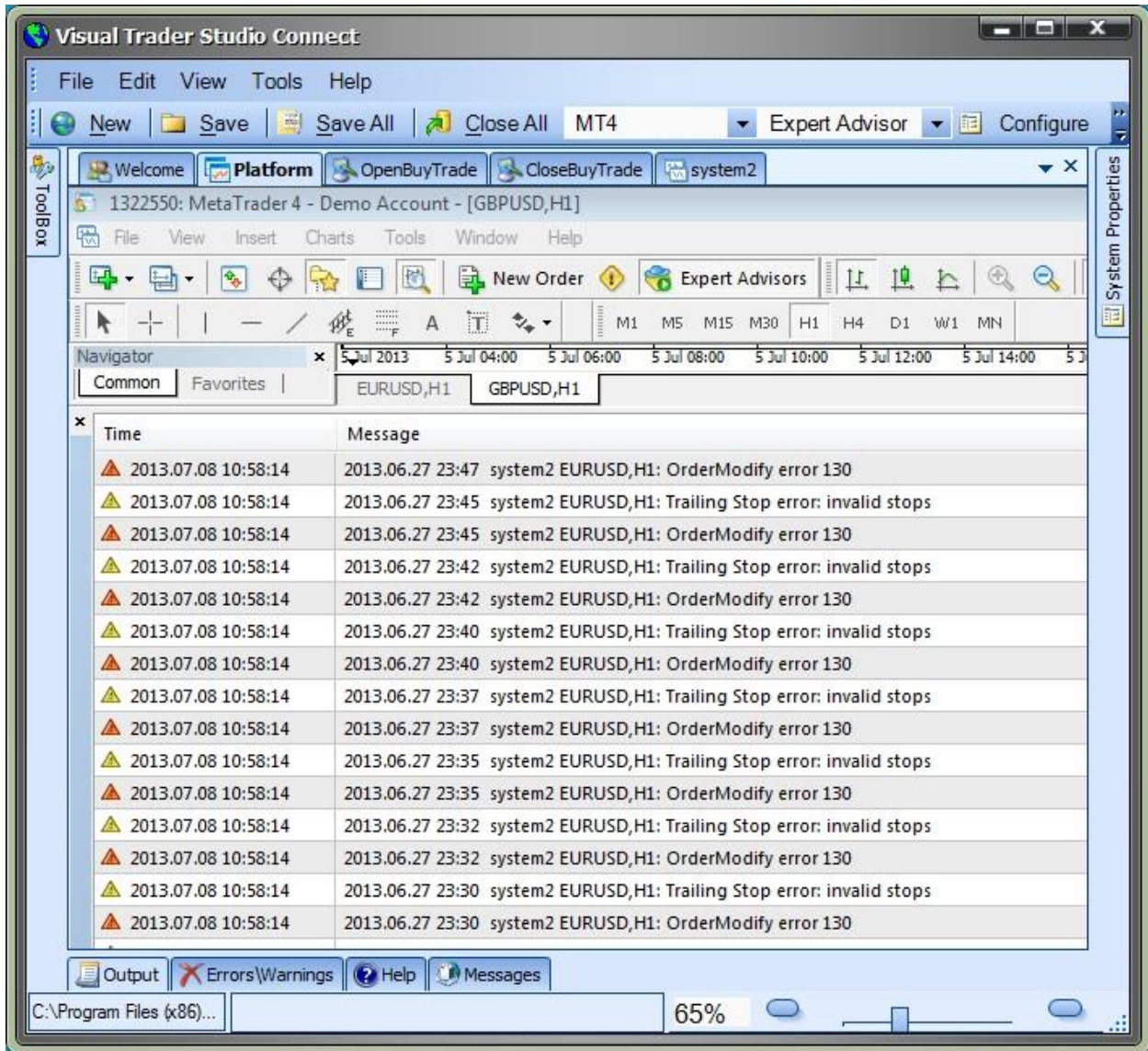
An "invalid value" can mean zero, too small, or too large. The actual allowable values are determined by the broker and the type of account (standard, mini, or micro).

The error is written to the [Journal](#) tab when testing and the [Experts](#) tab when running live.

In many cases, when using advanced exits such as the ones found in the Profit Exit Plug-in, it is normal to see some "invalid stop" errors.

This is because the value of the stoploss is being calculated on each tick and may be outside of the allowable value for the stoploss.

It is permissible for the EA to run this way until the stoploss value is valid or the trade is closed.



Profit Manager Plug-in

Requires VTS-Connect minimum version **4.0.0.49**

The **Profit Manager Plug-in** allows you to monitor all open positions of your Expert Advisors and automatically close positions based on a profit target or a maximum loss value.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.

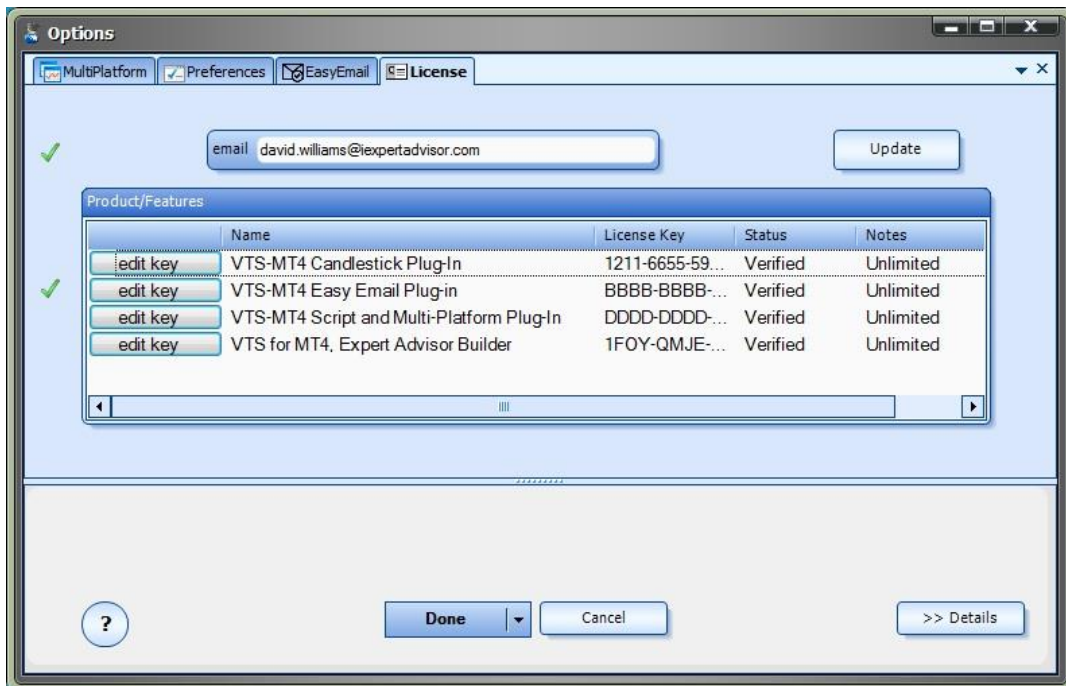


Enable the Profit Manager Plug-in

You must enter your License key to enable the **Profit Manager Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

- The **email** address is the email address used to purchase [VTS](#).
- The **License Key** is the key listed in the Members Area.
- The **Update** button is used to verify the email address and license key.
- The **edit key** button is used edit the key value.



Profit Manager in the OpenTrade Manager

Opening the Profit Manager

The *Profit Manager* is located in the [OpenTradeSystem Manager](#).



Click on the [OpenTrade](#) Manager icon to launch the [OpenTrade](#) manager. The *Profit Manager* is found on the **Profit** tab of the [OpenTrade](#) Manager.

OpenTrade (Manager type OpenTrade)

Trade | Maximum | Management | EzEmail | **Profit**

Enable Profit Manager

Target

Profit Target + 45.00 Choose ... Show as input

Maximum Loss - 50.00 Choose ... Show as input

Lookback Minutes + Choose ... Show as input

Order Selection

All

Symbol CHART

Order Type OP_BUY

MagicNumber MagicNumber Choose ...

Exact Ticket Choose ...

Action

Action Type CloseAllAndNoOpen

Reset Type MinuteDelay 15 Choose ...

Name:

Open Trade

? Done Cancel >> Details

Using the Profit Manager

The **Profit Manager** is used to monitor the overall profit of multiple open trades, and if configured, can include the results of recently closed trades.

The set of trades monitored by the **Profit Manager** is determined by the [Order Selection](#) configuration. The [Order Selection](#) configuration allows you to define the exact set of trades that you wish to manage.

When a predefined **profit target** or **maximum loss** is reached, the **Profit Manager** will close any open trades that meet the [Order Selection](#) criteria. Additionally, the **Profit Manager** can be configured to prevent further trades from opening until the EA is reset, or a defined number of minutes or candles have passed.

The **Profit Manager** configuration window contains three sections: [Target](#), [Order Selection](#) and [Action](#).

OpenTrade (Manager type OpenTrade)

Trade Maximum Management EzEmail Profit

Enable Profit Manager

Target

Profit Target + 45.00 Choose ... Show as input

Maximum Loss - 50.00 Choose ... Show as input

Lookback Minutes + Choose ... Show as input

Order Selection

All

Symbol CHART

Order Type OP_BUY

MagicNumber MagicNumber Choose ...

Exact Ticket Choose ...

Action

Action Type CloseAllAndNoOpen

Reset Type MinuteDelay 15 Choose ...

Name: Open Trade

? Done Cancel >> Details

The Profit Manager Target

The **Target** section of the **Profit Manager** is used to define the **Profit Target**, **Maximum Loss** and **LookbackMinutes** values. Each of these values is optional and can be disabled by selecting the named checkbox.

The screenshot shows a window titled "Target" with three rows of configuration options:

Option	Sign	Value	Action	Show as input
<input checked="" type="checkbox"/> Profit Target	+	45.00	Choose ...	<input type="checkbox"/>
<input checked="" type="checkbox"/> Maximum Loss	-	50.00	Choose ...	<input type="checkbox"/>
<input type="checkbox"/> Lookback Minutes	+		Choose ...	<input type="checkbox"/>

- **Profit Target**

- The **Profit Target** is the value (in the base currency of your account) that when reached will cause the **Profit ManagerAction** to occur. For example, if the **Profit Target** is \$45.00, the **Action** will occur when the calculated profit (as defined by the **Order Selection** criteria) reaches or **exceeds** \$45.00.
- The **Profit Target** is a positive number and must be entered **without a plus or minus sign.**
- The **Choose** button may be used to enter the name of any variable into the **Profit Target** text box.
- The **Show as input** check box allows the variable **Profit Target** to be defined as an **input** parameter when running the Expert Advisor. **NOTE:** Only numerical values can be set in the **Profit Target** text box to use the **Show as input** feature.

- **Maximum Loss**

- The **Maximum Loss** is the value (in the base currency of your account) that when reached will cause the **Profit ManagerAction** to occur. For example, if the **Maximum Loss** is \$50.00, the **Action** will occur when the calculated loss (as defined by the **Order Selection** criteria) reaches or goes **below** -\$50.00. Note, the value is entered as "50.00", **NOT** "-50.00".
- The **Maximum Loss** is a positive number and must be entered **without a plus or minus sign.**
- The **Choose** button may be used to enter the name of any variable into the **Maximum Loss** text box.
- The **Show as input** check box allows the variable **Maximum Loss** to be defined as an **input** parameter when running the Expert Advisor. **NOTE:** Only numerical values can be set in the **Maximum Loss** text box to use the **Show as input** feature.

- **Lookback Minutes**

- The **Lookback Minutes** value is used to include recently closed trades in the profit or loss calculation. When enabled, any trades **closed** within **Lookback** minutes that meet the Order Selection criteria are included in the total profit or loss calculation.
- The **Lookback Minutes** is a positive number and must be entered **without a plus or minus sign.**
- The **Choose** button may be used to enter the name of any variable into the **Lookback Minutes** text box.

- The **Show as input** check box allows the variable **Lookback Minutes** to be defined as an [input](#) parameter when running the Expert Advisor. **NOTE:** Only numerical values can be set in the **Lookback Minutes** text box to use the **Show as input** feature.

The Profit Manager Order Selection

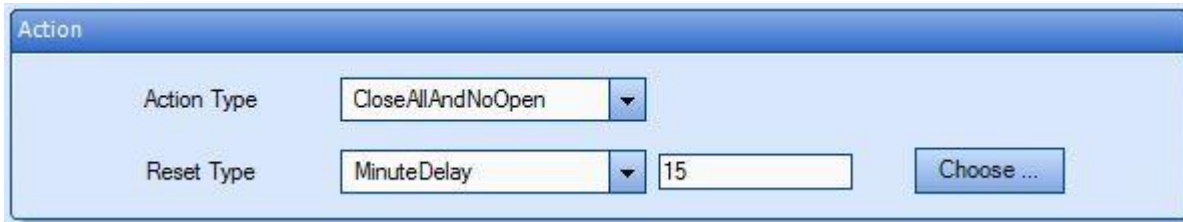
The **Order Selection** section of **Profit Manager** allows you to define exactly what orders (also called *Trades* or *Positions*) are monitored by the **Profit Manager**.

- The checkbox **All** will manage all open positions on the account and will disable all other check boxes,
- The checkbox **Exact Ticket** will manage only a single position whose ticket matches the ticket value and will disable all other check boxes.
- The remaining check boxes, **Symbol**, **Order Type** and **MagicNumber**, can be used individually or in combination to manage the exact set of orders that you wish.

Order Selection	<p>Determines what orders of the Account are managed or monitored.</p> <ul style="list-style-type: none"> ● All: all open orders on the account. ● Symbol: only open orders matching the selected symbol. ● Order Type: only open orders of the selected order type. ● MagicNumber: only open orders with a matching magicnumber. ● Exact Ticket: only open orders with the matching order ticket. <p>Note: Selecting Exact Ticket disables all other selection criteria.</p>
------------------------	---

The Profit Manager Action

The **Action** section of **Profit Manager** is used to define the **Action** to occur when the **Profit Target** or **Maximum Loss** is reached.



The screenshot shows a window titled "Action" with a light blue background. It contains two rows of controls. The first row is labeled "Action Type" and has a dropdown menu with "CloseAllAndNoOpen" selected. The second row is labeled "Reset Type" and has a dropdown menu with "MinuteDelay" selected. To the right of the "Reset Type" dropdown is a text box containing the number "15". To the right of the text box is a button labeled "Choose ...".

- **Action Type**

- The **Action Type** can be defined as **CloseAllAndNoOpen** or **NoOpen**.
 - **NoOpen** will not close any open trades, but it will prevent any further trades from opening.
 - **CloseAllAndNoOpen** will close all open trades and prevent further trades from opening.

- **Reset Type**

- The **Reset Type** can be defined as **Never**, **MinuteDelay** or **CandleDelay**.
 - **Never** will prevent trades from opening until the Expert Advisor has been manually reset.
 - **MinuteDelay** will prevent trades from opening until the number of minutes defined in the value text box has expired.
 - **CandleDelay** will prevent trades from opening until the number of candles defined in the value text box has expired.
 - The [Choose](#) button can be used to set the value text box to any variable.

The Profit Manager on a Price Chart

When the **Profit Manager** is running, the Expert Advisor displays information on the price chart.



The following values are displayed on each tick:

- The current calculated profit
- The defined Profit Target setting
- The defined Maximum Loss setting
- The defined Lookback setting
- The current trading state of the Expert Advisor

Client-Side Stops Plug-in

Requires VTS-Connect minimum version **4.0.0.50**

The **Client-Side Stops Plug-in** allows you to set Stoploss and Takeprofit values that are managed by your Expert Advisor and are not visible to your MetaTrader broker.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the Client-Side Stop Plug-in

You must enter your License key to enable the **Client-Side Stops Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

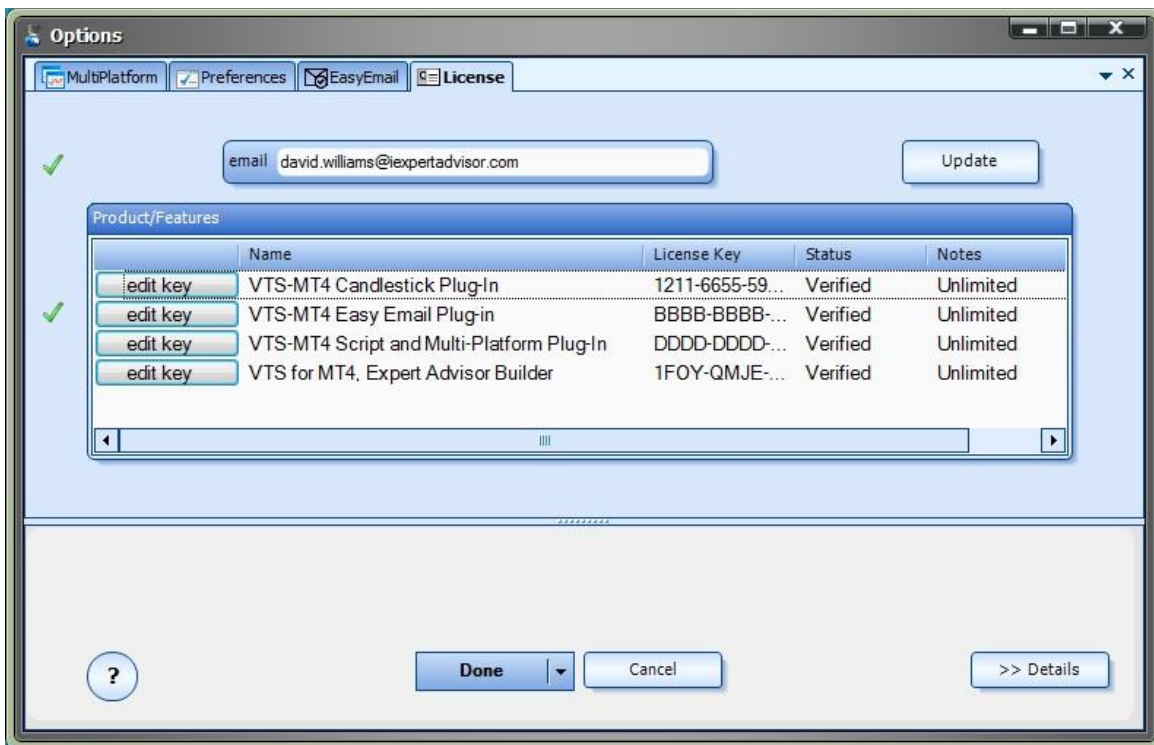
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The **email** address is the email address used to purchase [VTS](#).

The **License Key** is the key listed in the Members Area.

The **Update** button is used to verify the email address and license key.

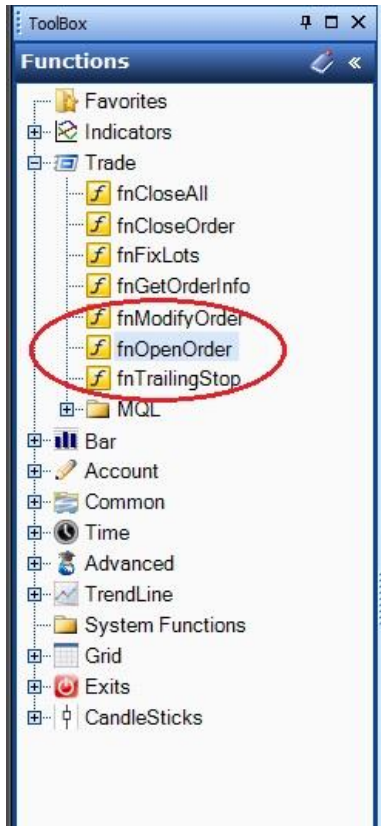
The **edit key** button is used edit the key value.



Client-Side Stops in the Toolbox

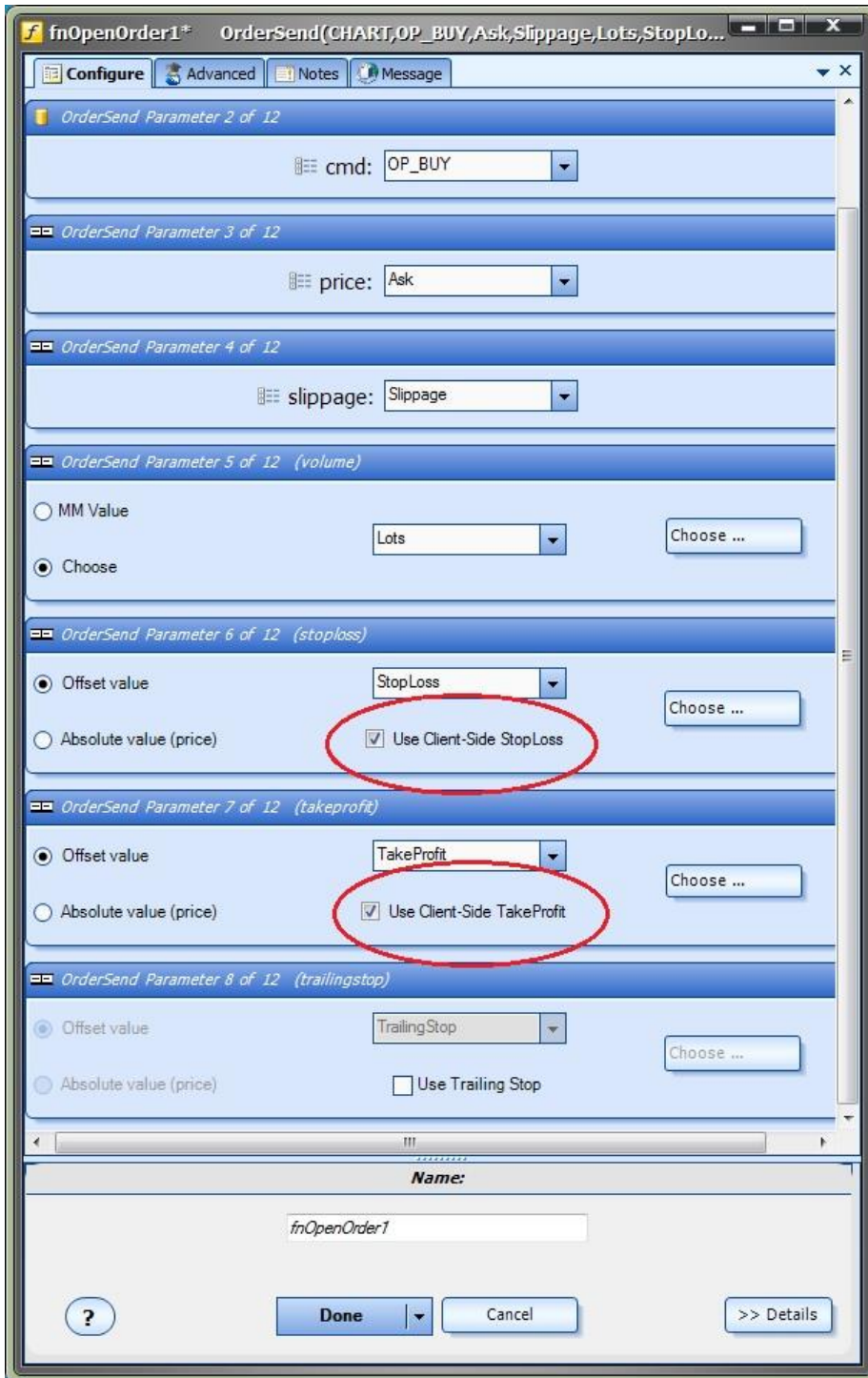
Client-Side Stops are set in the configuration window of the [fnOpenOrder](#) window.

The [fnOpenOrder](#) function is found in the Functions [Toolbox](#), under the Trade menu.



The parameter **stoploss** (parameter 6 of 12) contains a check box to "Allow Client-Side Stoploss".

The parameter **takeprofit** (parameter 7 of 12) contains a check box to "Allow Client-Side Takeprofit".



Emergency Stops

Setting Emergency Stops

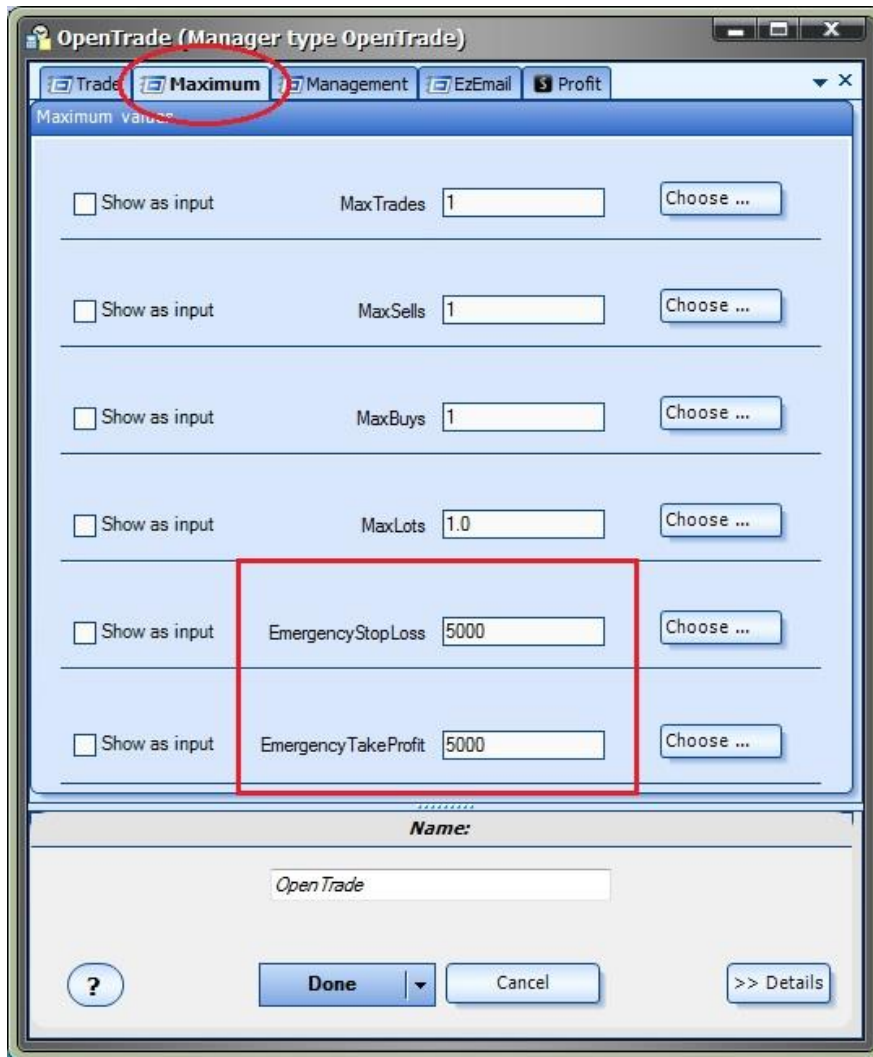
Emergency stop values are set in the *Maximum* tab of the [OpenTrade](#) Manager. The [OpenTrade](#) Manager is found in the [System Managers](#) window.



The values *EmergencyStoploss* and *EmergencyTakeProfit* are found on the Maximum tab.

Many MetaTrader brokers will not allow a trade to be opened with a zero-value stoploss or takeprofit. In this case, a very large value can be used as the stoploss and takeprofit that is assigned to the order.

These values can also be used as actual emergency stops. **In the event that your Expert Advisor is no longer running or can NOT connect to the broker's server for any reason, these emergency stops are the only values that will close the trade** (aside from a margin call !).



Using *Client-Side Stops*

VTS Client-Side stops are very easy to configure. There is checkbox on the the [fnOpenOrder](#) window for enabling a client-side stoploss and/or takeprofit.

When your Expert Advisor opens a trade for which *Client-Side Stops* have been enabled (via the fnOpenOrder window):

The EA opens the trade using the [Emergency](#) Stoploss and TakeProfit values. The default values are 5000. These can be set to any value, including zero if the MetaTrader broker allows EAs to open trades with a zero value stop.

On every tick, the EA will monitor the *Client-Side Stops* and close the trade if either the stoploss or takeprofit value has been reached.

WARNING: The EA must be attached and running for client-side stops to execute! In the event that your Expert Advisor is no longer running or can NOT connect to the broker's server for *any* reason, the [emergency stops](#) are the only values that will close the trade.

These steps outline how to use *Client-Side Stops*.

[Using fnOpenOrder](#)

[Using an Offset Value](#)

[Using an Absolute Price Value](#)

fnOpenOrder

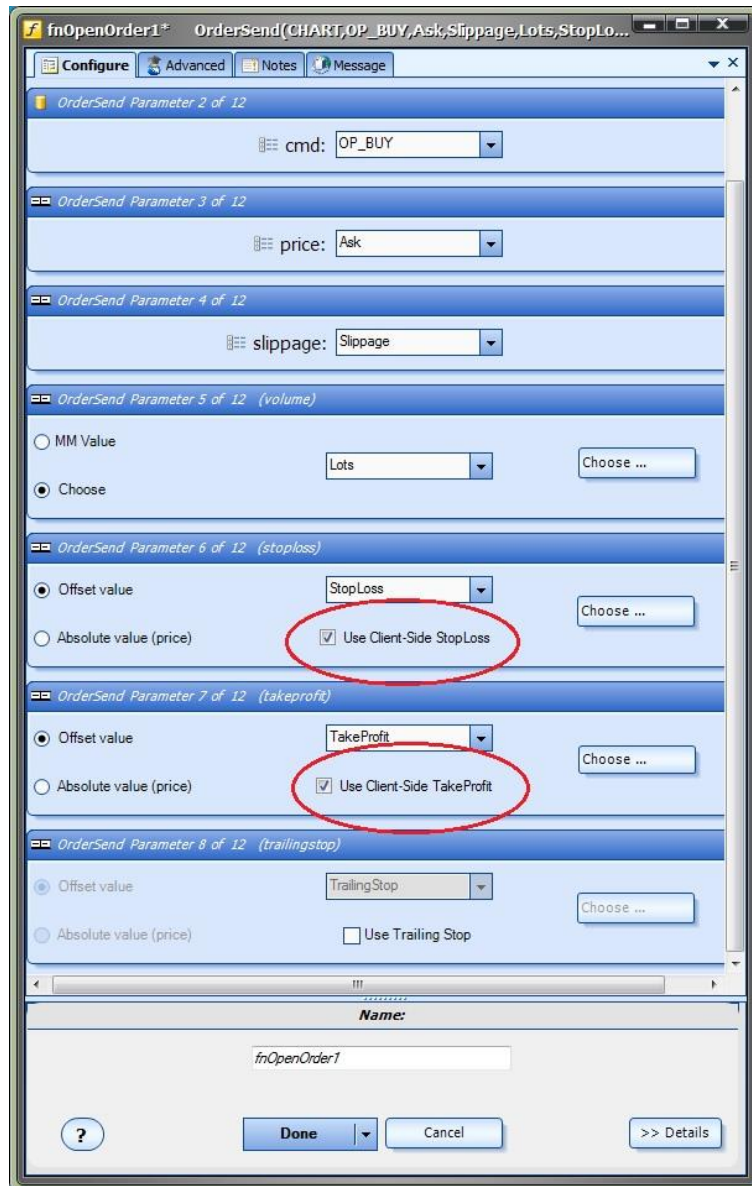
Client-Side Stops are very easy to configure. There is a checkbox on the [fnOpenOrder](#) window for enabling a client-side stoploss and takeprofit.

The parameter *stoploss* (parameter 6 of 12) contains a check box to "Allow Client-Side Stoploss".

The parameter *takeprofit* (parameter 7 of 12) contains a check box to "Allow Client-Side Takeprofit".

When *Client-Side Stops* are enabled, the [Emergency](#) stop values are used to open the trade with the broker. The normal *StopLoss* and *TakeProfit* values in VTS are used as the stop values for the *Client-Side Stop* functionality.

The *Offset* value and *Absolute* value options are available for *Client-Side Stops* and are described in the next two sections.



Using an Offset value with Client-Side Stops

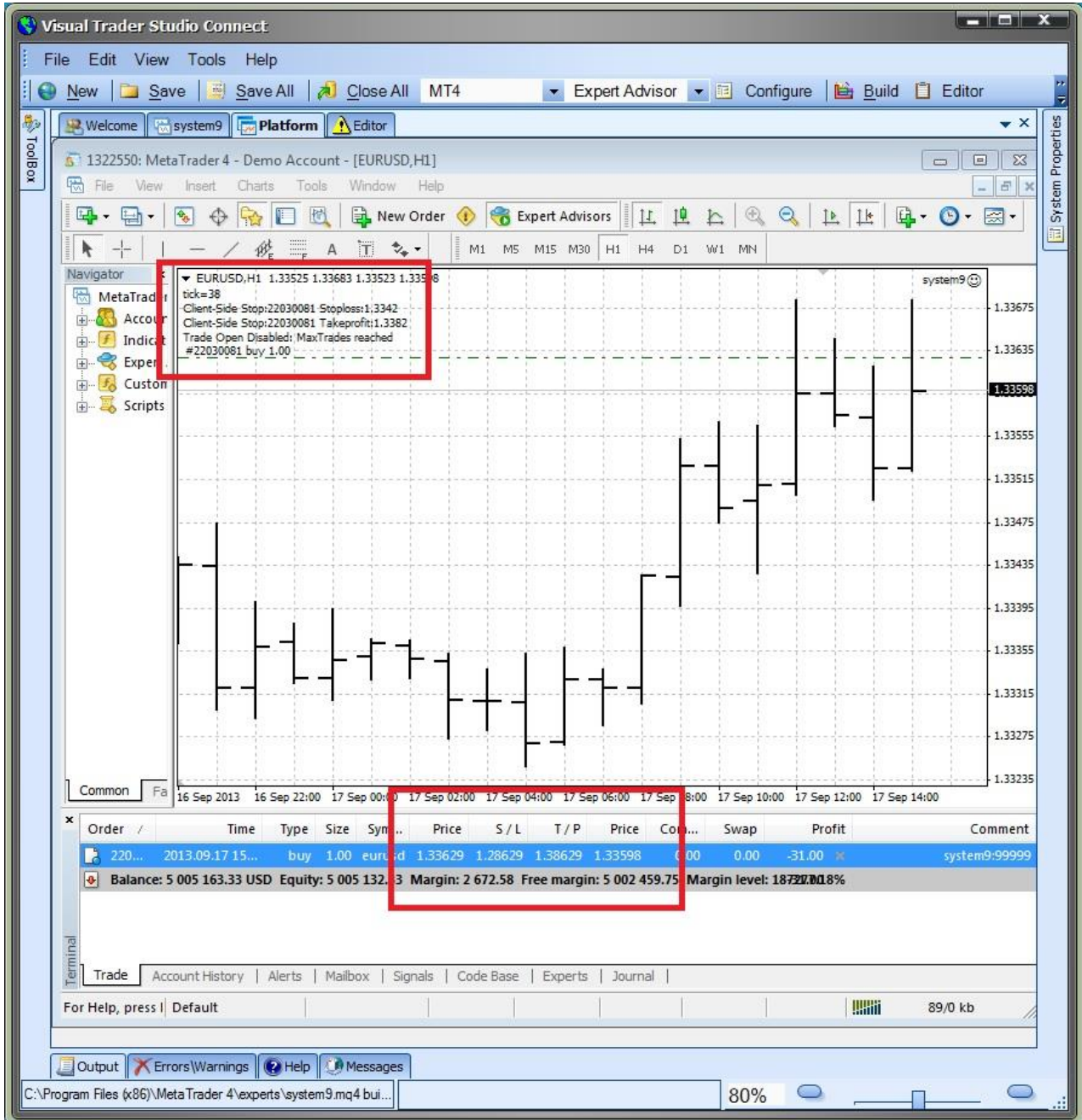
The functionality of the StopLoss and TakeProfit parameters of the [fnOpenOrder](#) function are the same when using *Client-Side Stops*.

stoploss	<p>The value at which to close the trade for a loss.</p> <p>VTS creates an extern (or Input) variable named <i>StopLoss</i> that can be used for this parameter; its default value is 200.</p> <p>A value can be manually entered into the text-box.</p> <p>A value can be selected using the Choose button.</p> <p>A value can be entered as an integer or as a price level by selecting the <i>Offset value</i> or <i>Absolute value</i> button. The <i>Absolute value</i> option is useful for using technical indicator values or price values such as <i>Low[1]</i> or <i>High[1]</i>.</p>
takeprofit	<p>The value at which to close the trade for a profit.</p> <p>VTS creates an extern (or Input) variable named <i>TakeProfit</i> that can be used for this parameter; its default value is 200.</p> <p>A value can be manually entered into the text-box.</p> <p>A value can be selected using the Choose button.</p> <p>A value can be entered as an integer or as a price level by selecting the <i>Offset value</i> or <i>Absolute value</i> button. The <i>Absolute value</i> option is useful for using technical indicator values or price values such as <i>Low[1]</i> or <i>High[1]</i>.</p>

When an *Offset value* is used, the value of the stoploss or takeprofit is captured at the time the trade is opened and it is stored locally by the EA. The stored value is used to monitor the open trade. The stored value is maintained correctly even if the EA is removed and reattached to the price chart. However, recall that the *Client-Side Stops* functionality will not execute unless the EA is attached and running.

On the chart below, the default *offset* values of 200 were used for the *Client-Side Stops* stoploss and takeprofit: These values are displayed on the top left corner of the price chart.

Notice in the *Trade* window that the actual order was opened using the default [emergency](#) stop values of 5000.



Using an Absolute Price Value

Using an Absolute Price value with Client-Side Stops

The functionality of the StopLoss and TakeProfit parameters of the [fnOpenOrder](#) function are the same when using *Client-Side Stops*.

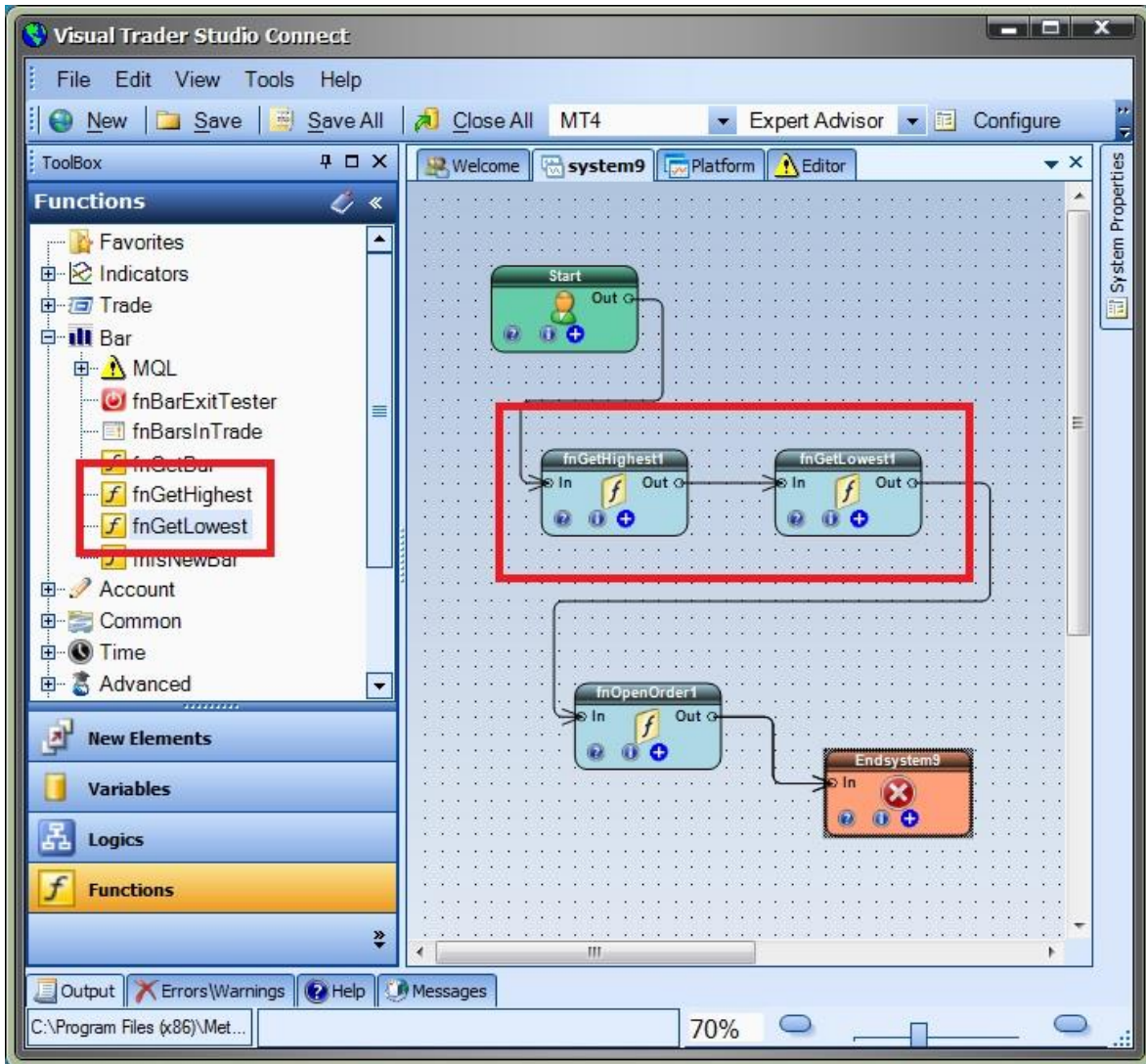
stoploss	<p>The value at which to close the trade for a loss.</p> <p>VTS creates an extern (or Input) variable named <i>StopLoss</i> that can be used for this parameter; its default value is 200.</p> <p>A value can be manually entered into the text-box.</p> <p>A value can be selected using the Choose button.</p> <p>A value can be entered as an integer or as a price level by selecting the <i>Offset value</i> or <i>Absolute value</i> button. The <i>Absolute value</i> option is useful for using technical indicator values or price values such as <i>Low[1]</i> or <i>High[1]</i>.</p>
takeprofit	<p>The value at which to close the trade for a profit.</p> <p>VTS creates an extern (or Input) variable named <i>TakeProfit</i> that can be used for this parameter; its default value is 200.</p> <p>A value can be manually entered into the text-box.</p> <p>A value can be selected using the Choose button.</p> <p>A value can be entered as an integer or as a price level by selecting the <i>Offset value</i> or <i>Absolute value</i> button. The <i>Absolute value</i> option is useful for using technical indicator values or price values such as <i>Low[1]</i> or <i>High[1]</i>.</p>

When an *Absolute price value* is used, the value of the stoploss or takeprofit is calculated on each incoming tick. Recall that the *Client-Side Stops* functionality will not execute unless the EA is attached and running.

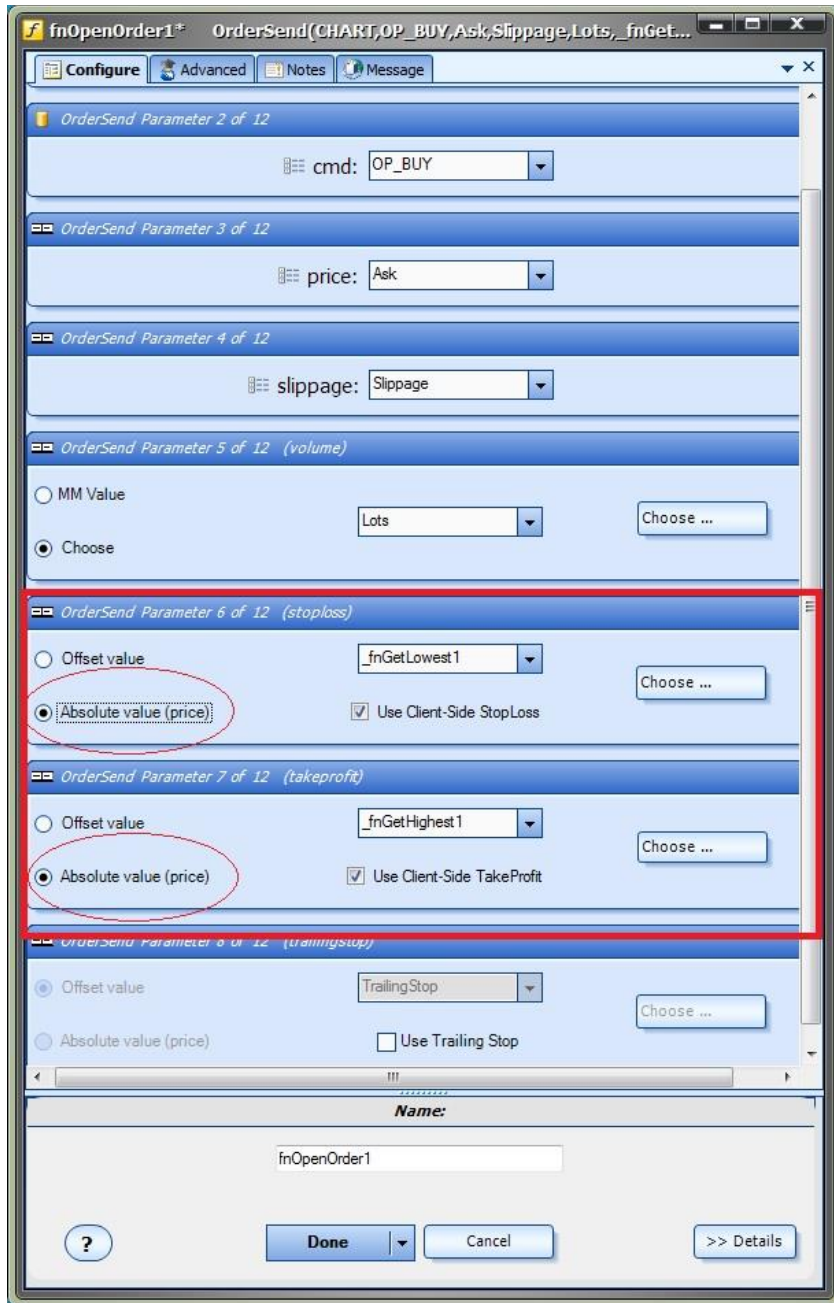
One method for using *Absolute price values* for a stoploss or takeprofit is to use the Highest or Lowest price of the last 12 bars.

This is done in VTS by using the functions *fnGetHighest* and *fnGetLowest* found under the *Bar* menu and then assigning these values as the stoploss and takeprofit values, respectively.

This is a working, simplified drawing showing the technique.

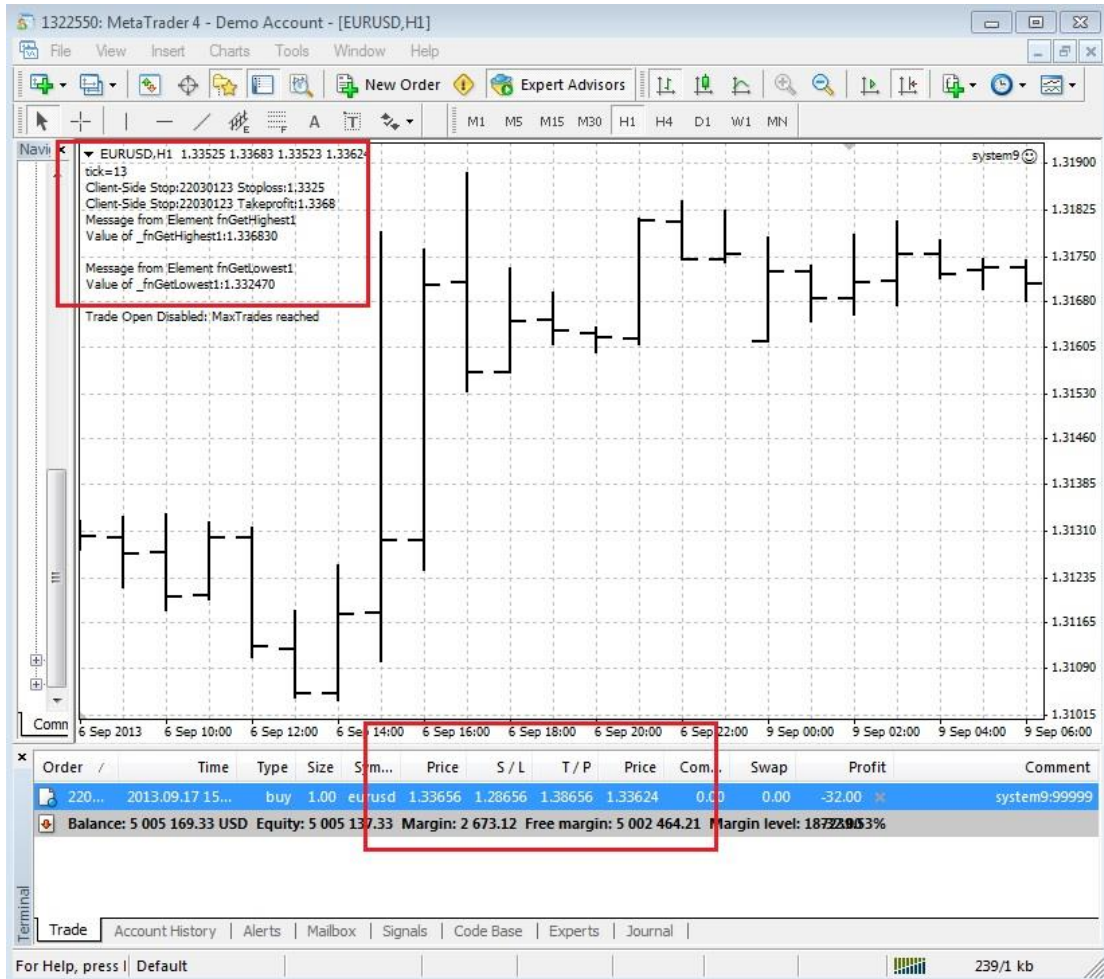


This is the [fnOpenOrder](#) configuration window.



On the chart below, the *Absolute price* values of the high and low were used for the client-side stoploss and takeprofit: These values are displayed on the top left corner of the price chart. These values are updated on every tick.

Notice in the *Trade* window that the actual order was opened using the default [emergency](#) stop values of 5000.



Fibonacci Trader Plug-in

Requires VTS-Connect minimum version 4.0.0.51

The *Fibonacci Trader Plug-in* allows an Expert Advisor to detect if any levels of a manual or automatically drawn Fibonacci retracement have been broken.

What is a Plug-in?

VTS stands for *Visual Traders Studio*.

The VTS *Expert Advisor Builder* is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A *VTS Plug-in* allows traders to easily implement advanced trading techniques using an add-on user interface.



Enabled the Fibonacci Trader Plug-in

You must enter your License key to enable the *Fibonacci Trader Plug-in*. Your license key for all of your VTS products can be found in the [Members Area](#).

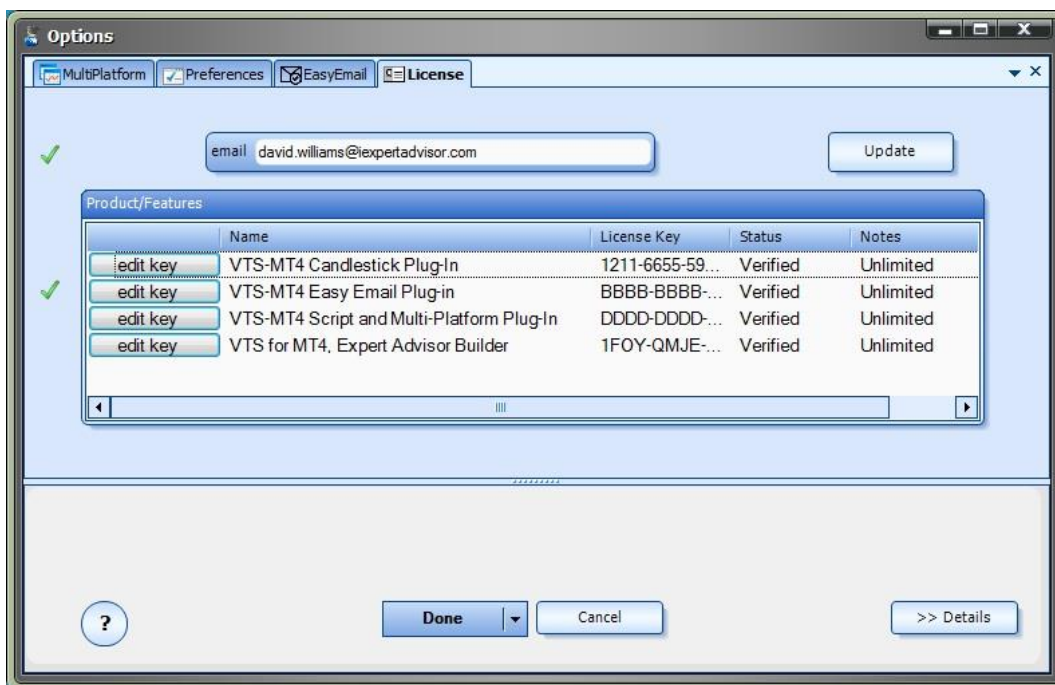
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The email address is the email address used to purchase [VTS](#).

The License Key is the key listed in the Members Area.

The Update button is used to verify the email address and license key.

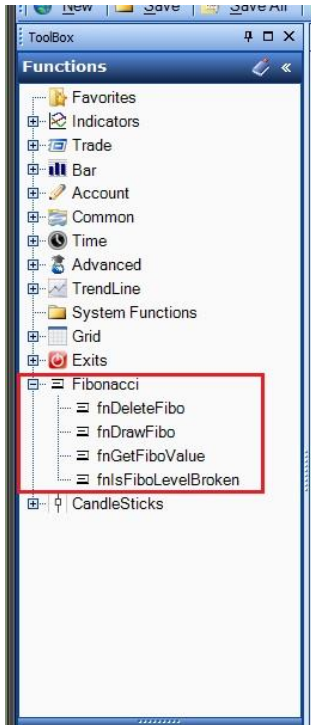
The edit key button is used edit the key value.



Fibonacci Trader Functions in the Toolbox

Once enabled, the Fibonacci functions are available in the [Toolbox](#) Function tab under the *Fibonacci* menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.



Fibonacci Trader Functions

The Fibonacci function library include these functions:

[fnDrawFibo](#)

Used to programatically draw a Fibonacci retracement on a price chart.

[fnIsFiboLevelBroken](#)

Used to determine if a price value has broken through a Fibonacci retracement level.

[fnGetFiboValue](#)

Used to get the value of a Fibonacci retracement level..

[fnDeleteFibo](#)

Used to programatically delete a Fibonacci retracement from a price chart.

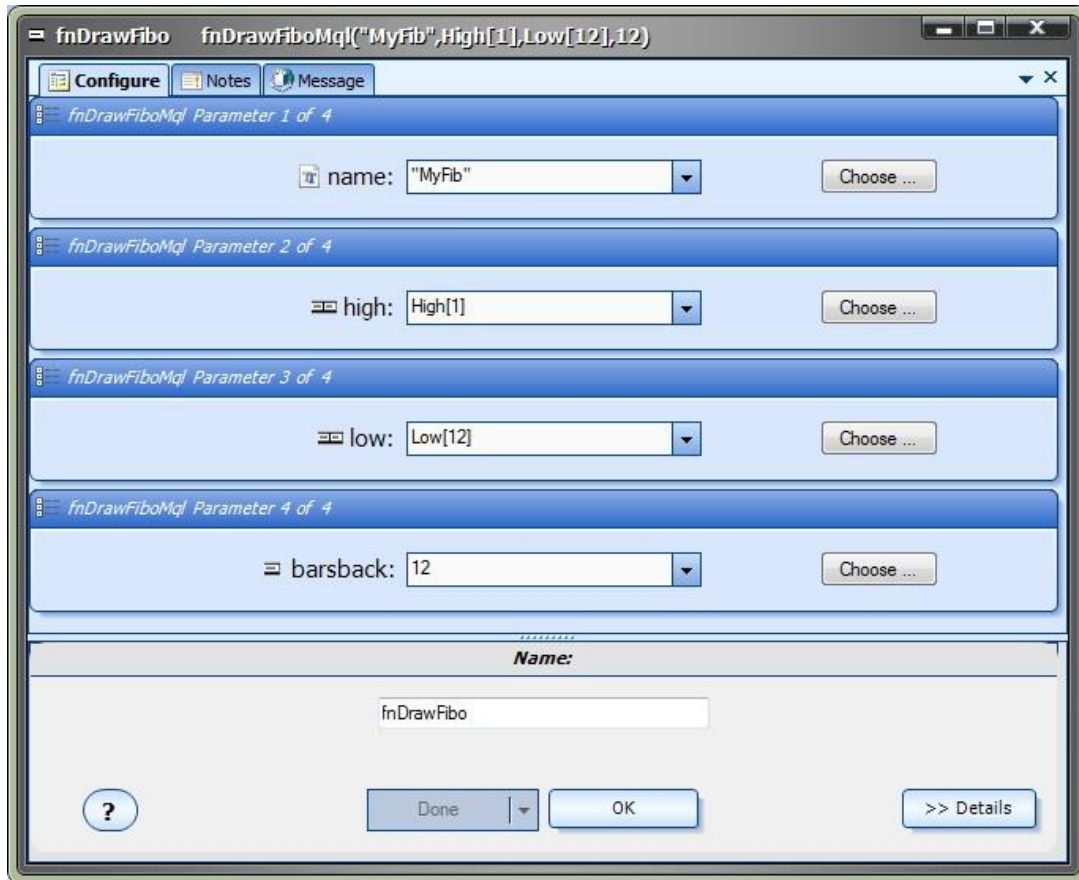
fnDrawFibo

The Fibonacci function *fnDrawFibo* is used to programatically draw a Fibonacci retracement on a price chart.

After the *fnDrawFibo* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
name	string	The name of the Fibonacci retracement. All Fibonacci retracements are identified by name. The name should be unique. The pull-down menu provides sample names, but the text can be overwritten to any allowable name. The name must be surrounded by double quotes, for example "myName". Avoid using blank spaces in the name: It may cause unpredictable behaviour.
high	Double	The highest point of a Fibonacci retracement.
low	Double	The lowest point of a Fibonacci retracement
barsback	Integer	The number of bars back to begin the drawing of the Fibonacci retracement



fnDeleteFibo

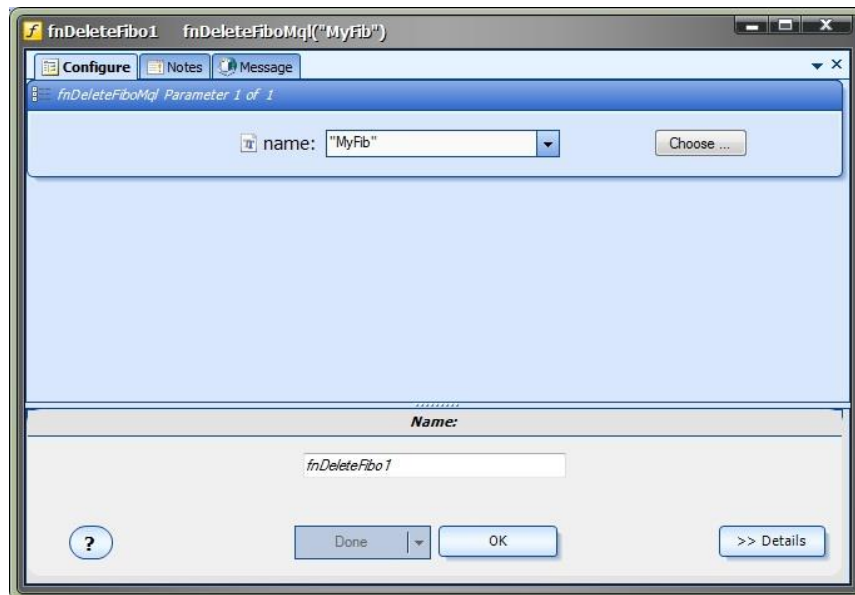
The *fnDeleteFibo* function is used to programmatically delete a Fibonacci retracement from a price chart.

After the *fnDeleteFibo* function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
name	string	The name of the Fibonacci retracement. All Fibonacci retracements are identified by name. The name should be unique. The pull-down menu provides sample names, but the text can be overwritten to any allowable name. The name must be surrounded by double quotes, for example "myName". Avoid using blank spaces in the name: It may cause unpredictable behavior.

Note: Any and all Fibonacci retracements that match the name are deleted from the price chart. This includes Fibonacci retracements created both manually and programmatically.



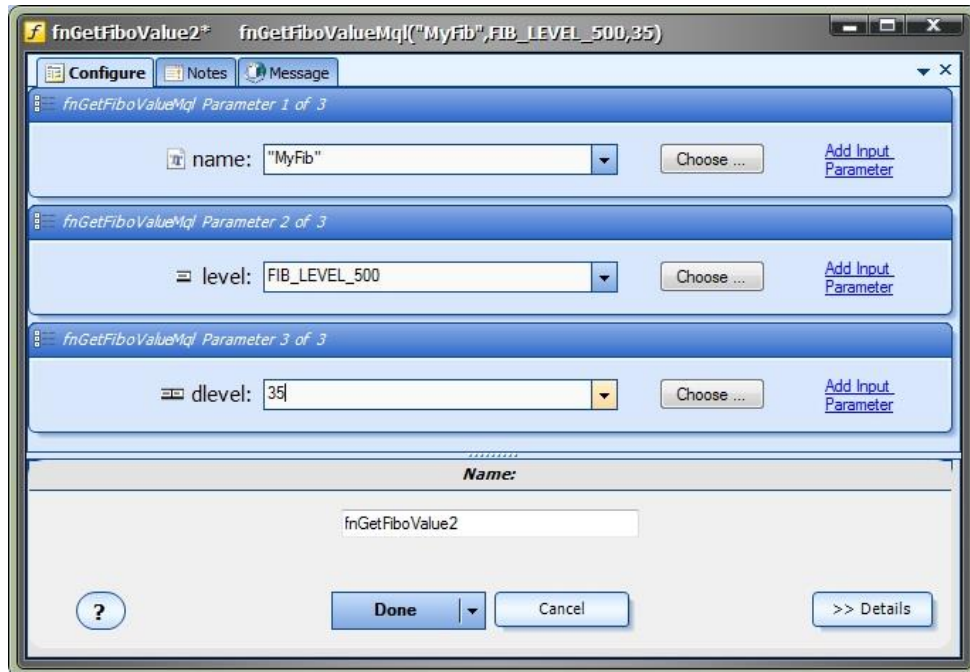
fnGetFiboValue

The *fnGetFiboValue* function is used to get the value of a Level of a Fibonacci retracement.

After the *fnGetFiboValue* function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
name	string	The name of the Fibonacci retracement. All Fibonacci retracements are identified by name. The name should be unique. The pull-down menu provides sample names, but the text can be overwritten to any allowable name. The name must be surrounded by double quotes, for example "myName". Avoid using blank spaces in the name: It may cause unpredictable behaviour.
level	LEVEL value (integer)	The Fibonacci retracement Level of which to to get the value. The available levels are: FIB_LEVEL_0 FIB_LEVEL_21.6 FIB_LEVEL_38.2 FIB_LEVEL_50.0 FIB_LEVEL_61.8 FIB_LEVEL_100.0 FIB_LEVEL_161.8 FIB_LEVEL_261.8 FIB_LEVEL_423.6
dlevel	double	This can be any user-define percentage. For example: 25% is expressed as 25.0. NOTE: If the dlevel parameter is set to <i>any</i> value besides 0, dlevel will be used to get the price and level will be ignored.



fnIsFiboLevelBroken

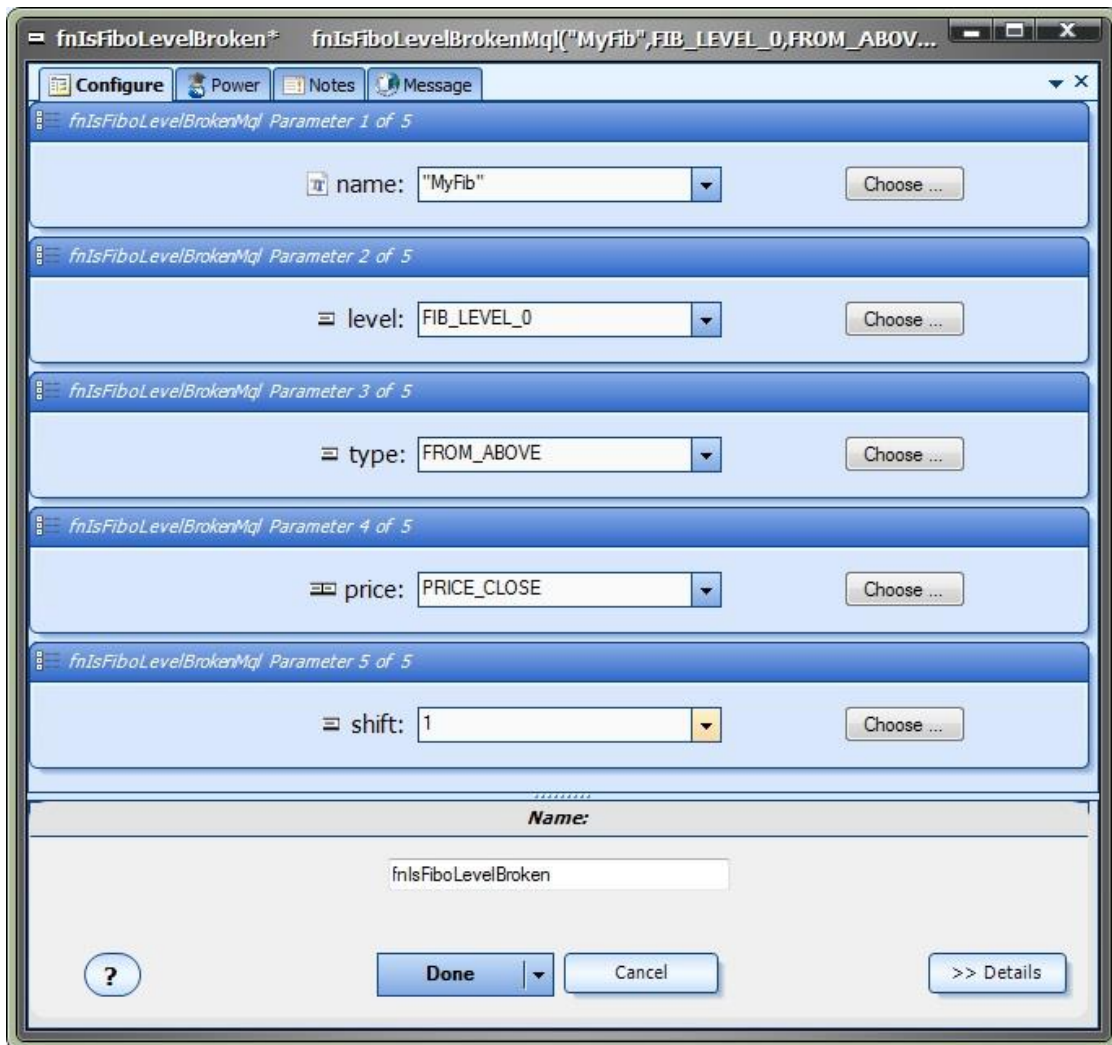
The *fnIsFiboLevelBroken* function is used to determine if a price value has broken through a Fibonacci retracement level.

After the *fnIsFiboLevelBroken* function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
name	string	The name of the trend line. All trend lines are identified by name. The name should be unique. The pull-down menu provides sample names, but the text can be overwritten to any allowable name. The name must be surrounded by double quotes, for example "myName". Avoid using blank spaces in the name: It may cause unpredictable behaviour.
level	LEVEL value (integer)	The Fibonacci retracement Level of which to get the value. The available levels are: FIB_LEVEL_0 FIB_LEVEL_21.6 FIB_LEVEL_38.2 FIB_LEVEL_50.0 FIB_LEVEL_61.8 FIB_LEVEL_100.0 FIB_LEVEL_161.8 FIB_LEVEL_261.8 FIB_LEVEL_423.6
type	FROM value (integer)	The direction from which the price value was broken. The pull-down menu offers these choices: FROM_ABOVE : The price value broke the trend line from above the line. FROM_BELOW : The price value broke the trend line from below the line.
price		The price value that breaks through the Fibonacci retracement Level. The pull-down menu offers these choices: PRICE_CLOSE PRICE_OPEN PRICE_HIGH PRICE_LOW PRICE_MEDIAN

		PRICE_TYPICAL PRICE_WEIGHTED
shift	integer	The candle index on the price chart of where to test if the trend line has been broken. Zero is the currently forming candle, one is one candle to the left, etc.



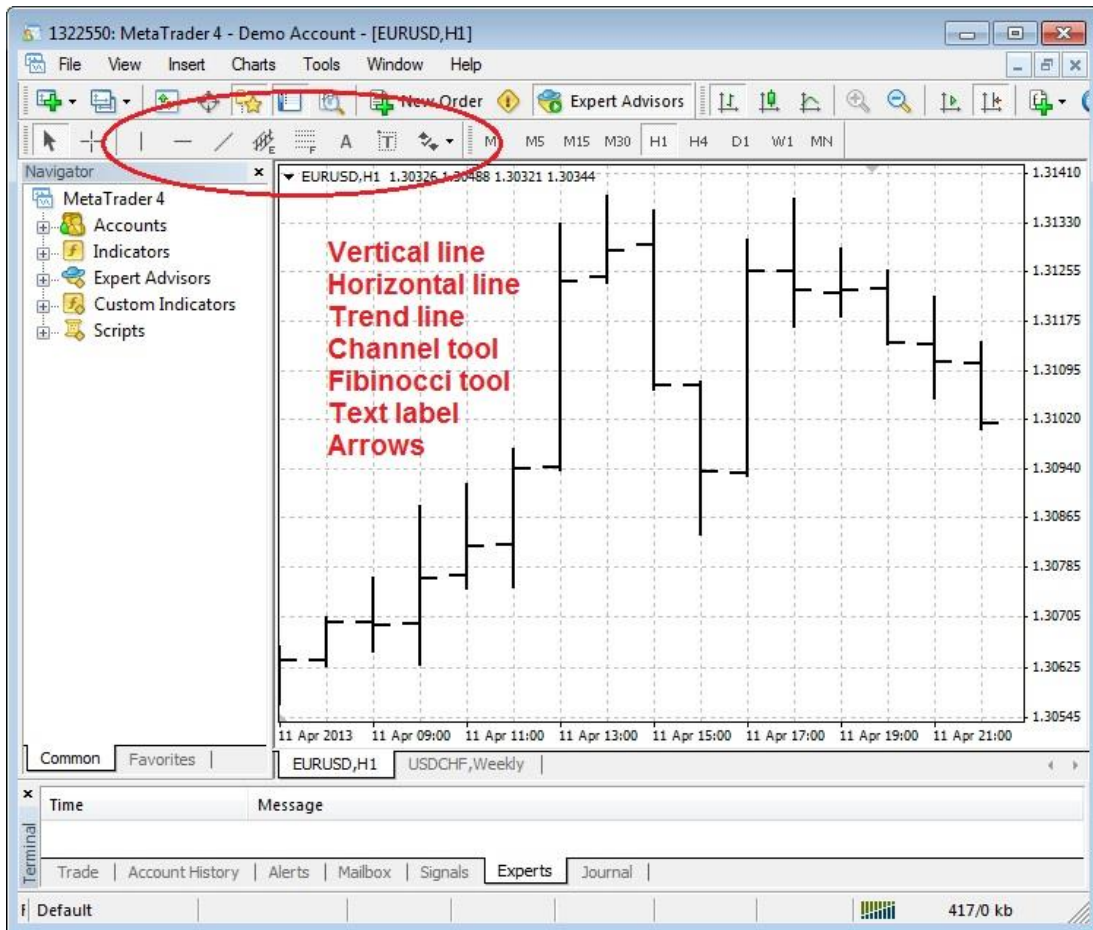
Using the Fibonacci Trader

Objects on the MetaTrader Platform

Every line, label and arrow that you see on a MetaTrader price chart is referred to as an Object. The MetaTrader platform provides a set of MQL functions for programatically creating and deleting Objects. The Object functions are available from the VTS [Function Toolbox](#) under the Advanced->Object menu. The Object functions are on the Advanced menu because they require advanced MQL knowledge and can be difficult to use.

Objects can be manually created using the toolbar on the top of the MetaTrader platform. The objects that can be created from the MT tool bar are:

- Vertical line
- Horizontal line
- Trend line
- Channels line
- Fibonacci lines
- Text
- Arrows



Manually Drawing a Fibonacci Retracement

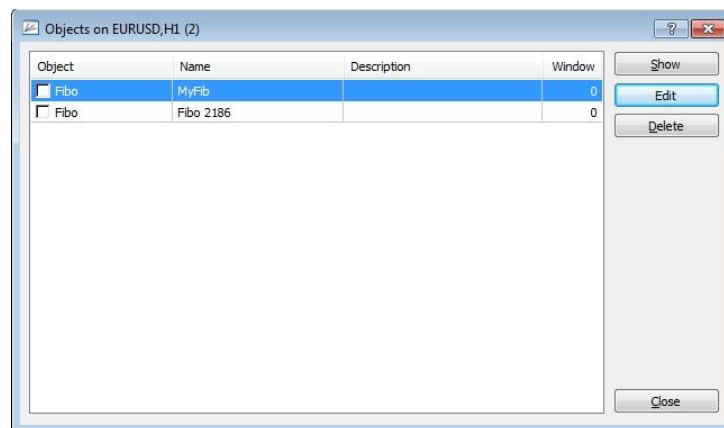
To manually draw a Fibonacci retracement on a MetaTrader price chart, click the Fibonacci retracement button, then click the start location on the chart, hold the mouse down while moving to the end location and release the mouse.

When a Fibonacci retracement is manually created, it is given a generated name by the MetaTrader platform. For example, it may be named "Fibo 1003".

The *Fibonacci Trader Plug-in* identifies Fibonacci retracements by their name. To change the name of a manually drawn Fibonacci retracement, in the MetaTrader platform go to:

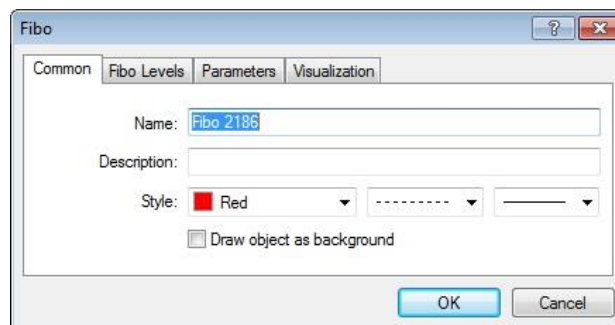
Charts->Objects->Object List

Find the Fibonacci retracement in the list, highlight it and select *Edit*.



This will allow changing the Fibonacci retracement properties including the name.

Note: The name of each Fibonacci retracement should be unique. Do not place more than one Fibonacci retracement on the same chart with the same name.

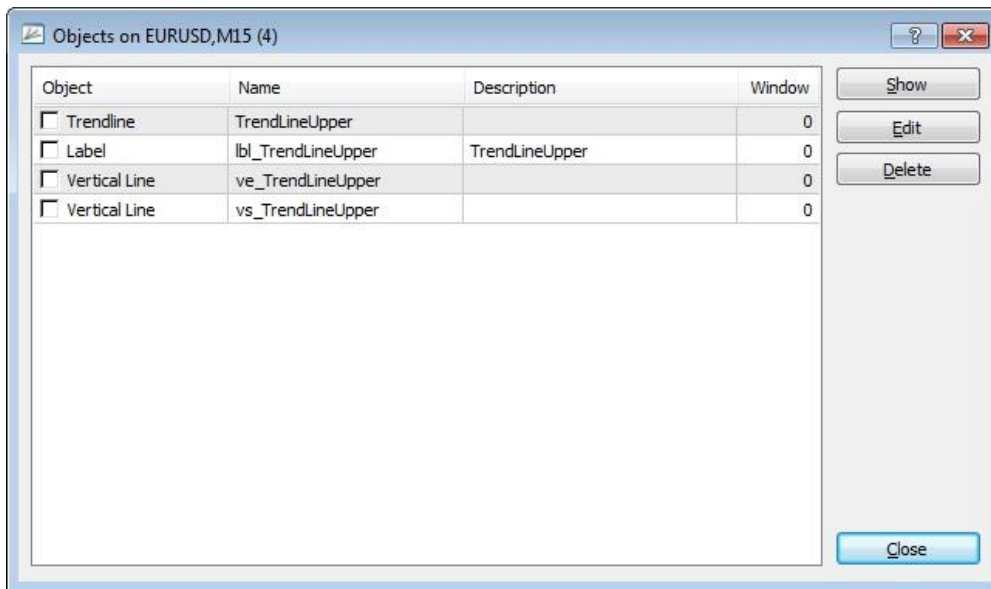


Removing a Fibonacci retracement (or any other Object)

To manually remove a single Fibonacci retracement from a MetaTrader price chart, in the MetaTrader platform go to:

Charts->Objects->Object List

select the Fibonacci retracement by name and click the Delete button.



Building an Expert Advisor to manage a manually drawn Fibonacci retracement

The *Fibonacci Trader Plug-in* can monitor any Fibonacci retracement on a chart. It does this by searching for a Fibonacci retracement by name.

After you manually [draw your Fibonacci retracement](#) on your price chart, record the exact name of the Fibonacci retracement. This name will be referenced in your Expert Advisor by the [fnIsFiboLevelBroken](#).

To build an Expert Advisor that opens a BUY trade when a Fibonacci retracement has been broken, drag and drop the Fibonacci function *fnIsFiboLevelBroken* on to the drawing pad and connect it before the [Logic](#) element.

Set the [parameters](#) of the *fnIsFiboLevelBroken* function:

name: set the value of the *name* parameter to the exact name of the Fibonacci retracement. For example "myfib".

level: set the value of the *level* parameter to the Fibonacci retracement level to test for the break.

type: set the value of the *type* parameter to the direction to test for the Fibonacci retracement break, either FROM_ABOVE or FROM_BELOW.

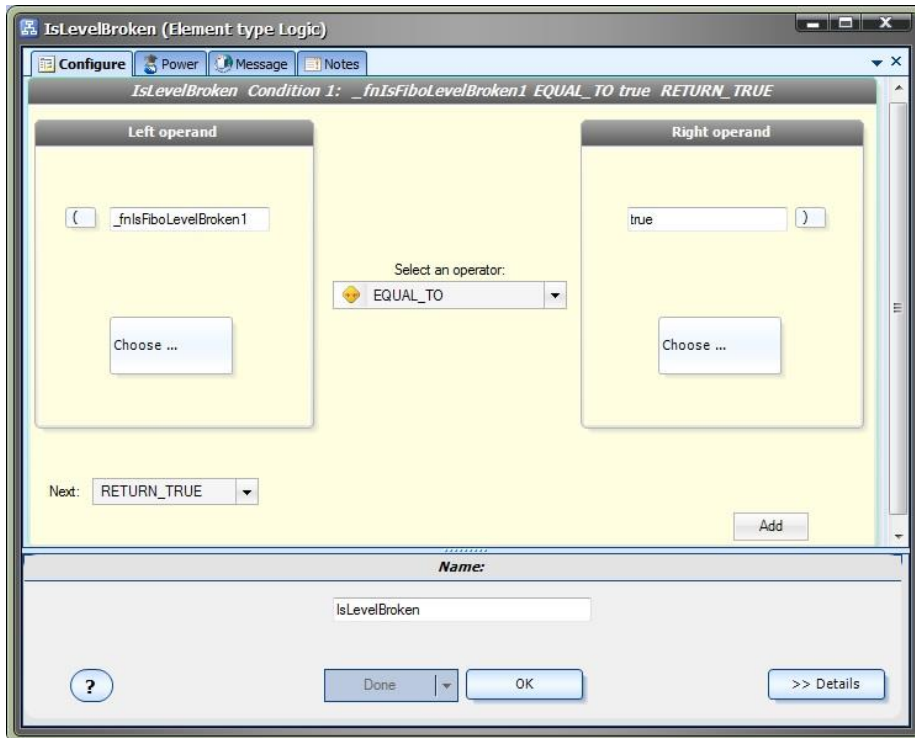
price: set the value of the *price* parameter to one of the [price constants](#), or *Bid* or *Ask*.

shift: set the value of the *shift* parameter to the index value of the candle (or bar) to test for the break.

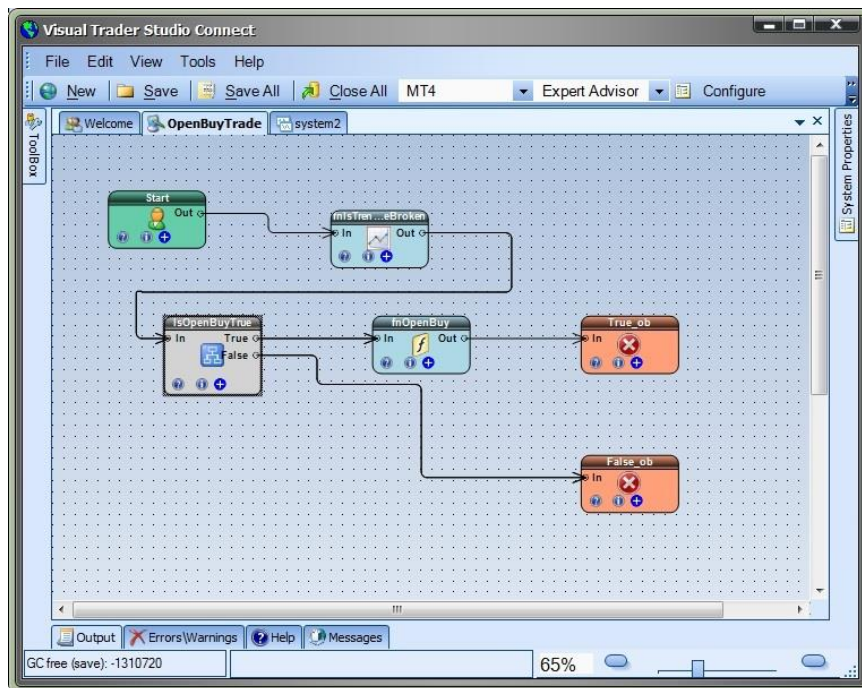
Normally, the *shift* value is set to 0 and the price value is set to *Bid* or *Ask*. This will test the latest price against the trend line on the far right edge of the chart.

Alternatively, any candle on the chart can be tested for a break.

After connecting and configuring the *fnIsFiboLevelBroken* function, configure the Logic element to test for the break.



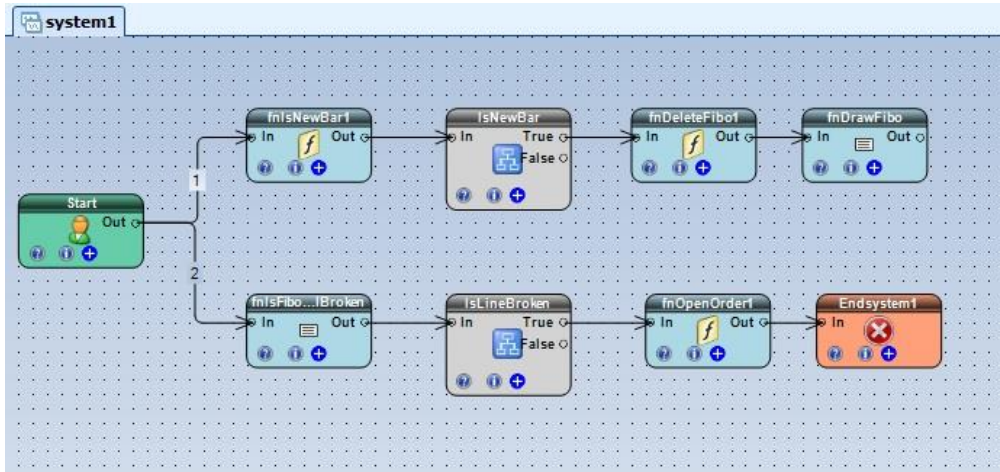
The full drawing is shown below.



Building an Expert Advisor to draw and manage a Fibonacci retracement

The *Fibonacci Trader Plug-in* can programatically draw a trend line on a price chart using the *fnDrawFibo* function.

To build an Expert Advisor that opens a BUY trade when a programatically drawn Fibonacci retracement level has been broken, refer to the drawing below:



Notice the two links coming from the [Start](#) Element. You can number any [link](#) by double-clicking it. The link number defines the order of execution.

Following Link #1:

The [Function](#) Element "[fnIsNewBar1](#)" gets a true value if a new bar has just been formed. If this EA is attached to a 1-hour price chart, a new bar will be formed at the beginning of each hour.

The [Logic](#) Element "[IsNewBar](#)" compares the value of "[fnIsNewBar1](#)" to true.

If the "[IsNewBar](#)" Logic evaluates to true, then the Function "[fnDeleteFibo1](#)" deletes the Fibonacci retracement, and the Function "[fnDrawFibo](#)" draws a new Fibonacci retracement.

Note: The Fibonacci retracements are identified by their name (e.g. "MyFib")

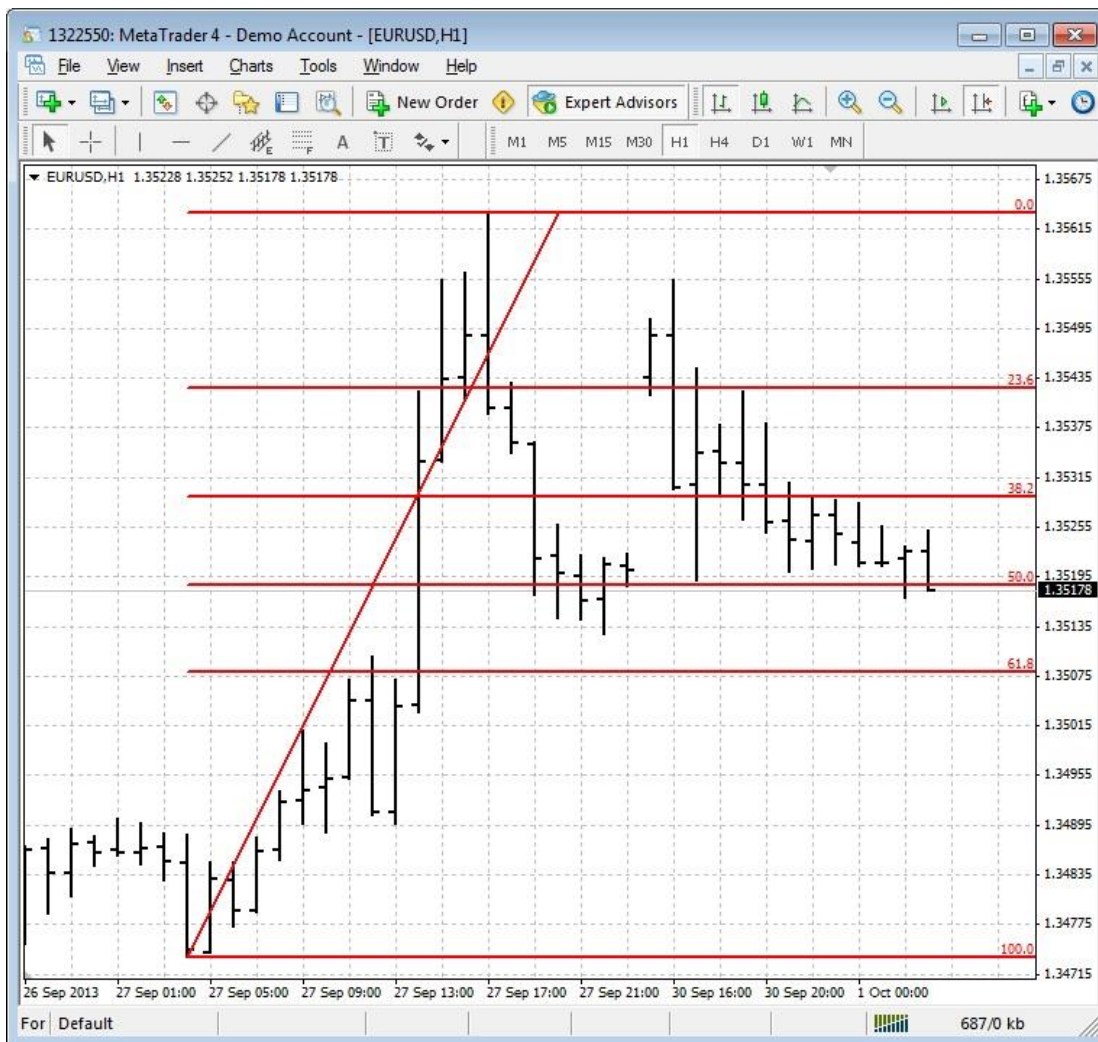
Following Link #2:

The [Function](#) Element "[fnIsFiboLevelBroken](#)" get a true value if the specified level has been broken, from the specified direction.

The [Logic](#) Element "[IsLineBroken](#)" compares the value of "[fnIsFiboLevelBroken](#)" to true.

If the "[fnIsFiboLevelBroken](#)" Logic evaluates to true, then the function "[fnOpenOrder1](#)" opens a trade.

Note: The variable MaxTrades will prevent multiple trades from being opened if "[fnOpenOrder1](#)" executes more than once.



Four coordinates are required to draw a line in two dimensional space. For example, on a typical graph, using the [Cartesian](#) coordinate system, the coordinates are usually defined as (x_1, y_1) and (x_2, y_2) .

The primary coordinates of the trend line are defined as the *StartBar*, the *StartPrice*, the *EndBar* and the *EndPrice*.

StartBar: set the value of the *StartBar* parameter to the bar (or candle) where the trend line should begin.

NOTE: On a MetaTrader price chart, the currently forming candle is defined as candle 0 and the candle numbers increase to the left. Therefore, the *StartBar* is a larger number than the *EndBar*!

StartPrice: set the value of the *StartPrice* parameter to a [price value constant](#). The price is the vertical location where the line begins. The *StartPrice* is the *price value of the StartBar*.

StartOffset: set the value of the *StartOffset* parameter to a positive or negative integer value (for example:10, 20, ...). The *StartOffset* value moves the vertical start location up or down relative to the *StartPrice*.

EndBar: set the value of the *EndBar* parameter to the bar (or candle) where the trend line should end.

NOTE: A trend line is projected to the right *indefinitely* regardless of where the *EndBar* is located. The *EndBar* simply defines the angle of the line, not its actual end point.

EndPrice: set the value of the *EndPrice* parameter to a [price value constant](#). The price is the vertical location where the trend line ends. The *EndPrice* is the *price value of the EndPrice*.

NOTE: If the *EndPrice* is set as the Close, High, or Low of bar number 0, or to Ask or Bid, then the *EndPrice* may change value on each tick.

EndOffset: set the value of the *EndOffset* parameter to a positive or negative integer value (for example:10, 20, ...). The *EndOffset* value moves the vertical end location up or down relative to the *EndOffset*.

LineColor: optionally set the value of the *LineColor* parameter to a Color. The default is Black.

DrawLabel: optionally set the value of the *DrawLabel* parameter to a true. (The default is false.) This will draw a label with the name of the trend line on the price chart. The label can be removed programatically using [fnDeleteAllLabels](#), or manually from from the [MetaTrader tool bar](#).

DrawMarks: optionally set the value of the *DrawMarks* parameter to a true. (The default is false.) This will draw a vertical lines through the start and end bars. The vertical lines can be removed programatically using [fnDeleteAllVerticalLines](#), or manually from from the [MetaTrader tool bar](#).



Signal Aggregator Plug-in

Requires VTS-Connect minimum version 4.0.0.54

The [Signal Aggregator Plug-in](#) allows a large group of trading signals to be evaluated using Fuzzy Logic by assigning a custom weight to each trade signal.

What is a Plug-in?

VTS stands for *Visual Traders Studio*.

The VTS *Expert Advisor Builder* is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A *VTS Plug-in* allows traders to easily implement advanced trading techniques using an add-on user interface.



Enabled the Signal Aggregator Plug-in

You must enter your License key to enable the [Signal Aggregator Plug-in](#). Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

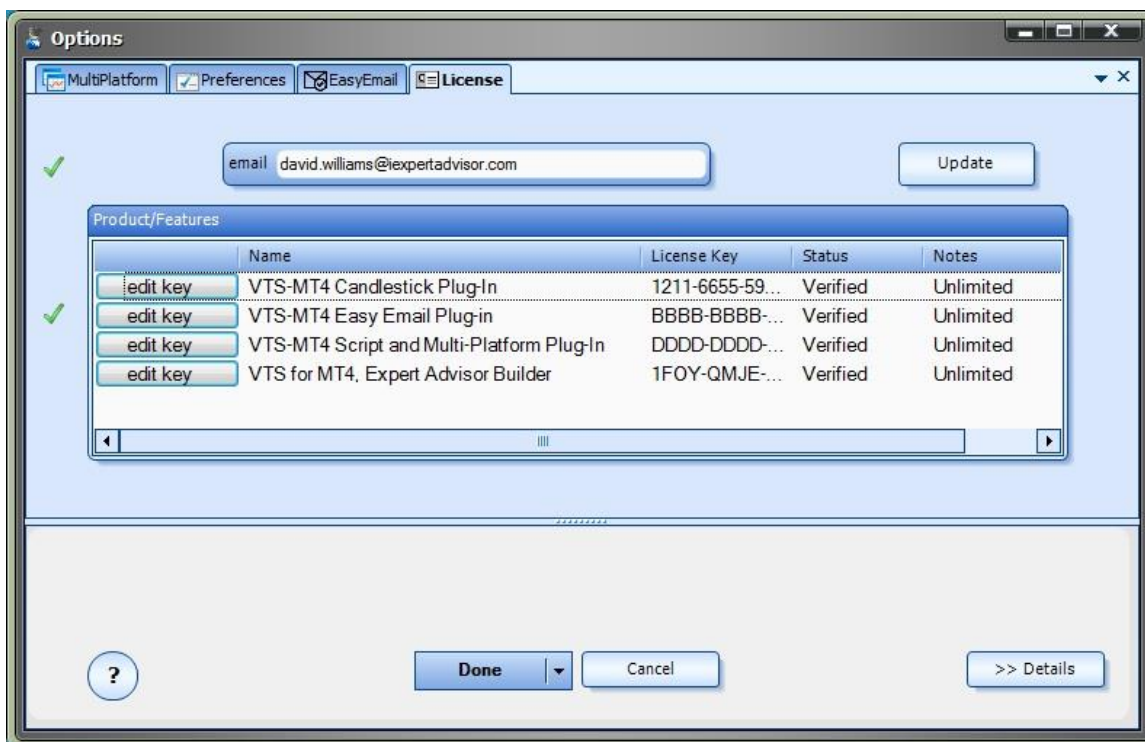
The email address is the email address used to purchase [VTS](#).

The License Key is the key listed in the Members Area.

The Update button is used to verify the email address and license key.

The edit key button is used edit the key value.

NOTE: After entering the license key, VTS must be restarted to build and enable the [Signal Aggregator](#) features.



What is the Signal Aggregator?

Most trading systems evaluate a *trading signal* and execute actions if the signal is *true* or *false*.

Some trading systems will use multiple signals. Generally, all of the signals must evaluate to *true* for an action to be executed.

A *trade signal* is any logic derived from the market that can be evaluate to *true* or *false*. A *trade signal* is usually based on technical analysis: indicators values, prices, candle-stick patterns, etc.

An example of a simple *trade signal* is:

RSI > 75.0

This *trade signals* reads "RSI greater than 75.0". If the current RSI value is greater than 75.0, then this *trade signal* will evaluate to *true*.

This kind of signal can be very powerful and is the basis for virtually all automated trading systems.

The *Signal Aggregator* brings technical analysis to another level. It combines an unlimited number of trade signals and applies an individual weight to each signal.

The *Signal Aggregator* works likes this:

Trade Signals are added to the *Signal Aggregator*.

Each Signal returns a value of *true* or *false*.

Each Signal is assigned a weight.

If a Signal evaluates to *true*, its weight is added to the *Signal Aggregator's* total weight.

If the *Signal Aggregator's* total weight is greater than a threshold value, a trading action is executed.

This aggregation allows some signals to be assigned a greater or lesser weight than other signals. The result is that a single *trade signal* does not control the entire trade action.

This is an example of six *trade signals*, each with its own assigned weight:

Trade Signal	Weight	Notes
ADX > 35.0	20	ADX is greater than 35.0
Close[1] > Close[2]	20	The Close price of the previous candle is greater than the Close price before the previous candle
RSI < 75.0	20	The RSI is less than 75
Hour > 5 and Hour < 11	10	The time is between 5 and 11 o'clock
High[1] > iHighest(24,...)	25	The current High price is greater than the highest high of the last 24 candles
Daily ATR > Weekly ATR	5	The daily ATR is greater than the weekly ATR
Total*	100	Total weight of all signals

*The sum of all weights is 100.

For this *Signal Aggregator* a *threshold* value is chosen, for example: threshold = 75.

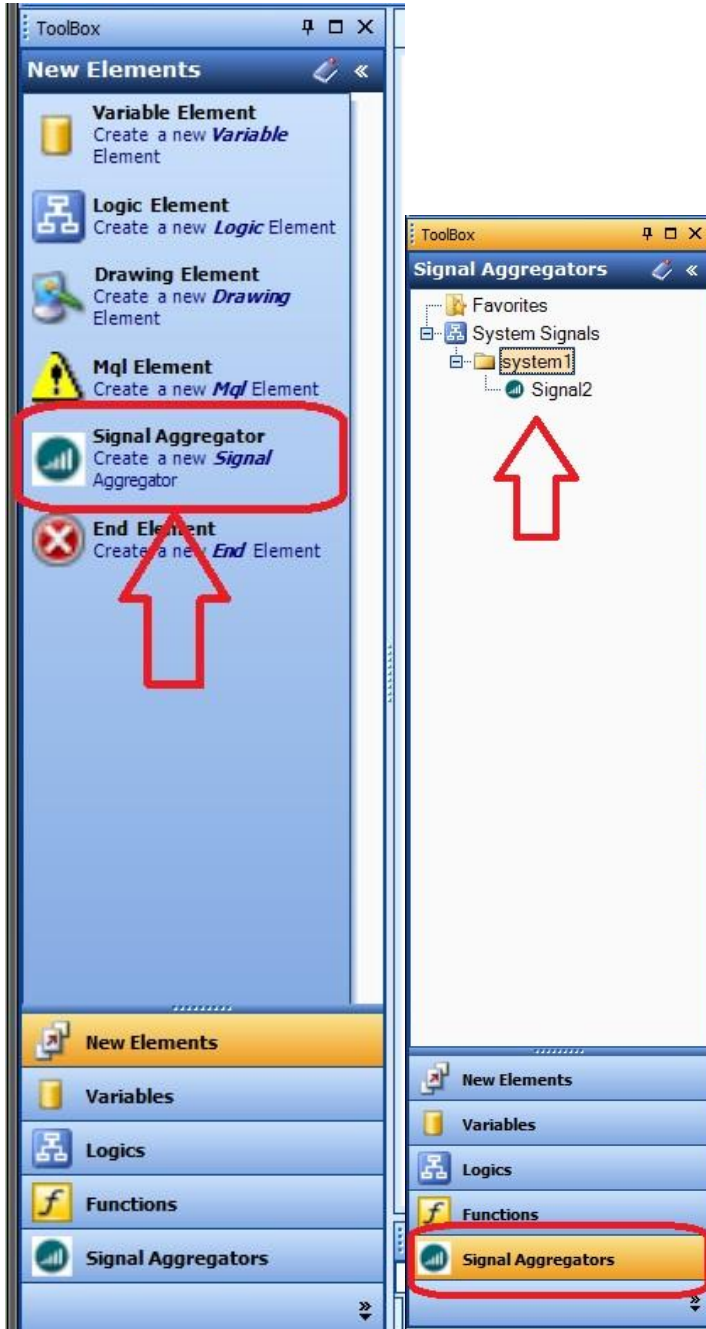
If a Trade Signal evaluates to *true*, its weight is added to the *Signal Aggregator's* total weight.

If the *Signal Aggregator's* total weight is greater than 75, a trade action is executed.

There are several combinations of *trueTrade Signals* that can lead to a threshold value greater than 75, but no one signal can drive the trade action.

Signal Aggregator in the Toolbox

The [Signal Aggregator](#) appears in the [Toolbox](#) in the [New Elements](#) pane and in its own pane for previously defined [Signal Aggregators](#).



Creating a Signal Aggregator Element

To begin creating a [Signal Aggregator](#), drag a *Signal Aggregator* from the [Toolbox](#) onto the [Drawing Pad](#) of an open VTS system.

The *Signal Aggregator* instruction screen is shown:



The following [Elements](#) can be dragged onto the *Signal Aggregator*:

From the *New Element* pane: [Logic Element](#)

From the *New Element* pane: [MQL Element](#)

From the *Logics* pane: any previously defined Logic Element

From the *Functions* pane: Any function that returns a value of *true* or *false*.

Signal Type: Element

Adding a Signal Type Element to a *Signal Aggregator*

A signal type Element is a previously defined [Logic](#) or [Function](#) Element.

Any Logic Element can be added.

Any Function Element that returns true or false can be added.

Note: When a [Function Drawing](#) is added to a *Signal Aggregator*, the [Function Drawing's](#) Element must be present and connected on the same drawing as the *Signal Aggregator*.

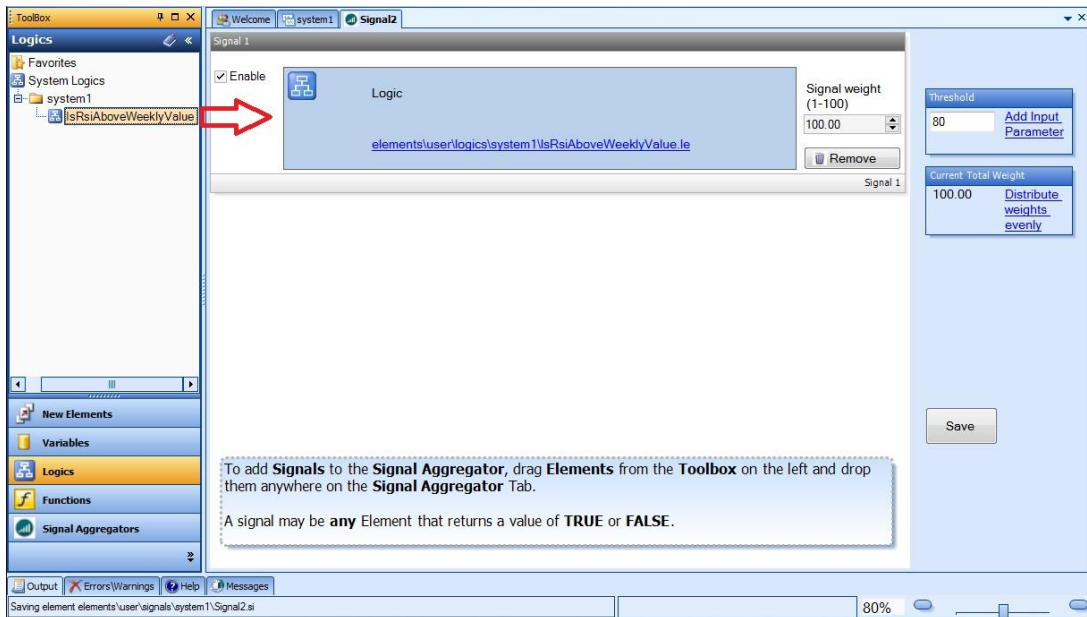
For example, follow these steps to add the [CandleStick Pattern](#) Drawing Function *Black_Body* to the *Signal Aggregator*:

Drag, drop and connect the *Black_Body* Element from the Toolbox CandleSticks menu onto the Drawing Pad in front of the *Signal Aggregator* Element.

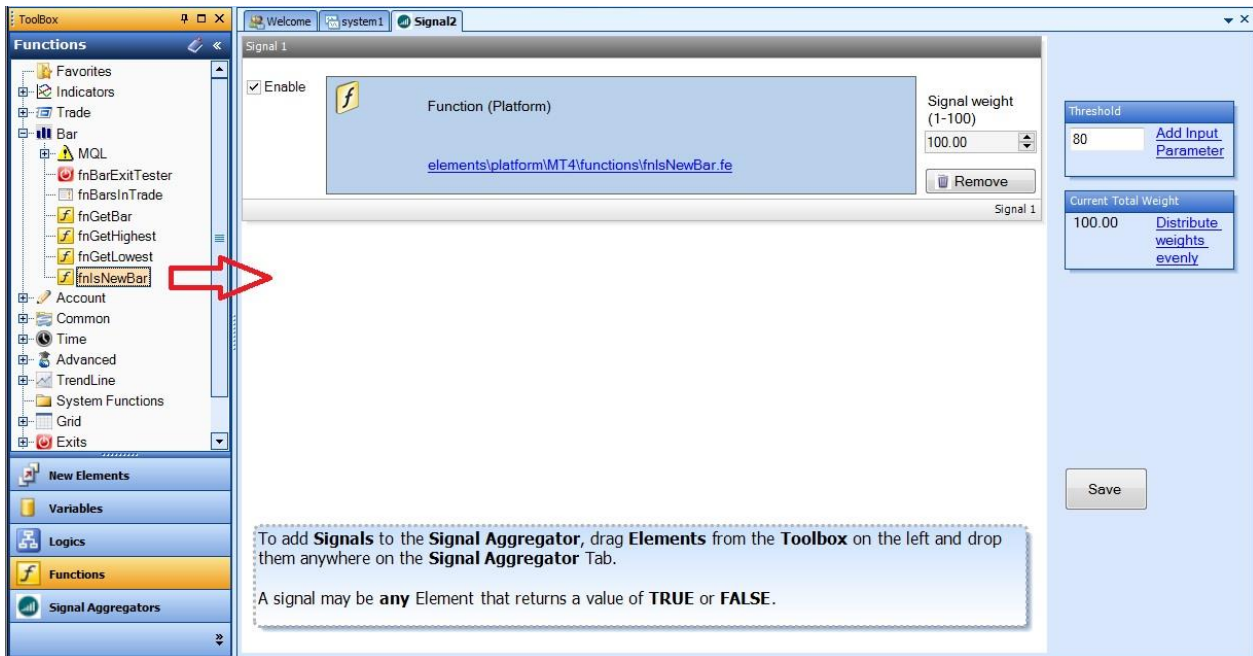
Configure and save the *Black_BodyCandleStick Pattern* function, for example save it as "*Black_Body1*".

Drag the "*Black_Body1*" Element from the Toolbox System Functions menu onto the *Signal Aggregator*.

This image shows a user-define Logic named *IsRsiAboveWeeklyValue* added to the *Signal Aggregator*:



This image shows the VTS built-in Function *fnIsNewBar* added to the Signal Aggregator:



This image shows the MetaTrader Platform Function *IsDemo* added to the Signal Aggregator:



Signal Type: MQL

Adding a Signal Type MQL to a *Signal Aggregator*

A signal type MQL is used to add native MQL code to a Signal Aggregator.

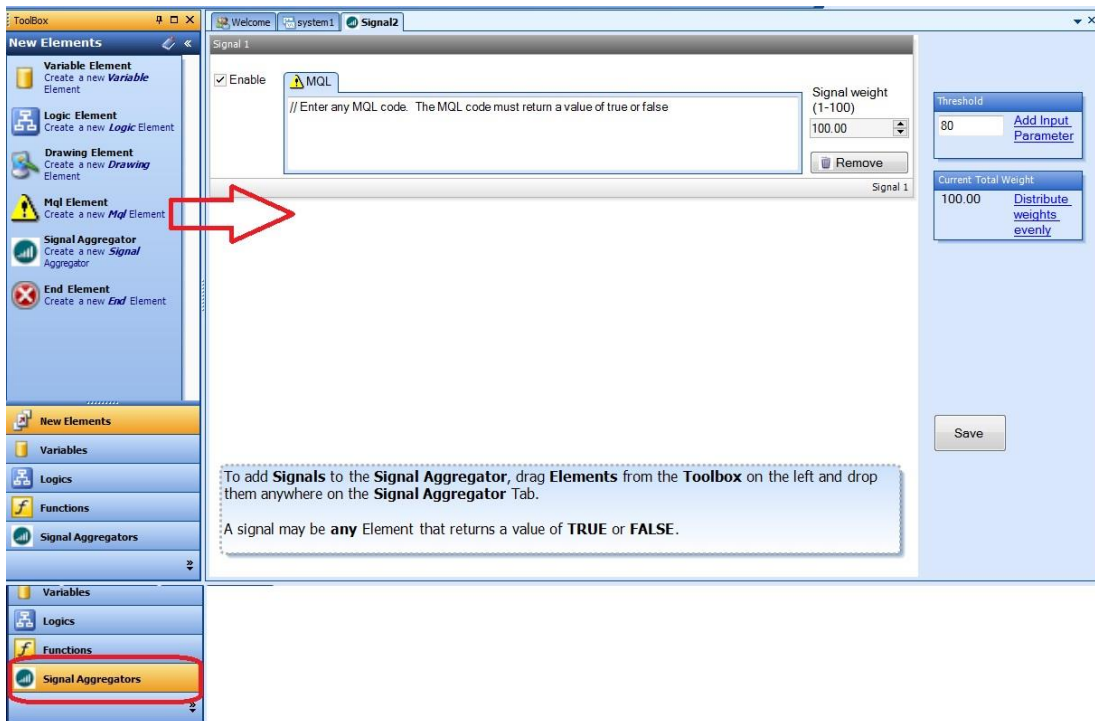
The MQL code must be an expression that can be evaluated to *true* or *false*. For example:

Ask > Bid

Close < Open

Warning: Invalid MQL code added to the MQL Element will prevent the Expert Advisor from building. Use this feature carefully.

This image shows a MQL Element added to a *Signal Aggregator*:

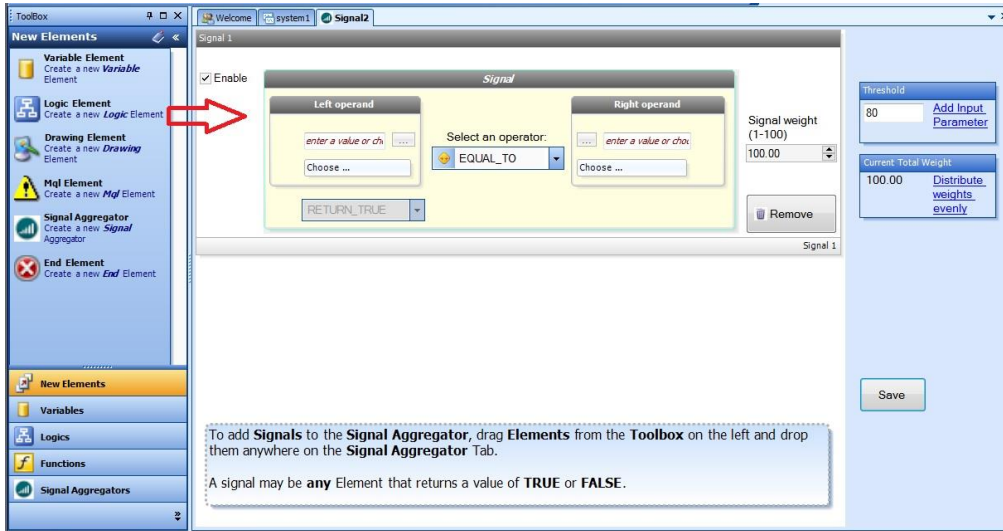


Signal Type: Logic Condition

Adding a Signal Type Logic Condition to a *Signal Aggregator*

A logical condition can be defined within a *Signal Aggregator* by dragging a New Logic Element into the *Signal Aggregator*.

This image shows a Logic Condition added to a *Signal Aggregator*:

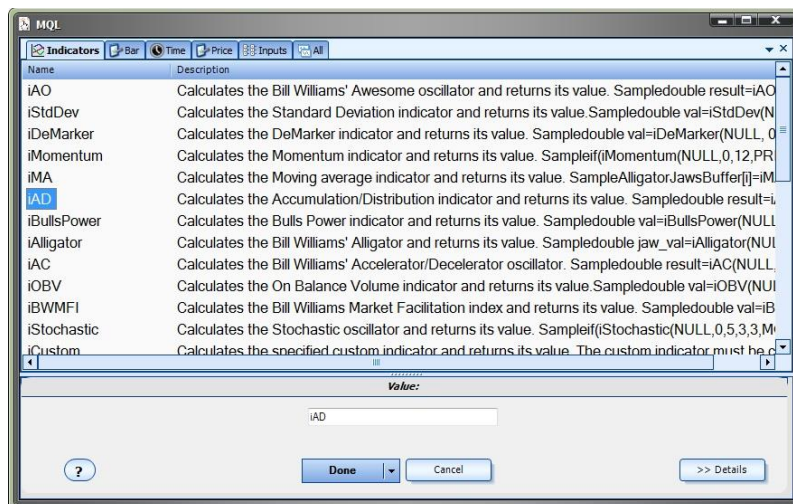


The logic condition of the *Signal Aggregator* is configured similar to how a condition is configured in a Logic Element, except:

A logic condition of a *Signal Aggregator* always returns *true*.

A logic condition of the *Signal Aggregator* is limited to a single condition.

MetaTrader Indicator functions can be directly added to the logical condition by clicking the *Choose* button and selecting the *Indicator* tab:



Configuring the *Weight* and *Threshold*

After adding Signals to the [Signal Aggregator](#), the *Signal weight* and *Threshold* values are adjusted.

The *Threshold* is the value that the sum of the active signals must exceed for the *Signal Aggregator* to generate a value of *true*.

For example, if the *Threshold* value is 80.0, when the sum of the *Signal weights* of the Signals that return true exceeds 80.0, the *Signal Aggregator* will return a value of *true*.

The *Threshold* value can be defined as an input parameter to the EA by clicking the link *Add Input Parameter*.

The *Signal weight* is defined for each Signal.

The *Signal weight* is a value between 1-100.

The sum of all active *Signal weights* should be equal to 100.

Click the link *Distribute weights evenly* to assign the same weight to all active Signals.

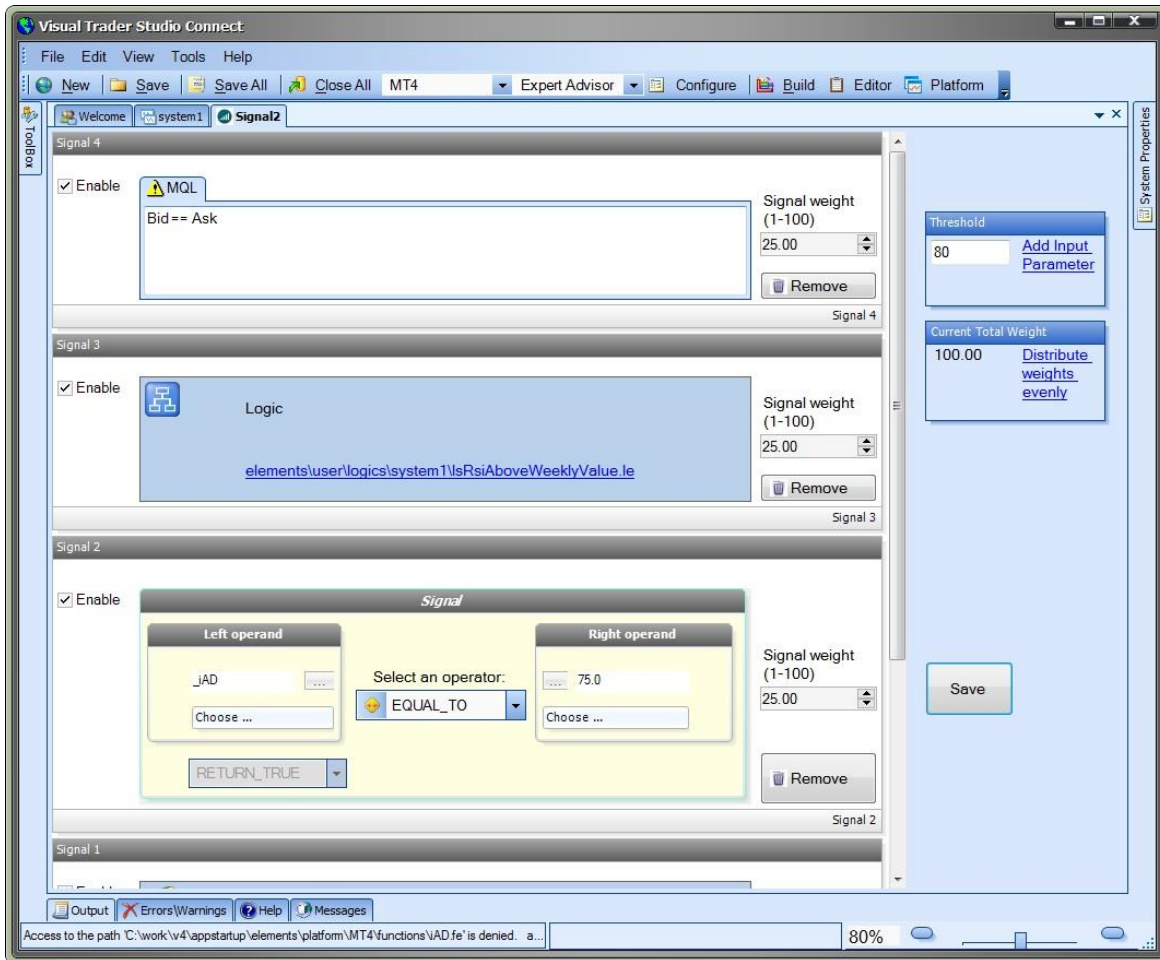
The *Current Total Weight* is displayed. This is a read-only value that calculates the current sum of all active Signal's weights.

The Save button is used to save the *Signal Aggregator*.

Each Signal has an *Enable* button. When the *Enable* button is unchecked:

The Signal's weight is not added to the *Current Total Weight*.

The Signal is not included in the generated MQL code.



Using a *Signal Aggregator* on a Drawing

After a [Signal Aggregator](#) has been saved it is available in the [Toolbox](#) to be dragged onto the [Drawing Pad](#).

A [Signal Aggregator](#) is used on a Drawing similar to a [Logic](#) Element.

A *Signal Aggregator* has a single input.

A *Signal Aggregator* has two outputs: *true* and *false*.

Execution will follow the [true link](#) when the sum of the active signals exceeds the *Threshold* value.

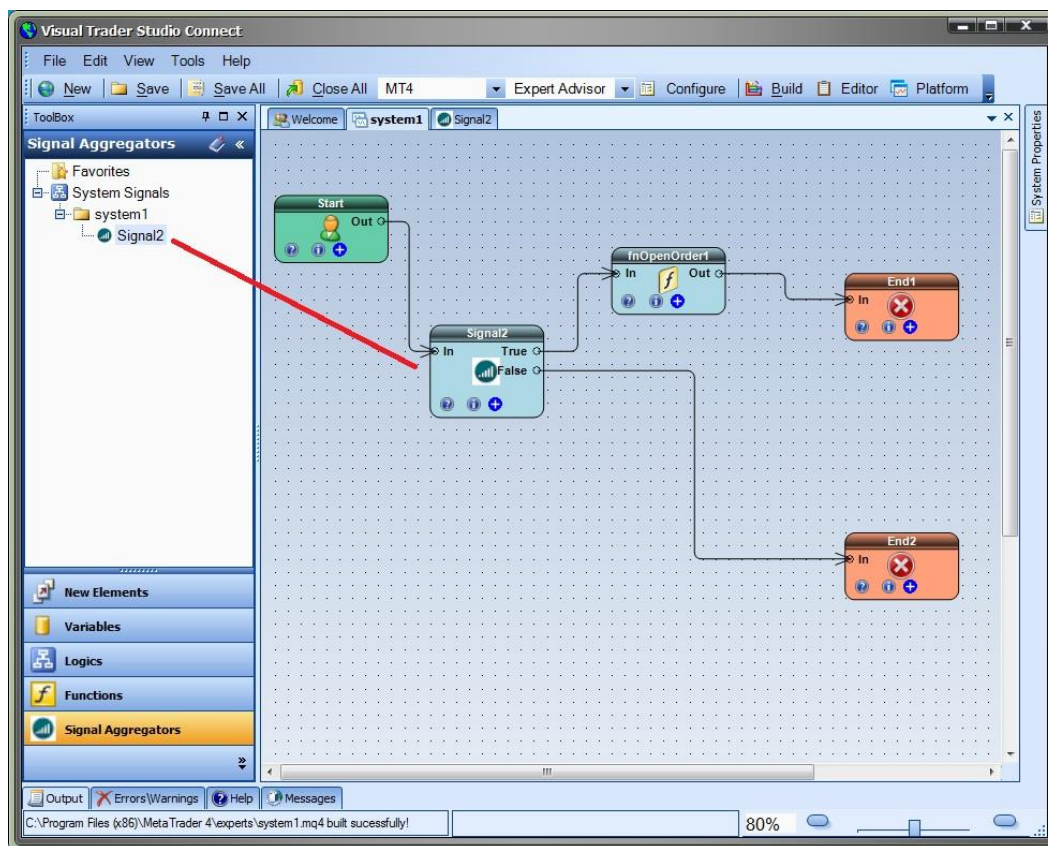
Execution will follow the [false link](#) when the sum of the active signals is equal to or below the *Threshold* value.

A [Signal Aggregator](#) is opened for configuration:

Clicking the (+) on the *Signal Aggregator* Element

Right-clicking the *Signal Aggregator* in the [Toolbox](#) and selecting *Configure*.

In the Drawing below, when the sum of the active signals exceeds the *Threshold* value of the *Signal2Signal Aggregator*, execution will follow the *true* output, and the [fnOpenOrder1](#) function is called to open a trade.



MQL-Mentor Plug-in

Requires VTS-Connect minimum version 4.0.0.56

The MQL-Mentor Plug-in allows you to use VTS to learn MQL syntax.

The MQL-Mentor Plug-in features:

MQL code can be viewed per Element

MQL syntax can be checked

MQL code can be added to an MQL file without being overwritten by the VTS code generator

What is a Plug-in?

VTS stands for *Visual Traders Studio*.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A *VTS Plug-in* allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the MQL-Mentor Plug-in

You must enter your License key to enable the *MQL-Mentor Plug-in*. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The email address is the email address used to purchase [VTS](#).

The License Key is the key listed in the Members Area.

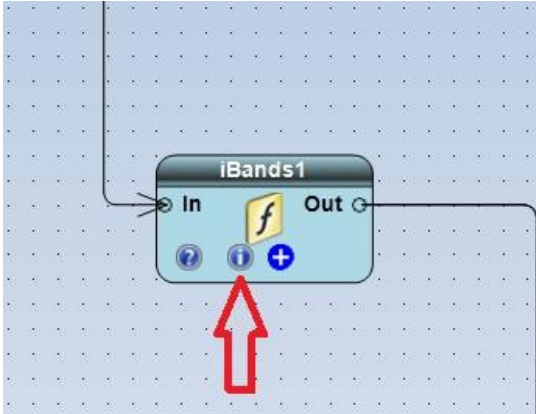
The Update button is used to verify the email address and license key.

The edit key button is used edit the key value.

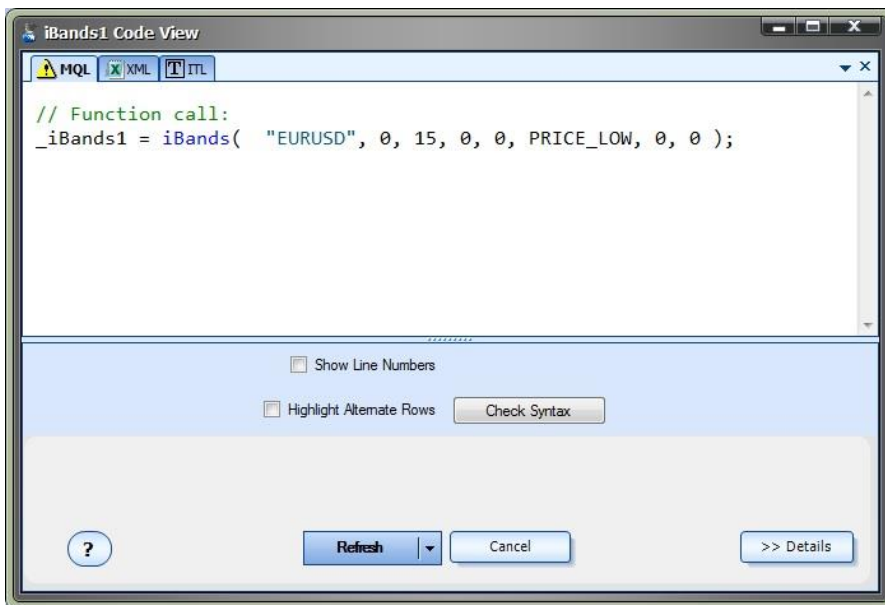


View an Element's MQL Code

To view an Element's MQL code click the (i) button on the [Element](#):



Clicking the (i) button will open the information window:



The latest generated MQL code for the Element is displayed in the MQL Tab.

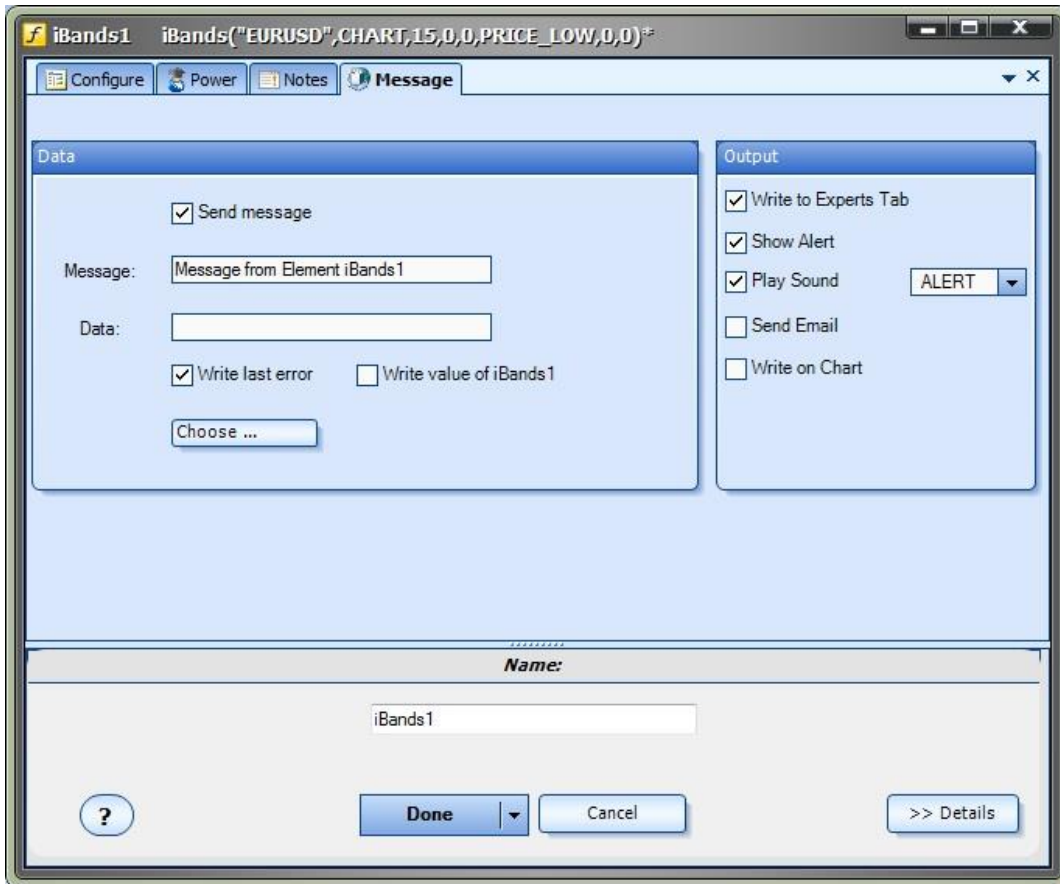
The number of each line may be displayed by checking the *Show Line Numbers* checkbox

The color of alternate rows may be displayed by checking the *Highlight Alternate Rows* checkbox

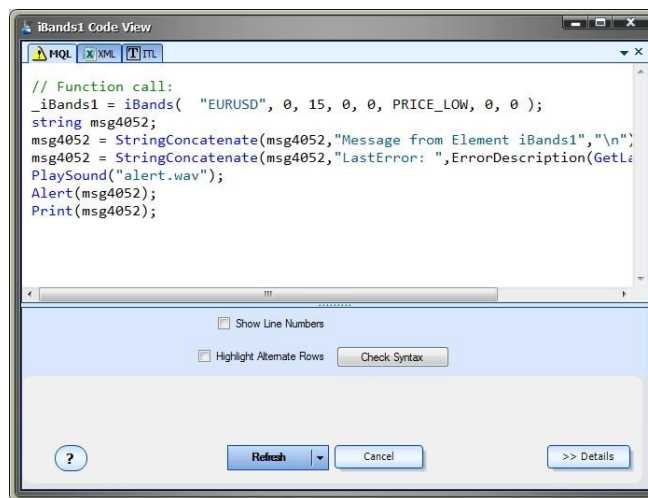
The syntax of the MQL may be checked by clicking the *Check Syntax* button

Changes made to the configuration of the Element will result in re-generated MQL code.

For example, if the [Message](#) tab of the Element is configured as below:



The MQL is re-generated as follows:



Note the additional MQL code to construct the message and the calls to the MQL functions *PlaySound*, *Alert* and *Print*.

Using the Code Block Feature

The Code Block features allows you to create a code block within an MQL file that will not be overwritten by the VTS code generator.

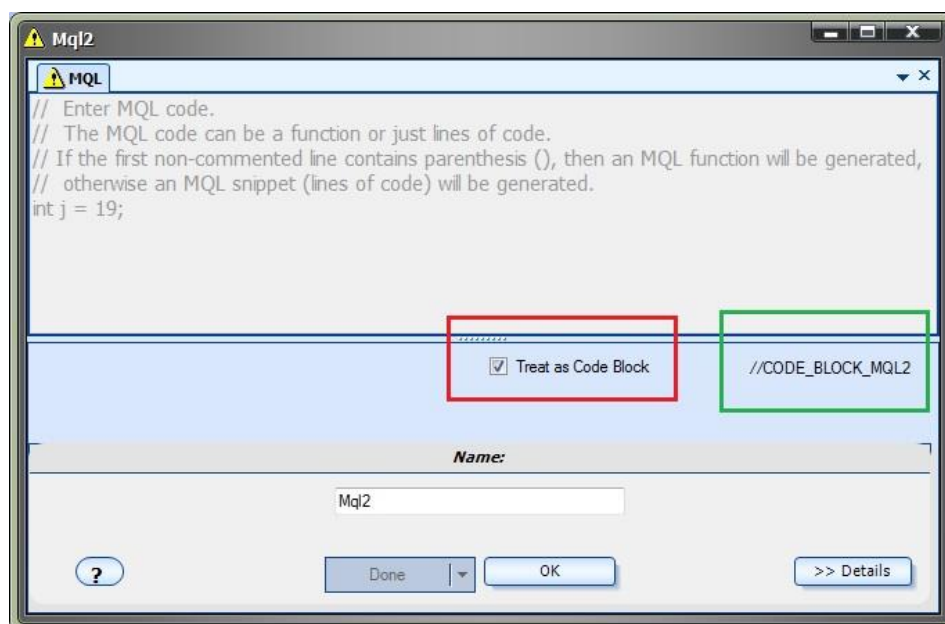
This allows you to make manual edits to your MQL code using the [MetaEditor](#) without your code be overwritten when the system is re-built inside of VTS.

To create a Code Block:

Drag a new [MQL Element](#) from the [ToolBox](#):



Configure the new MQL Element by clicking the (i) button:



Check the *Treat as Code Block* checkbox. this will:

Set the editor to read-only so the MQL code can not be changed inside of VTS

Show the tag used to find and identify the code block inside of the MQL file

Save and Build the VTS System

Open the MetaEditor (inside of VTS by clicking the [Editor](#) button, or outside of VTS via your MetaTrader platform)

Find the *START* and *END* sections of the code block. For example:

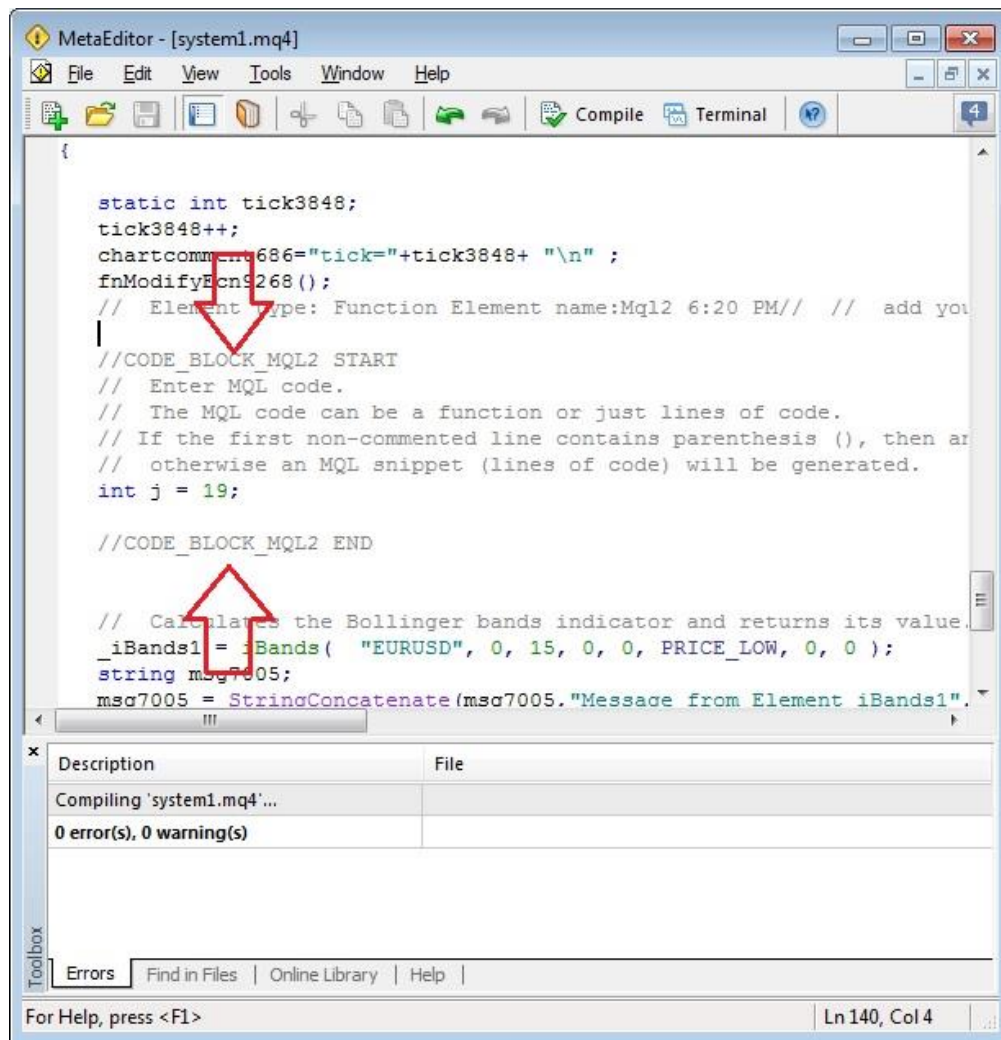
CODE_BLOCK_MQL2_START

CODE_BLOCK_MQL2_END

Any MQL code added between these tags it will not be overwritten by the VTS code generator:

VTS will search the MQL file for code block tags, save the code to the MQL Element and continue.

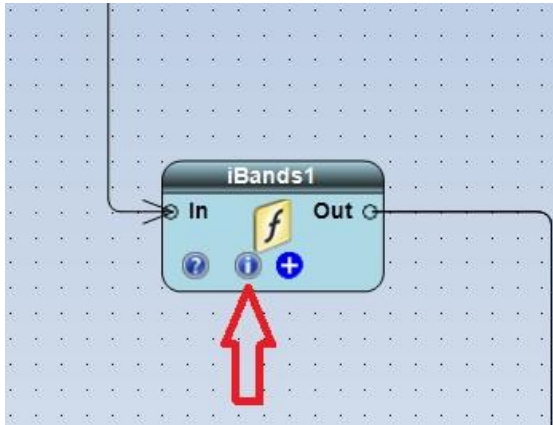
At any time the *Treat as Code Block* checkbox may be unchecked and the code from the MQL Element inside of VTS will be used instead of the code within the code block of the MQL file. Note: All changes in the MQL file will be lost.



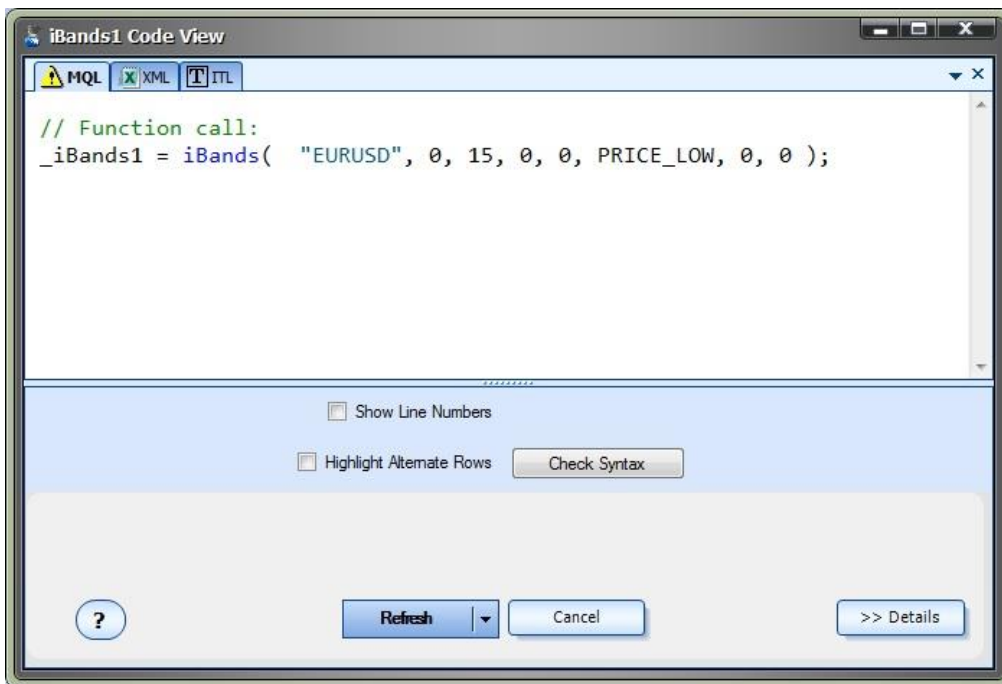
Syntax Check

The syntax of any Element's MQL can be checked by clicking the *Check Syntax* checkbox.

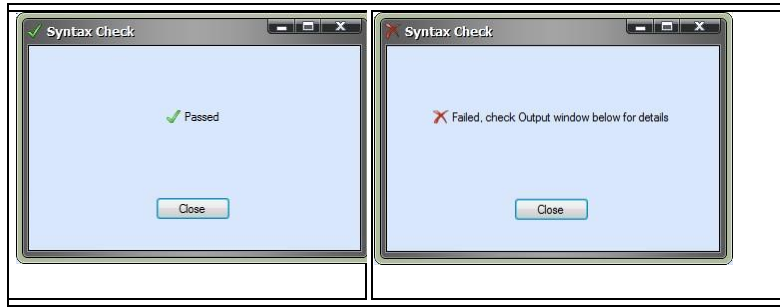
To view an Element's MQL code click the (i) button on the [Element](#):



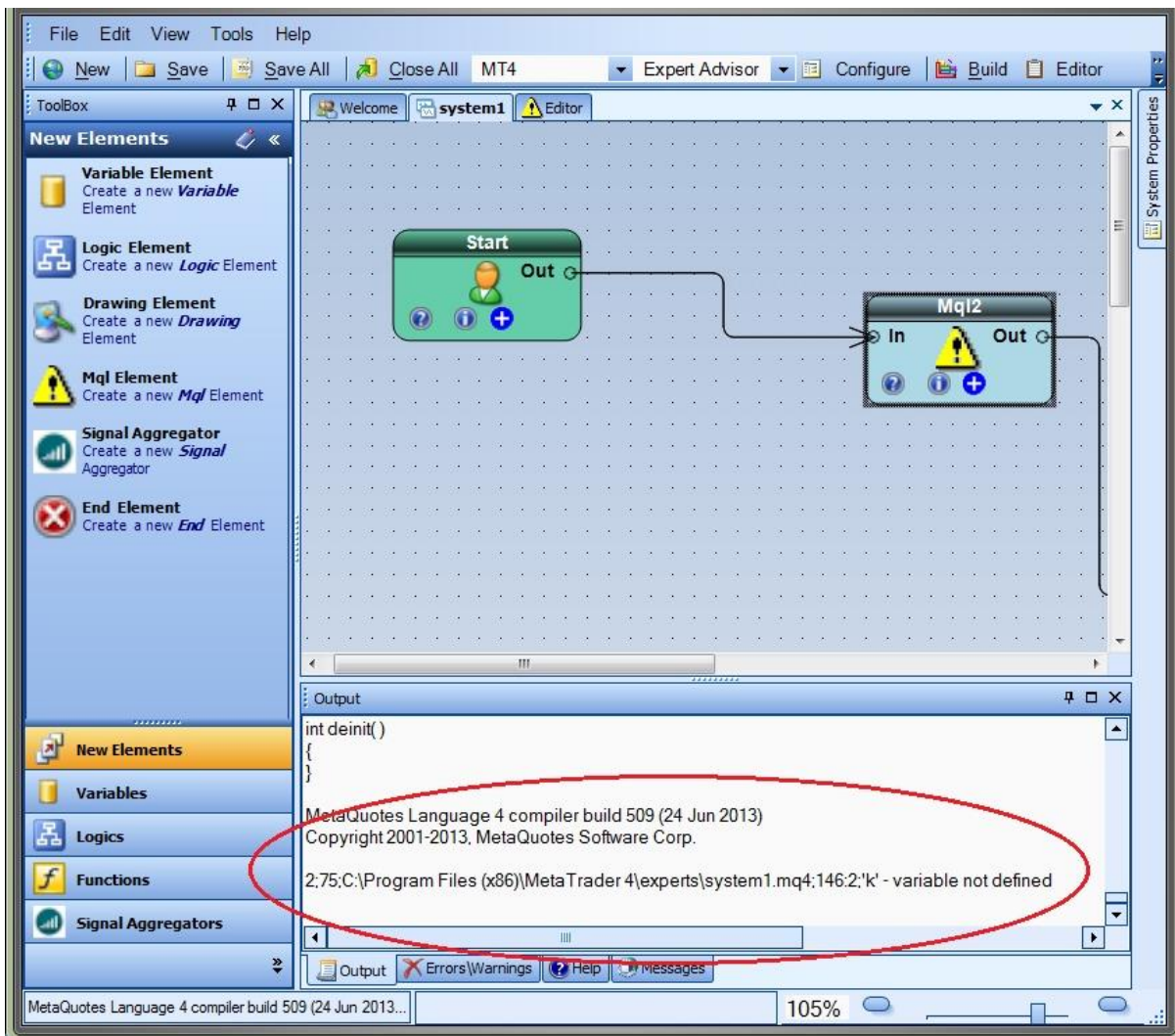
Clicking the (i) button will open the information window:



Clicking the *Check Syntax* checkbox will display a Pass or Fail window.



If the syntax check fails, expand the [Output](#) window at the bottom of VTS to see the exact reason.



Partial-Close Plug-in

Requires VTS-Connect minimum version 4.0.0.58

The Partial-Close Plug-in allows you to create an Expert Advisor that closes portions of a winning trade at specified profit levels.

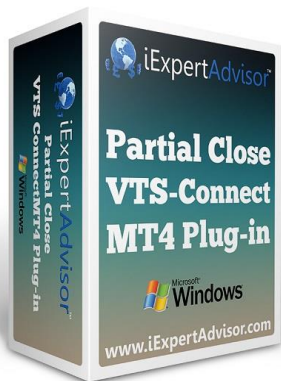
What is a Plug-in?

VTS stands for *Visual Traders Studio*.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A *VTS Plug-in* allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the Partial-Close Plug-in

You must enter your License key to enable the *Partial-Close Plug-in*. Your license key for all of your VTS products can be found in the [Members Area](#).

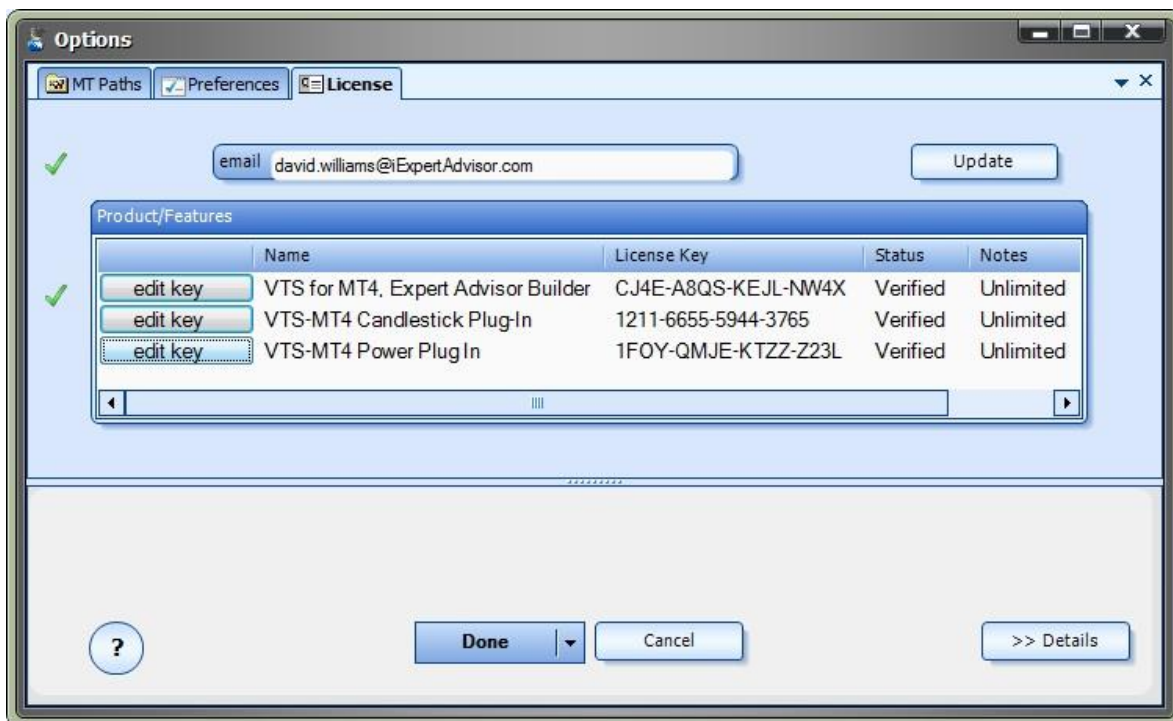
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The email address is the email address used to purchase [VTS](#).

The License Key is the key listed in the Members Area.

The Update button is used to verify the email address and license key.

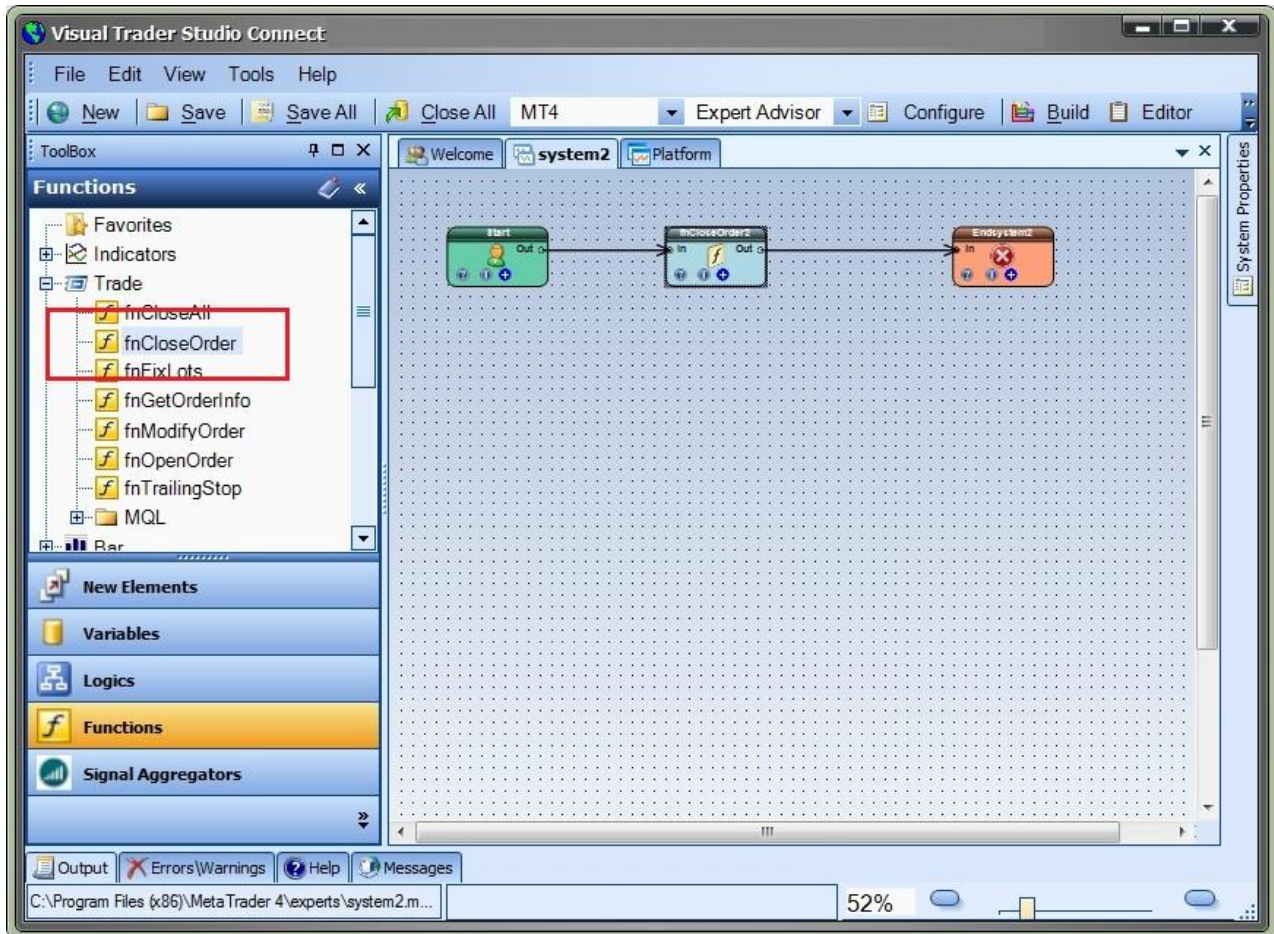
The edit key button is used edit the key value.



Partial-Close Function in the Toolbox

The only [function](#) required to implement the partial-close feature is the function [fnCloseOrder](#).

[fnCloseOrder](#) is found in the functions [ToolBox](#) under the *Trade* menu.

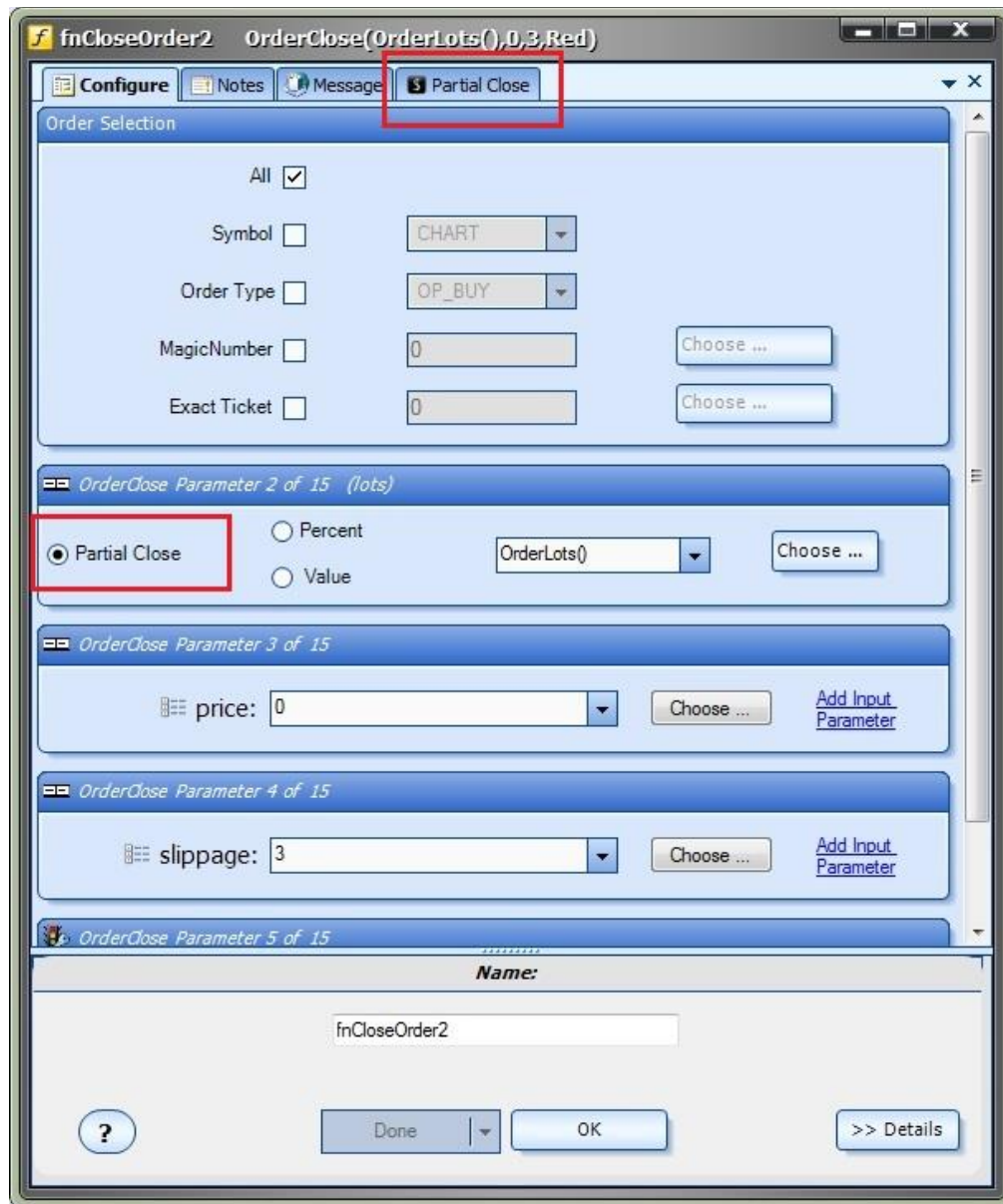


Partial-Close Access

When the *Partial-Close Plug-In* has been enabled, the radio button "Partial Close" is displayed in the *Lots* parameter of the [fnCloseOrder](#) function configuration window.

When the "Partial Close" radio button is selected, the "Partial Close" tab appears. The "Partial Close" tab allows for the configuration of partial-close levels.

To disable the Partial-Close feature, select either the *Percent* or *Value* radio button of the *Lots* parameter.



Configuring Partial-Close Levels

Partial-Close Levels are added, edited or deleted on the "Partial Close" tab of the [fnCloseOrder](#) function configuration window.

When the Partial-Close feature is first enabled, one "Partial Close Level" is automatically created.

To add a "Partial Close Level" click the *Add* button.

To remove a "Partial Close Level" click the down arrow on the *Add* button and select *Remove*.

Note: The top-most "Partial Close Level" can not be removed.

An unlimited number of "Partial Close Levels" may be added.

To configure a "Partial Close Level" enter values for the *PIP Profit* and *Lot Size* values.

The *Pip Profit* value is the numbers of pips of profit the trade needs to reach to set the lot size of the trade to *Lot Size*.

The *Lot Size* is the number of lots that will remain open when the *Pip Profit* value has been reached.

The example in the image below shows these values:

Pip Profit	Lot Size	Notes
25	4.0	When the trade reaches 25 PIPs of profit, the number of open lots is modified to 4.0. This trade was opened with a lot size of 5, therefore 1 lot is closed to set the Lot Size to 4.0.
50	3.0	When the trade reaches 50 PIPs of profit, the number open lots is modified to 3.0.
75	0	When the trade reaches 75 PIPs of profit, the number open lots is modified to 0. The trade is closed. The last level may close all remaining lots or leave some portion of the trade open indefinitely, to be closed by a trailing or fixed stop, etc.

Pip Profit is the difference in points between the current market price and trade's open price. It does not account for the number of lots open or the profit in dollars (or base currency).

If a level has ever been reached the lot size is modified. Therefore, it is possible the trade's lot size may be 3.0 lots with a profit of less than 50 PIPs. This is because the trade at one point was 50 PIPs in profit.

NOTE: The levels will always be arranged in the order shown above:

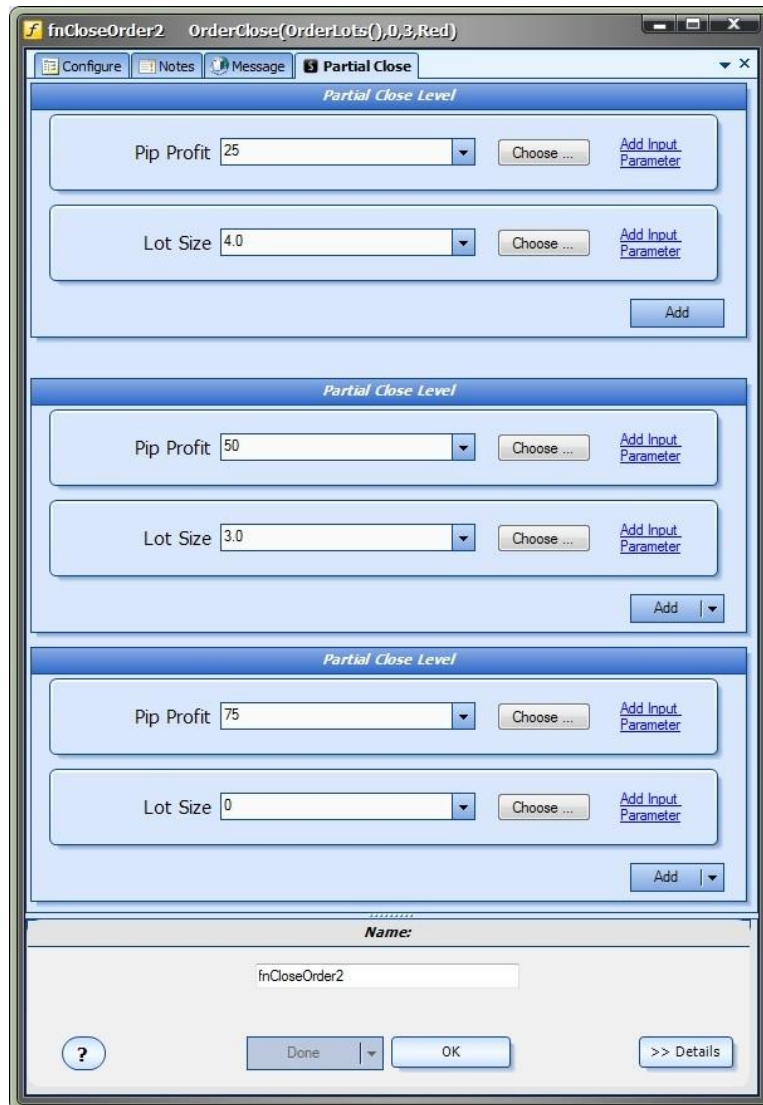
Pip Profit from smallest to largest and Lot Size from largest to smallest.

This sorting does not occur in VTS. It is implemented in the generated MQL code because the level values may be variables that can not be sorted until their value is known.

The Partial-Close feature only allows Lot sizes to become smaller as Profit increases.

The Choose button is used to set the Lot Size or Pip Profit values to any defined variable.

The "Add Input Parameter" link is used to create an input parameter for the Lot Size or Pip Profit values.



Partial-Close Price Chart

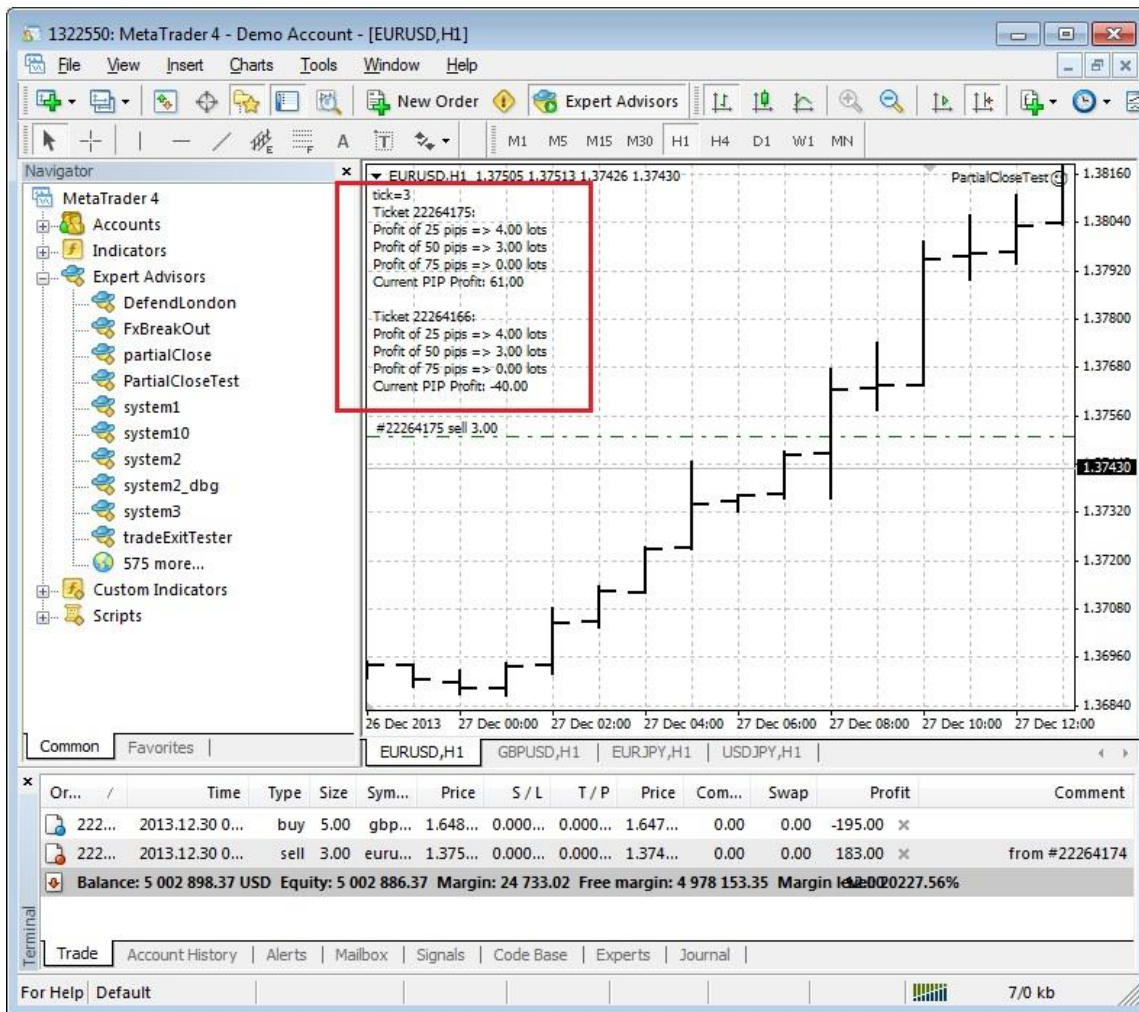
The price chart below shows an Expert Advisor running with three *Partial-Close* levels defined. There are two open trades.

For each open trade:

The ticket number is displayed.

The configuration is displayed showing the number of lots to be left remaining open at each profit level.

The current [PIP](#) profit for the ticket is displayed.



EA-Indicator Plug-in

Requires VTS-Connect minimum version 4.0.0.60

The EA-Indicator Plug-in allows you to create a Custom Indicator that draws Buy and Sell lines based on the logic of your Expert Advisor. The Custom Indicator can then be used by any Expert Advisor.

What is a Plug-in?

VTS stands for *Visual Traders Studio*.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A *VTS Plug-in* allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the EA-Indicator Plug-in

You must enter your License key to enable the *EA-Indicator Plug-in*. Your license key for all of your VTS products can be found in the [Members Area](#).

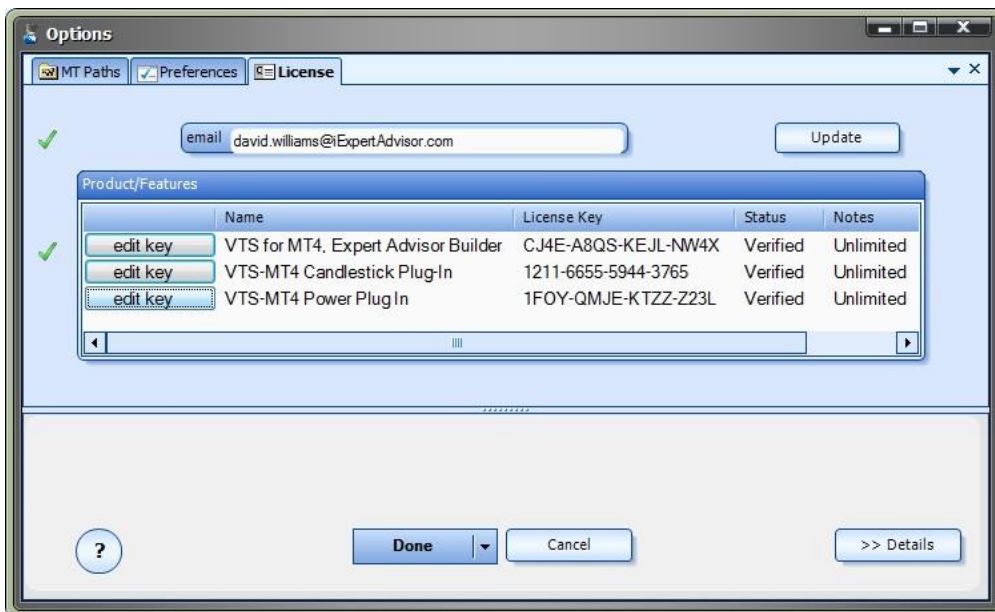
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The email address is the email address used to purchase [VTS](#).

The License Key is the key listed in the Members Area.

The Update button is used to verify the email address and license key.

The edit key button is used edit the key value.



Using the EA-Indicator Plug-in

The *EA-Indicator Plug-in* allows you to create a Custom Indicator that draws *Buy* and *Sell* lines based on the logic of your Expert Advisor.

The signals are generated from the following trading functions:

OpenBuyTrade: when true, **Blue** line is greater than 0

CloseBuyTrade: when true, **Blue** line is less than 0

OpenSellTrade: when true, **Red** line is greater than 0

CloseSellTrade: when true **Red** line is less than 0

VTS generates MQL code to build a custom indicator based on the logic of the 4 functions above.

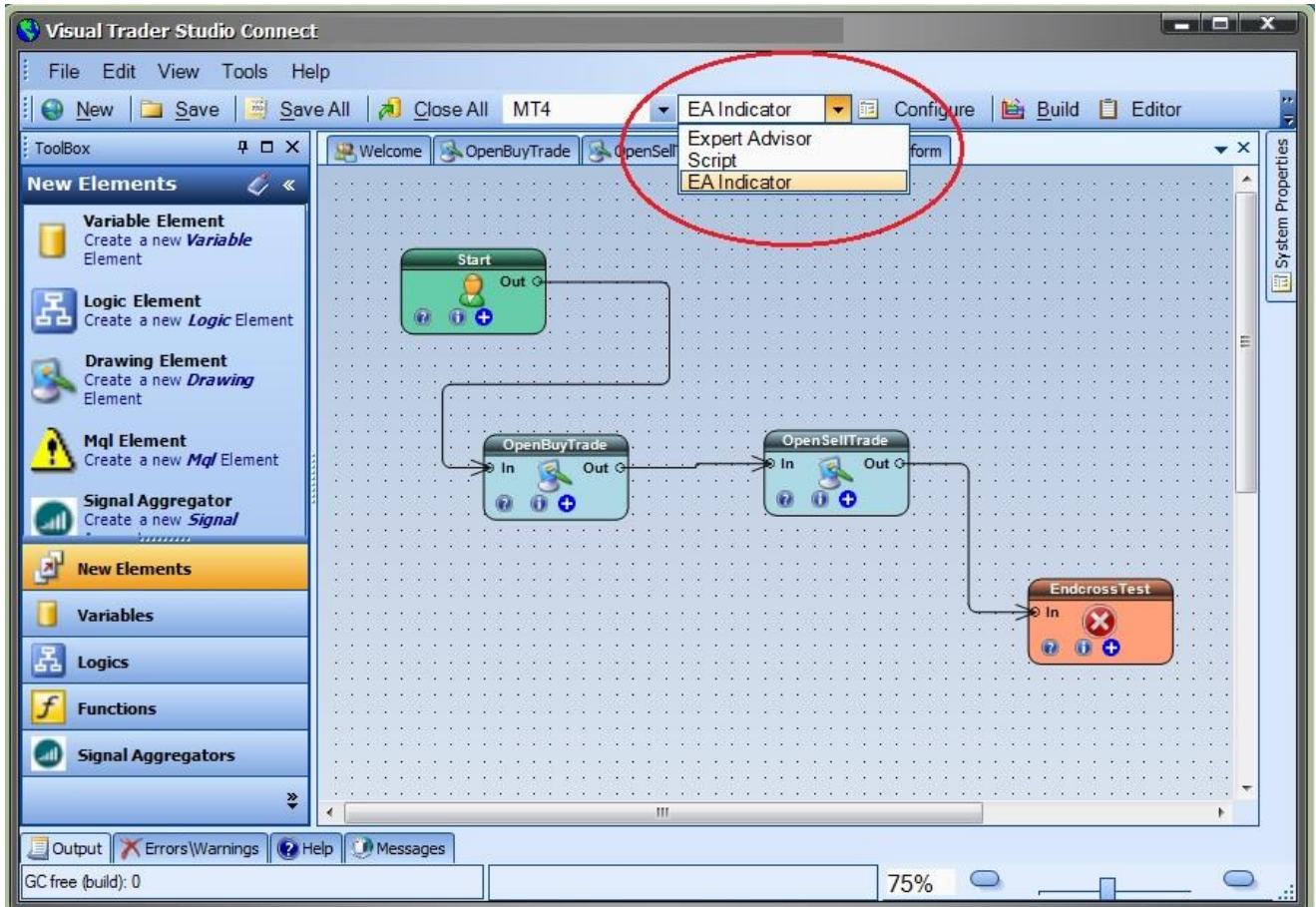
After creating an Expert Advisor in VTS that uses at least one of the functions OpenBuyTrade,

CloseBuyTrade, OpenSellTrade, and/or CloseSellTrade:

Set the VTS *target* to "EA Indicator"

Press the **Build** button

A Custom Indicator is created in the Platform folder's "custom indicator" folder (experts\indicators)

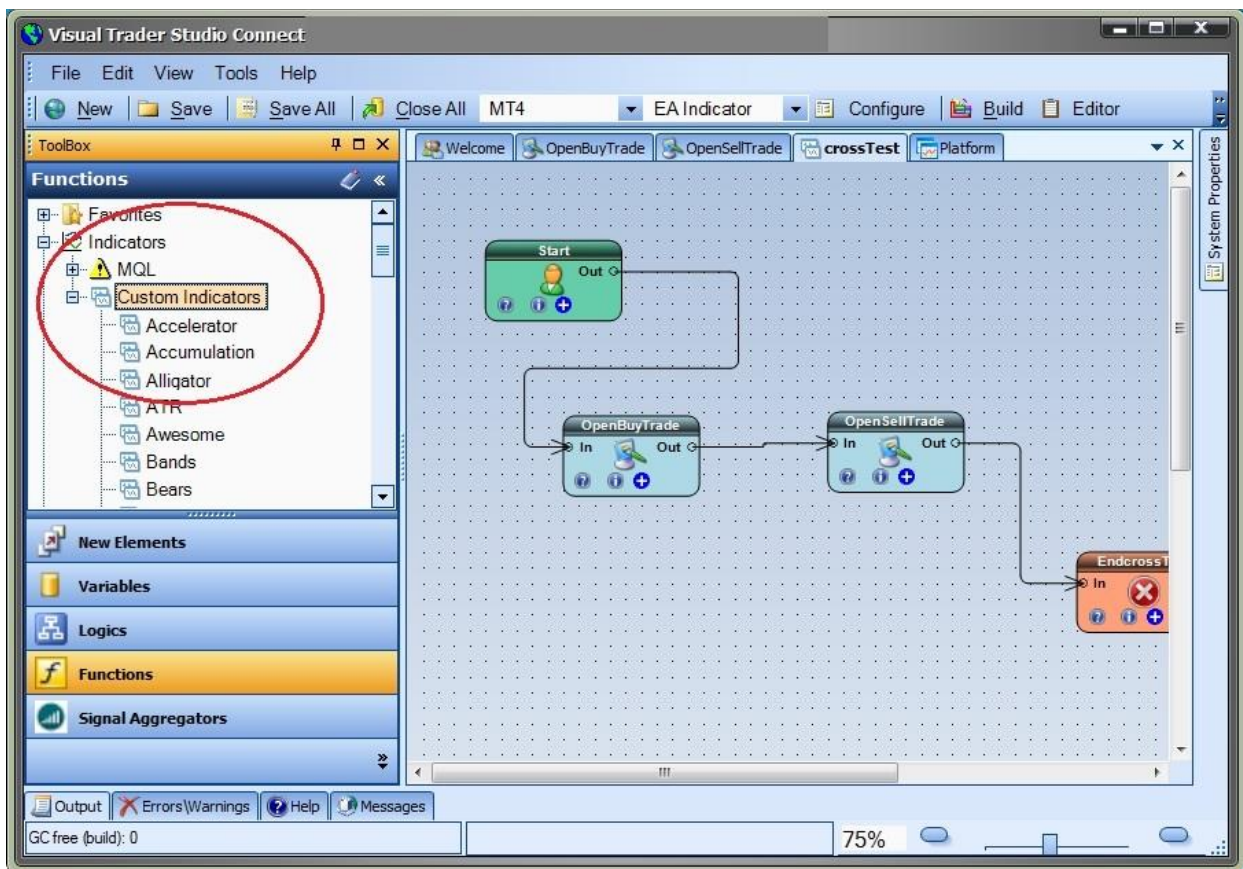


Custom Indicators in the Toolbox

After a custom indicator has been built using the *EA-Indicator plug-in*, it is available from the VTS [Toolbox](#) to be used by any Expert Advisor.

The indicator's input and output values are automatically configured using the latest version of the drawings. There is no need to configure a [custom indicator](#) that has been built using the EA-Indicator plug-in.

The "Custom Indicator" menu in the VTS [Toolbox](#) can be refreshed by right-clicking and selecting "refresh".



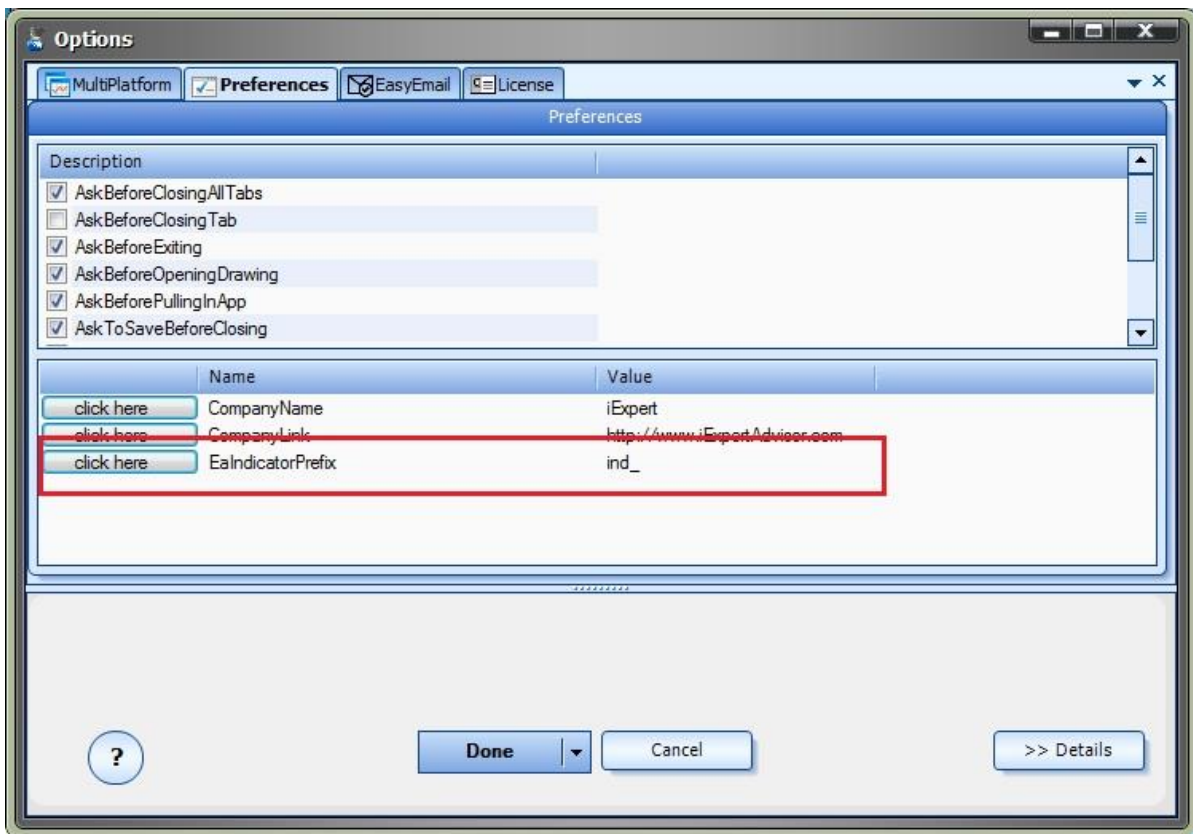
Set the Indicator Prefix

The name of the custom indicator is the name of the VTS system with a preceding prefix.

The default prefix is "ind_".

For example, if the name of the VTS System is system1, the name of the custom indicator is ind_system1.

The prefix may be changed in VTS from the [Tools->Options->Preferences](#) window. The name of the value is "EaIndicatorPrefix".



EA-Indicator Price Chart

The price chart below shows a custom indicator built from the *Moving Average Crossover System*.

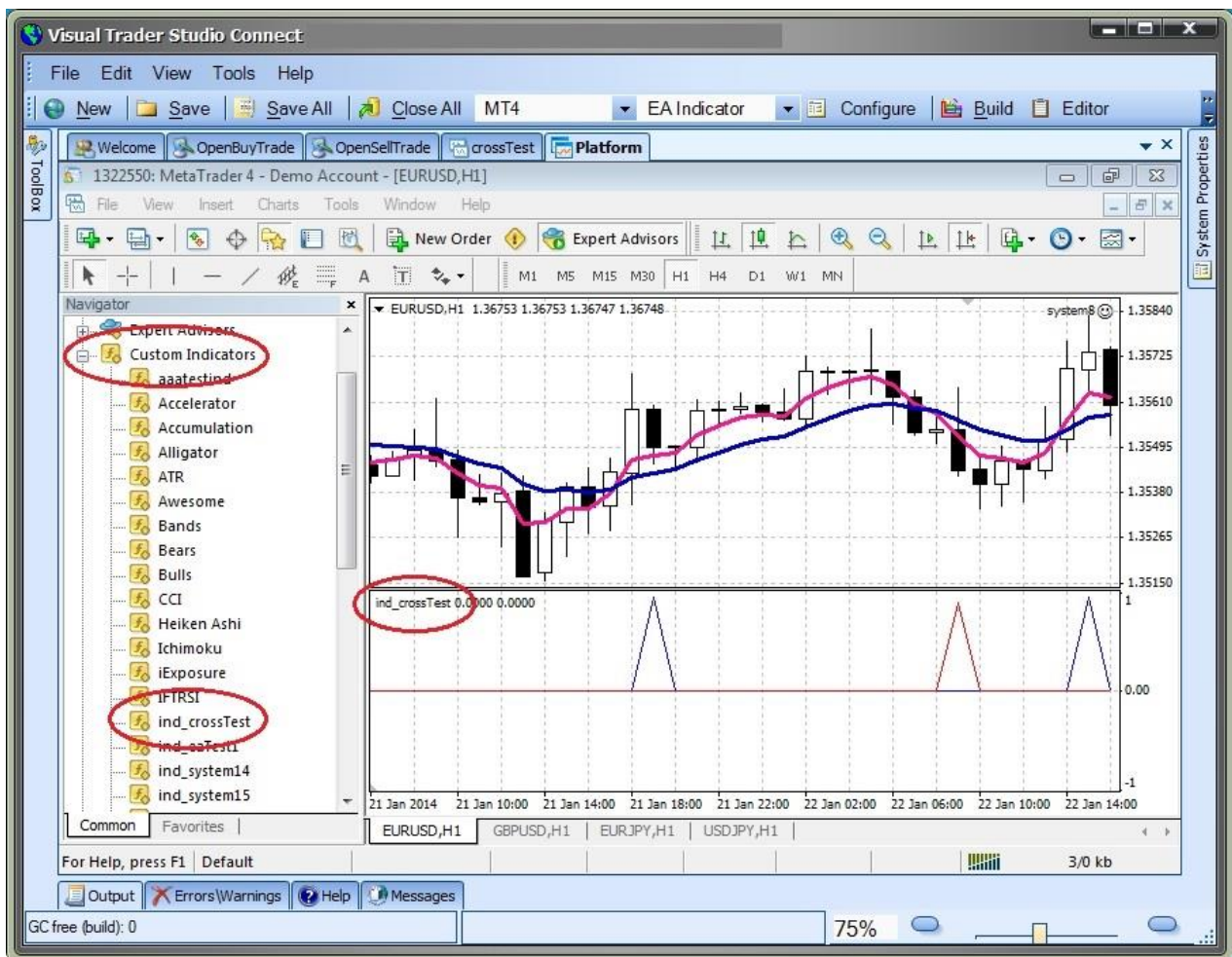
The ind_crossTest indicator is shown in the window below the price chart.

On the price chart, the thick *pink* line is a *fast* moving average, the thick *blue* line is a *slow* moving average.

When the *fast* moving average crosses the *slow* moving average trade signals are generated.

The trade signals are seen as spikes in the red and blue lines in the *lower* indicator window.

Note: The MetaTrader Platform may need to be closed and restarted before a newly created custom indicator appears in "Custom Indicators" folder of the Navigator window. This is a limitation of MetaTrader.



EA-Indicator Video

To view a video of a the EA-Indicator plug-in, click here:

<http://expert-advisor-builder.com/how-to-build-a-metatrader-moving-average-crossover-custom-indicator-using-vts/>

Important Notes:

The *EA-Indicator Plug-in* uses the four trading function drawings to create the MQL logic for the custom indicator. The drawing functions must be named: OpenBuyTrade, CloseBuyTrade, OpenSellTrade, CloseSellTrade.

In order to draw the back history for the indicator, the logic on the trading drawings must use "shift-able" expressions. The *EA-Indicator Plug-in* finds "shift-able" price and indicator functions and generates the correct MQL code to draw the history of the indicator on the chart.

If "shift-able" expressions are not used, the indicator will still display correct values in real time, but will not generate signals for historical data. For example, if the expression "Bid" is used in the logic of a trading drawing, the platform cannot evaluate the value of Bid for past bars - the Bid value simply is not saved. However, if the expression iHigh is used, the platform can accurately draw historical data because High values *are* saved for each bar.

In summary, any function that provides a [shift](#) parameter will draw the correct history. This includes all technical indicators, all price functions and various other MQL functions.

All of the drawings used in the VTS Expert Advisor system are included in the custom indicator code because they may be needed to evaluate the logic of the drawing functions. For this reason, there will be warnings for unreferenced functions. *These warnings may safely be ignored.*

This is an example of the warnings that can be safely ignored:

MetaQuotes Language 4 compiler build 509 (24 Jun 2013)

Copyright 2001-2013, MetaQuotes Software Corp.

1;39;;;Function "fnComment6194" is not referenced and will be removed from exp-file

1;39;;;Function "OpenSellTrade" is not referenced and will be removed from exp-file

1;39;;;Function "OpenBuyTrade" is not referenced and will be removed from exp-file

1;39;;;Function "ea_start" is not referenced and will be removed from exp-file

1;39;;;Function "ea_init" is not referenced and will be removed from exp-file

1;39;;;Function "ea_deinit" is not referenced and will be removed from exp-file

Exp file "C:\Program Files (x86)\MetaTrader 4\experts\indicators\ind_MaCrossOver.ex4"
produced - 0 error(s), 6 warning(s)

C:\Program Files (x86)\MetaTrader 4\experts\indicators\ind_MaCrossOver.mq4 built sucessfully!

Because all of the drawings used in the VTS Expert Advisor system are included in the custom indicator code, the custom indicator will have the same input parameters as the Expert Advisor. This may not be desired for the custom indicator. To inhibit the input parameters, used the [Input Manager](#) and uncheck the *show* button to hide any inputs not required.

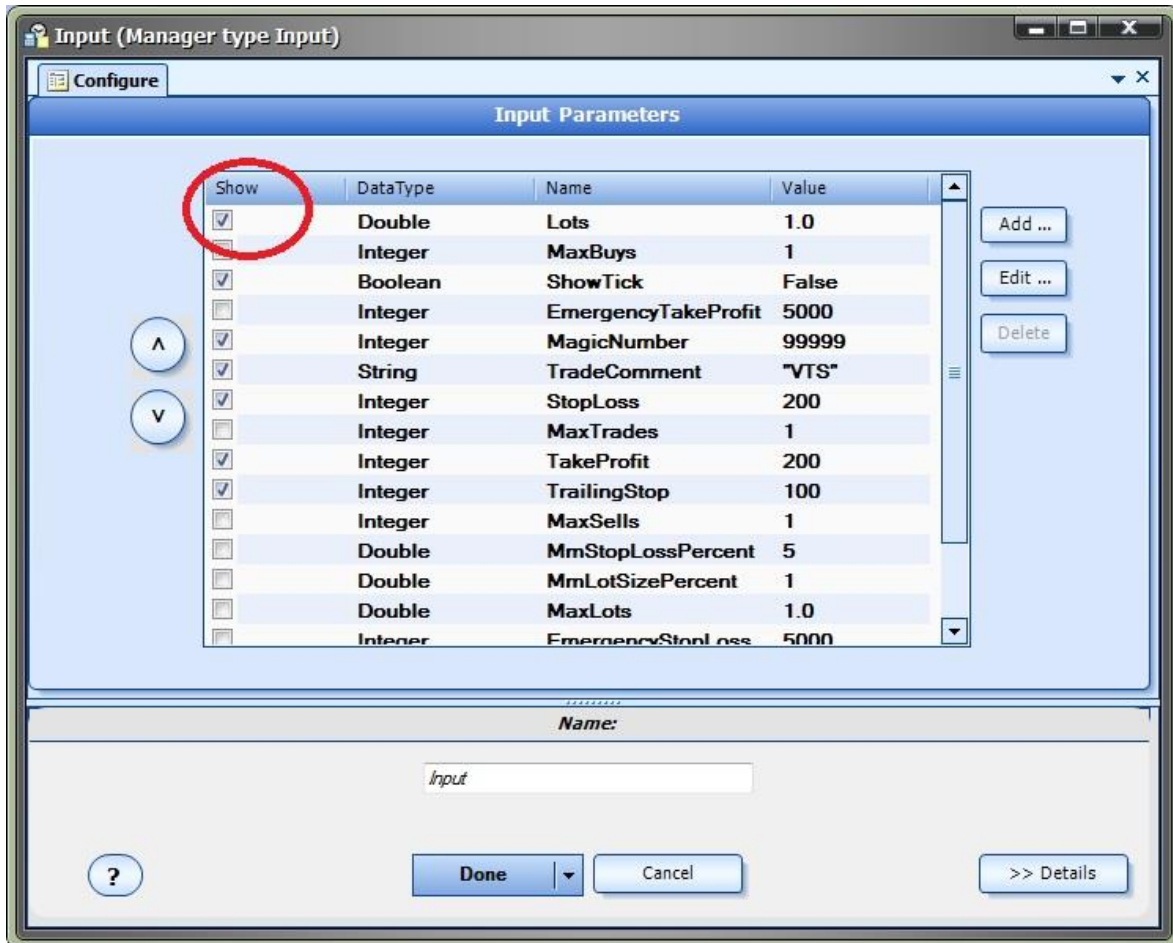


Chart Objects Plug-in

Requires VTS-Connect minimum version 4.0.0.65

The *Chart Objects* Plug-in allows you to easily add objects to any MetaTrader price chart. Chart objects include: horizontal lines, vertical lines, arrows, thumbs, labels, any text, check marks and stops signs. The chart object functions are available from the VTS Toolbox and are dragged, dropped and connected just like any other function.

What is a Plug-in?

VTS stands for *Visual Traders Studio*.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A *VTS Plug-in* allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the Chart Objects Plug-in

You must enter your License key to enable the *Chart Objects Plug-in*. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The email address is the email address used to purchase [VTS](#).

The License Key is the key listed in the Members Area.

The Update button is used to verify the email address and license key.

The edit key button is used edit the key value.

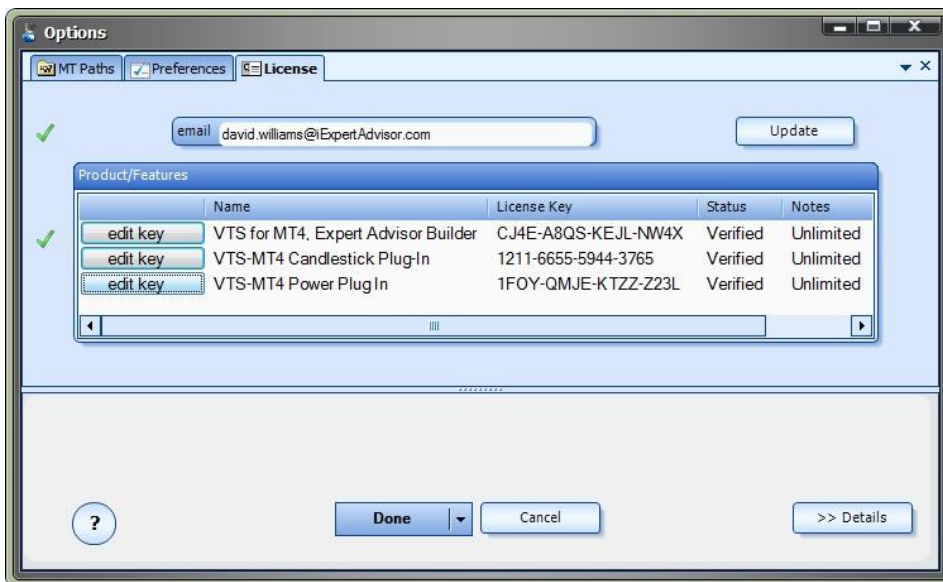


Chart Object Functions in the Toolbox

Once enabled, the *Chart Object* functions are available in the [Toolbox](#) Function tab under the *ChartObjects* menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.

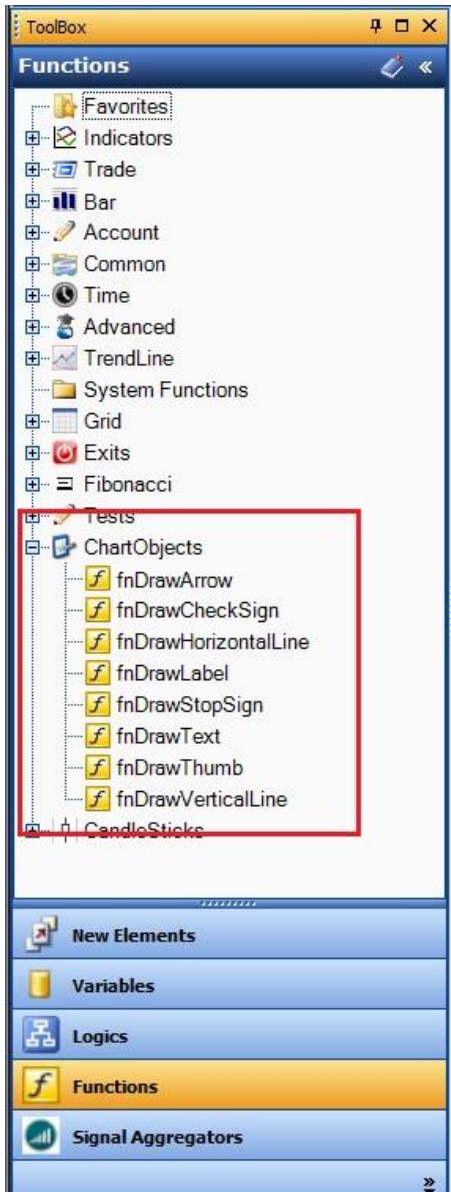


Chart Objects Functions

The Chart Objects function library includes these functions:

[fnDrawHorizontalLine](#)

Used to draw a horizontal line. Positioned by price.

[fnDrawVerticalLine](#)

Used to draw a vertical line. Positioned by candle.

[fnDrawArrow](#)

Used to draw an up or down arrow. Positioned by price and candle.

[fnDrawThumb](#)

Used to draw an up or down thumb. Positioned by price and candle.

[fnDrawLabel](#)

Used to draw an label containing text. Positioned by price and candle.

[fnDrawText](#)

Used to draw text. Positioned by x and y coordinates.

[fnDrawCheckSign](#)

Used to draw a check mark. Positioned by price and candle.

[fnDrawStopSign](#)

Used to draw a stop sign. Positioned by price and candle.

fnDrawHorizontalLine

The *fnDrawHorizontalLine* is used to draw a horizontal line on a price chart. It is positioned by price.

After the *fnDrawHorizontalLine* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

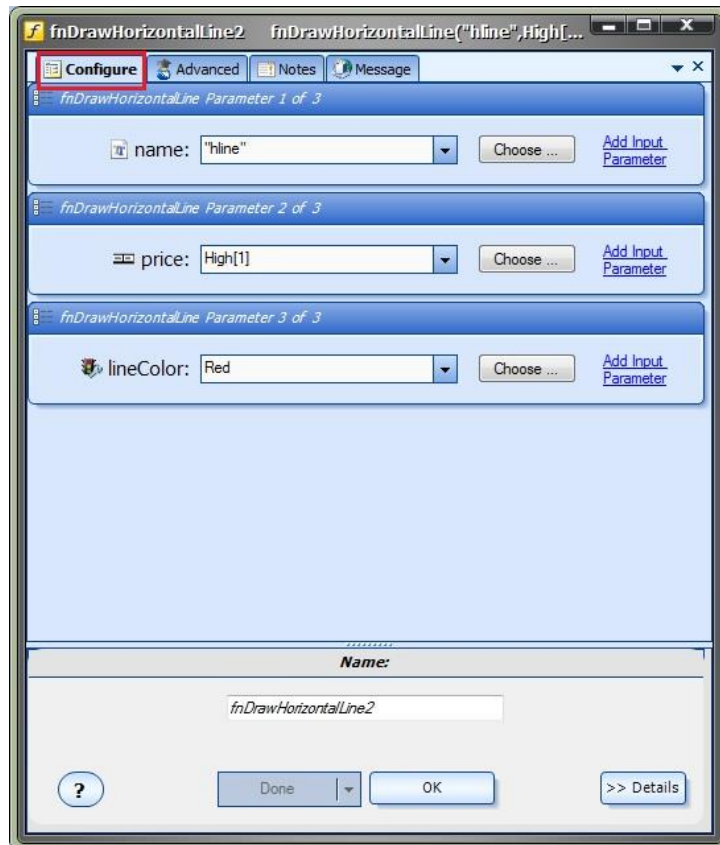
Each Chart Object function offers a number of parameters on the *Configuration* Tab, and also a number of parameters on the *Advanced* Tab.

Parameters on the *Configuration* Tab should be inspected and set each time the function is added to a drawing.

Parameters on the *Advanced* Tab usually do not need to be changed from their default values.

Parameters on the *Configuration* Tab

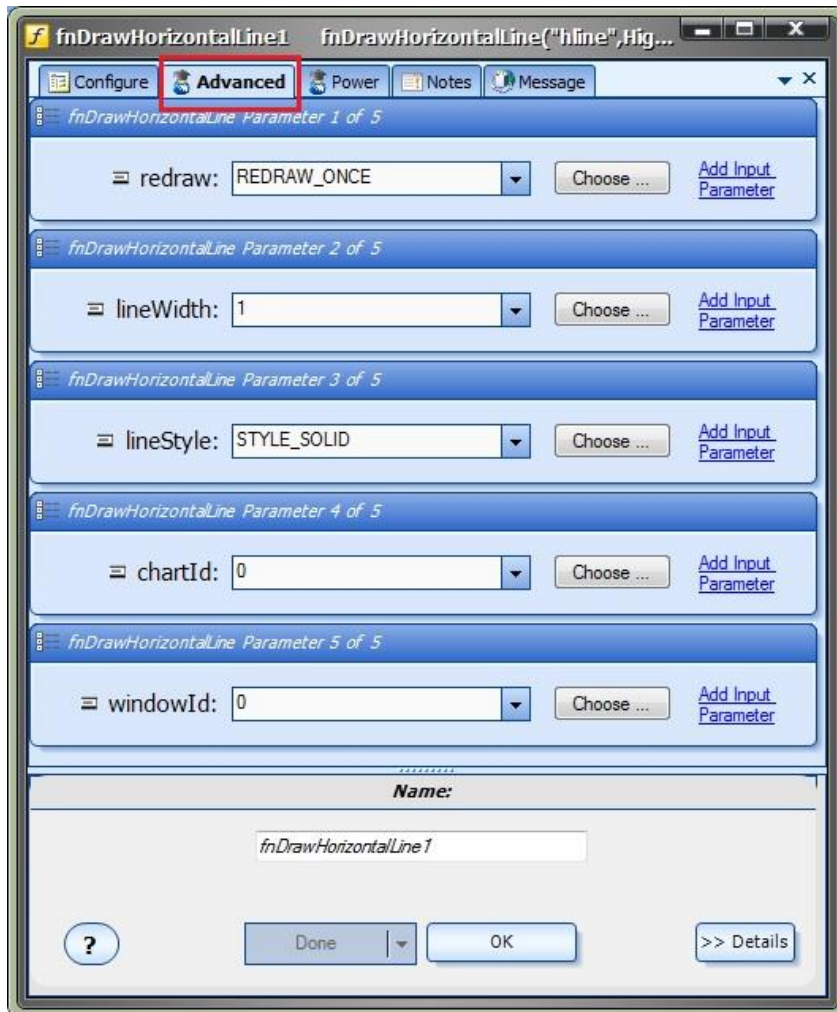
Parameter Name	Data type	Description
name	String	The name of the object. Each object on a chart must have a unique name. Note: String parameters require enclosing quotes. Objects names can viewed from the MetaTrader platform by going to: Charts->Objects->Object List
price	Double	The price value used to position the object's horizontal location on the price chart.
lineColor	Color	The color of the object.



Parameters on the *Advanced* Tab

Parameter Name	Data type	Description
redraw	Integer	<p>The interval when to redraw the object. The pull-down menu offers these choices:</p> <p>REDRAW_ONCE: draw the object once only.</p> <p>REDRAW_BAR: redraw the object on each new bar.</p> <p>REDRAW_TICK: redraw the object on each new tick.</p> <p>Note: The object function must be connected on the drawing so that it executes on each tick for the redraw options to work correctly.</p> <p>Objects that use the shift parameter will move on the chart as new candles are formed.</p>

lineWidth	Integer	The width of the line. Note: Styles only work when the lineWidth is 1.
lineStyle	Integer	The line style. The pull-down menu offers these choices: STYLE_SOLID STYLE_DASH STYLE_DOT STYLE_DASHDOT STYLE_DASHDOTDOT Note: Styles only work when the lineWidth is 1.
chartID	Integer	The chart identifier. Use 0 for the current chart.
windowID	Integer	The window identifier. Use 0 for the current window.



fnDrawVerticalLine

The *fnDrawVerticalLine* is used to draw a vertical line on a price chart. It is positioned by a [shift](#) value.

After the *fnDrawVerticalLine* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

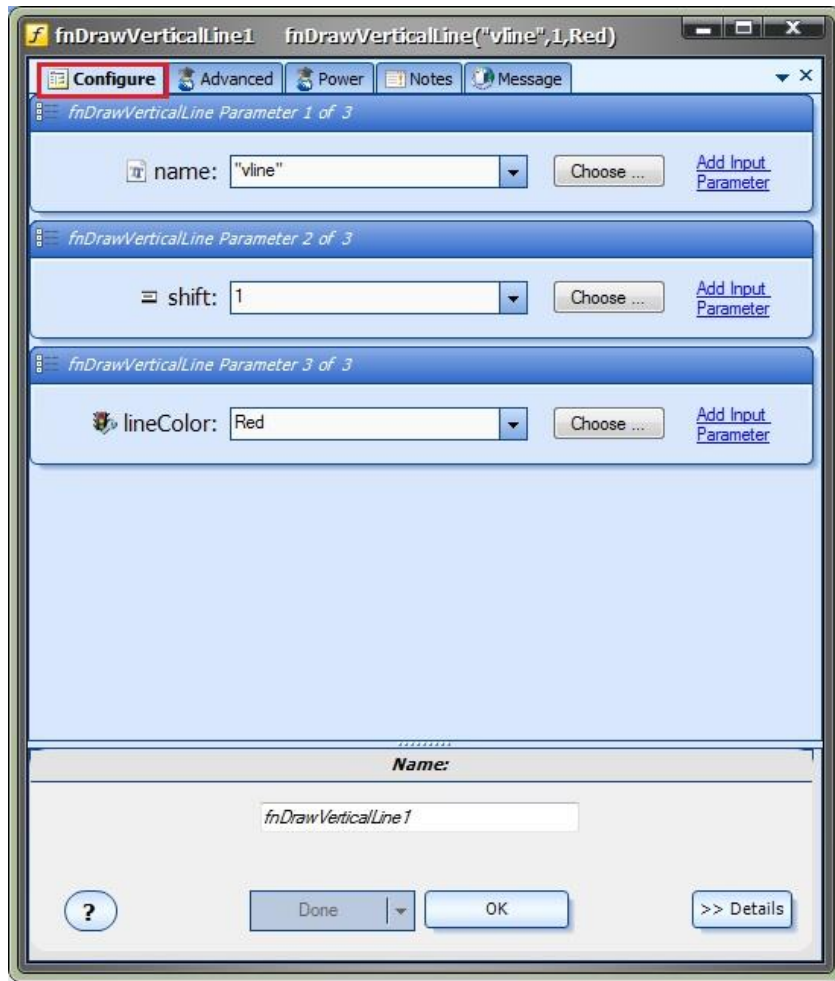
Each Chart Object function offers a number of parameters on the *Configuration* Tab, and also a number of parameters on the *Advanced* Tab.

Parameters on the *Configuration* Tab should be inspected and set each time the function is added to a drawing.

Parameters on the *Advanced* Tab usually do not need to be changed from their default values.

Parameters on the Configuration Tab

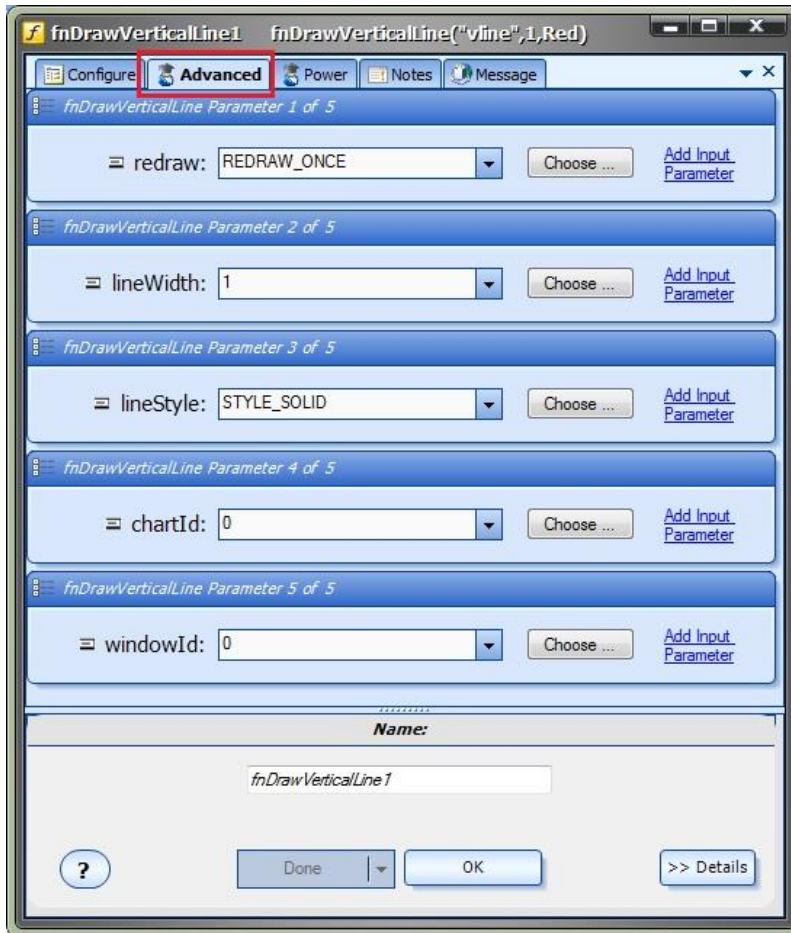
Parameter Name	Data type	Description
name	String	The name of the object. Each object on a chart must have a unique name. Note: String parameters require enclosing quotes. Objects names can viewed from the MetaTrader platform by going to: Charts->Objects->Object List
shift	Integer	The candle shift used to position the object's vertical location on the price chart.
lineColor	Color	The color of the object.



Parameters on the *Advanced* Tab

Parameter Name	Data type	Description
redraw	Integer	<p>The interval when to redraw the object. The pull-down menu offers these choices:</p> <p>REDRAW_ONCE: draw the object once only.</p> <p>REDRAW_BAR: redraw the object on each new bar.</p> <p>REDRAW_TICK: redraw the object on each new tick.</p> <p>Note: The object function must be connected on the drawing so that it executes on each tick for the redraw options to work correctly.</p>

		Objects that use the shift parameter will move on the chart as new candles are formed.
lineWidth	Integer	The width of the line. Note: Styles only work when the lineWidth is 1.
lineStyle	Integer	The line style. The pull-down menu offers these choices: STYLE_SOLID STYLE_DASH STYLE_DOT STYLE_DASHDOT STYLE_DASHDOTDOT Note: Styles only work when the lineWidth is 1.
chartID	Integer	The chart identifier. Use 0 for the current chart.
windowID	Integer	The window identifier. Use 0 for the current window.



fnDrawArrow

The **fnDrawArrow** is used to draw an up or down arrow on a price chart. It is positioned by price and shift value.

After the **fnDrawArrow** function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

Each Chart Object function offers a number of parameters on the **Configuration** Tab, and also a number of parameters on the **Advanced** Tab.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the **Advanced** Tab usually do not need to be changed from their default values.

Parameters on the Configuration Tab

Parameter Name	Data type	Description
type	Integer	Type of arrow. The pull-down menu offers these choices: OBJ_ARROW_UP OBJ_ARROW_DOWN
name	String	The name of the object. Each object on a chart must have a unique name. Note: String parameters require enclosing quotes. Objects names can viewed from the MetaTrader platform by going to: Charts->Objects->Object List
shift	Integer	The candle shift used to position the object's vertical location on the price chart.
price	Double	The price value used to position the object's horizontal location on the price chart.
lineColor	Color	The color of the object.



Parameters on the *Advanced* Tab

Parameter Name	Data type	Description
redraw	Integer	<p>The interval when to redraw the object. The pull-down menu offers these choices:</p> <p>REDRAW_ONCE: draw the object once only.</p> <p>REDRAW_BAR: redraw the object on each new bar.</p> <p>REDRAW_TICK: redraw the object on each new tick.</p> <p>Note: The object function must be connected on the drawing so that it executes on each tick for the redraw options to work correctly.</p> <p>Objects that use the shift parameter will move on the chart as new candles are formed.</p>
size	Integer	The size of the object.

lineStyle	Integer	The line style. The pull-down menu offers these choices: STYLE_SOLID STYLE_DASH STYLE_DOT STYLE_DASHDOT STYLE_DASHDOTDOT Note: Styles only work when the size is 1.
chartID	Integer	The chart identifier. Use 0 for the current chart.
windowID	Integer	The window identifier. Use 0 for the current window.



fnDrawThumb

The *fnDrawThumb* is used to draw an up or down thumb on a price chart. It is positioned by price and shift value.

After the *fnDrawThumb* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

Each Chart Object function offers a number of parameters on the **Configuration** Tab, and also a number of parameters on the **Advanced** Tab.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the **Advanced** Tab usually do not need to be changed from their default values.

Parameters on the Configuration Tab

Parameter Name	Data type	Description
type	Integer	Type of thumb. The pull-down menu offers these choices: OBJ_ARROW_THUMB_UP OBJ_ARROW_THUMB_DOWN
name	String	The name of the object. Each object on a chart must have a unique name. Note: String parameters require enclosing quotes. Objects names can viewed from the MetaTrader platform by going to: Charts->Objects->Object List
shift	Integer	The candle shift used to position the object's vertical location on the price chart.
price	Double	The price value used to position the object's horizontal location on the price chart.
lineColor	Color	The color of the object.



Parameters on the Advanced Tab

Parameter Name	Data type	Description
redraw	Integer	The interval when to redraw the object. The pull-down menu offers these choices: REDRAW_ONCE : draw the object once only. REDRAW_BAR : redraw the object on each new bar. REDRAW_TICK : redraw the object on each new tick. Note: The object function must be connected on the drawing so that it executes on each tick for the redraw options to work correctly. Objects that use the shift parameter will move on the chart as new candles are formed.
size	Integer	The size of the object.
lineStyle	Integer	The line style. The pull-down menu offers these choices:

		STYLE_SOLID STYLE_DASH STYLE_DOT STYLE_DASHDOT STYLE_DASHDOTDOT Note: Styles only work when the size is 1.
chartID	Integer	The chart identifier. Use 0 for the current chart.
windowID	Integer	The window identifier. Use 0 for the current window.



fnDrawLabel

The **fnDrawLabel** is used to draw a label containing text on a price chart. It is positioned by absolute x and y values.

After the **fnDrawLabel** function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

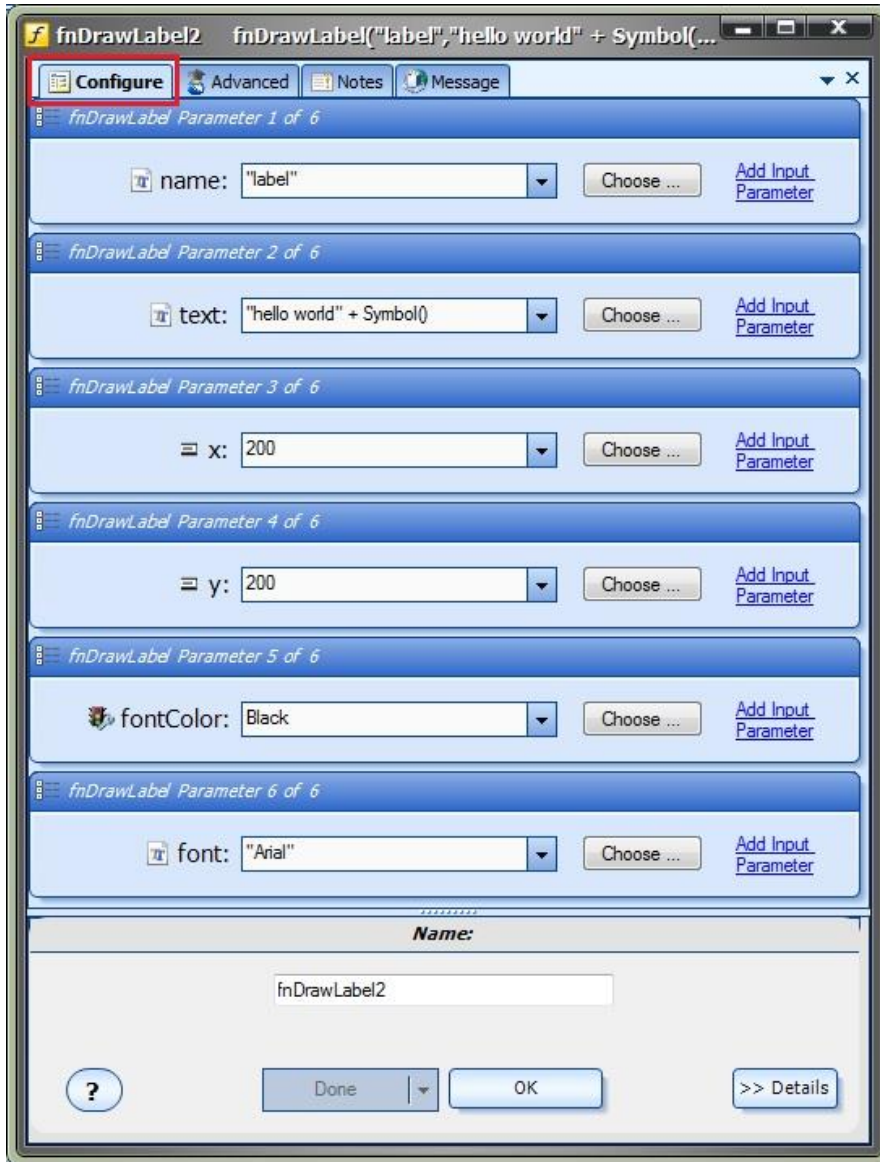
Each Chart Object function offers a number of parameters on the **Configuration** Tab, and also a number of parameters on the **Advanced** Tab.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the **Advanced** Tab usually do not need to be changed from their default values.

Parameters on the Configuration Tab

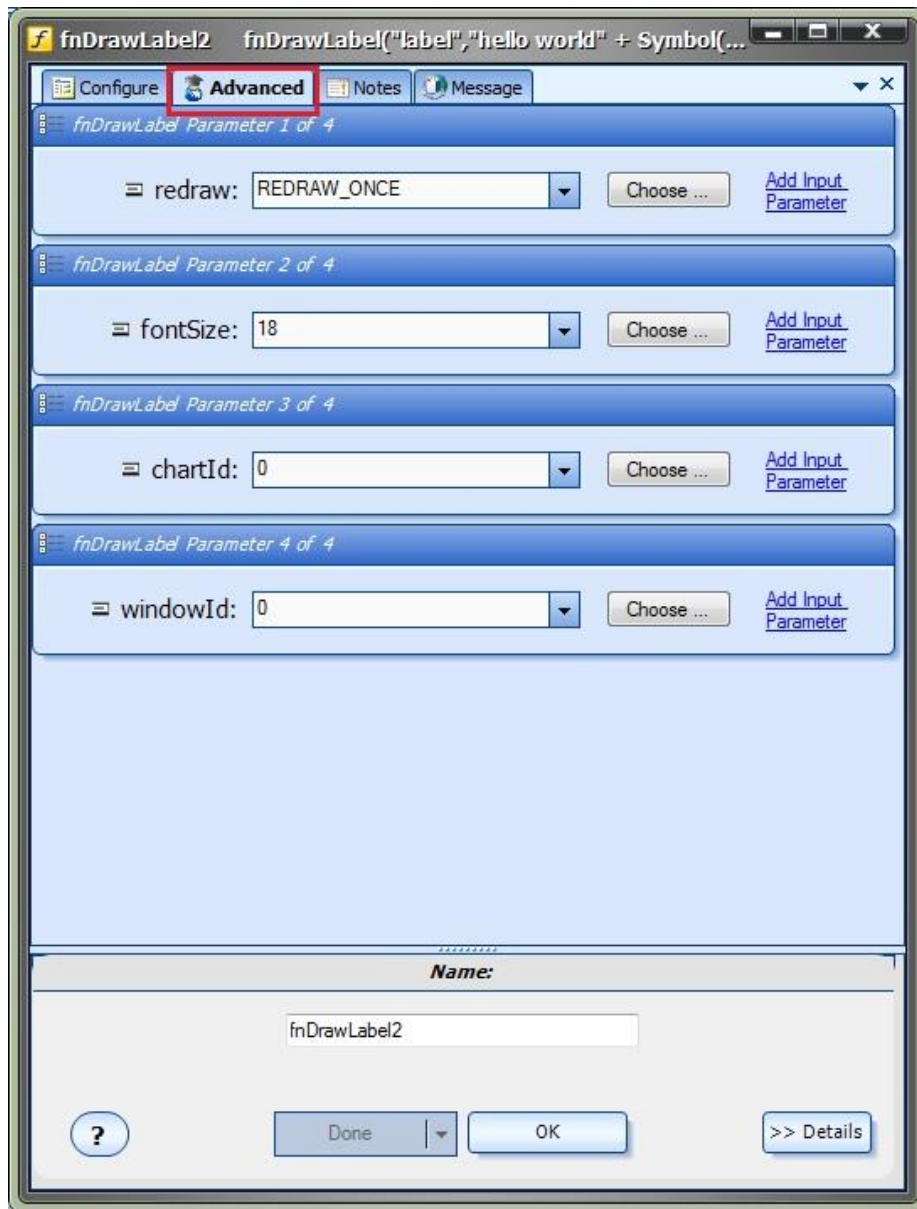
Parameter Name	Data type	Description
name	String	The name of the object. Each object on a chart must have a unique name. Note: String parameters require enclosing quotes. Objects names can viewed from the MetaTrader platform by going to: Charts->Objects->Object List
text	String	The text to be written on the chart. Note: String parameters require enclosing quotes
x	Integer	The x coordinate value used to position the object's horizontal location on the price chart. Note: The chart position where x and y are 0 is the bottom left corner of the chart.
y	Integer	The y coordinate value used to position the object's vertical location on the price chart. Note: The chart position where x and y are 0 is the bottom left corner of the chart.
fontColor	Color	The color of the font.
font	String	The name of the font.



Parameters on the Advanced Tab

Parameter Name	Data type	Description
redraw	Integer	<p>The interval when to redraw the object. The pull-down menu offers these choices:</p> <p>REDRAW_ONCE: draw the object once only.</p> <p>REDRAW_BAR: redraw the object on each new bar.</p> <p>REDRAW_TICK: redraw the object on each new tick.</p> <p>Note: The object function must be connected on the drawing so that it executes on each tick for the redraw options to work correctly.</p>

		Objects that use the shift parameter will move on the chart as new candles are formed.
fontSize	Integer	The size of the font.
chartID	Integer	The chart identifier. Use 0 for the current chart.
windowID	Integer	The window identifier. Use 0 for the current window.



fnDrawText

The **fnDrawText** is used to draw a label containing text on a price chart. It is positioned by price and candle.

After the **fnDrawText** function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

Each Chart Object function offers a number of parameters on the **Configuration** Tab, and also a number of parameters on the **Advanced** Tab.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the **Advanced** Tab usually do not need to be changed from their default values.

Parameters on the **Configuration** Tab

Parameter Name	Data type	Description
name	String	The name of the object. Each object on a chart must have a unique name. Note: String parameters require enclosing quotes. Objects names can viewed from the MetaTrader platform by going to: Charts->Objects->Object List
text	String	The text to be written on the chart. Note: String parameters require enclosing quotes
shift	Integer	The candle shift used to position the object's vertical location on the price chart.
price	Double	The price value used to position the object's horizontal location on the price chart.
fontColor	Color	The color of the font.



Parameters on the *Advanced* Tab

Parameter Name	Data type	Description
redraw	Integer	<p>The interval when to redraw the object. The pull-down menu offers these choices:</p> <p>REDRAW_ONCE: draw the object once only.</p> <p>REDRAW_BAR: redraw the object on each new bar.</p> <p>REDRAW_TICK: redraw the object on each new tick.</p> <p>Note: The object function must be connected on the drawing so that it executes on each tick for the redraw options to work correctly.</p> <p>Objects that use the shift parameter will move on the chart as new candles are formed.</p>
fontSize	Integer	The size of the font.

font	String	The name of the font.
chartID	Integer	The chart identifier. Use 0 for the current chart.
windowID	Integer	The window identifier. Use 0 for the current window.



fnDrawCheckSign

The **fnDrawCheckSign** is used to draw a check mark on a price chart. It is positioned by price and candle.

After the **fnDrawCheckSign** function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

Each Chart Object function offers a number of parameters on the **Configuration** Tab, and also a number of parameters on the **Advanced** Tab.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the **Advanced** Tab usually do not need to be changed from their default values.

Parameters on the Configuration Tab

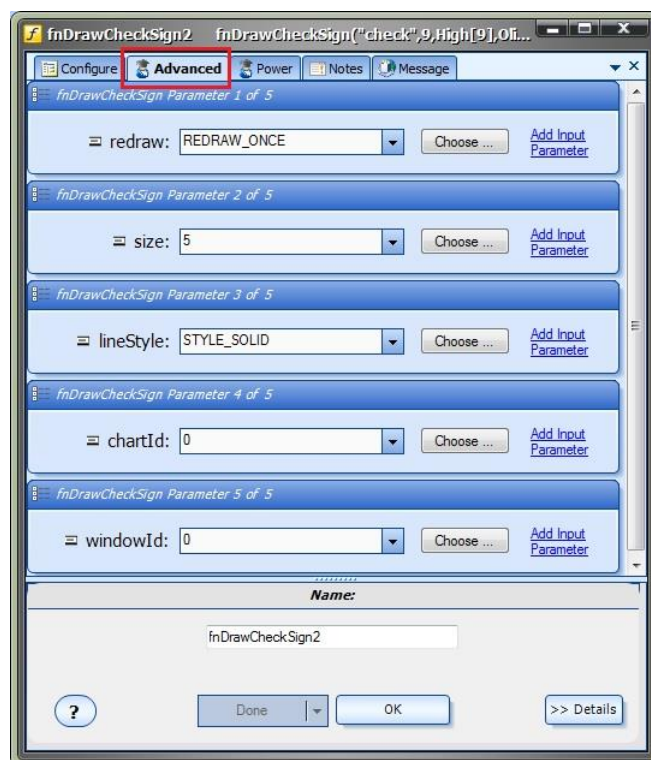
Parameter Name	Data type	Description
name	String	The name of the object. Each object on a chart must have a unique name. Note: String parameters require enclosing quotes. Objects names can viewed from the MetaTrader platform by going to: Charts->Objects->Object List
shift	Integer	The candle shift used to position the object's vertical location on the price chart.
price	Double	The price value used to position the object's horizontal location on the price chart.
lineColor	Color	The color of the object.



Parameters on the *Advanced* Tab

Parameter Name	Data type	Description
redraw	Integer	<p>The interval when to redraw the object. The pull-down menu offers these choices:</p> <p>REDRAW_ONCE: draw the object once only.</p> <p>REDRAW_BAR: redraw the object on each new bar.</p> <p>REDRAW_TICK: redraw the object on each new tick.</p> <p>Note: The object function must be connected on the drawing so that it executes on each tick for the redraw options to work correctly.</p>

		Objects that use the shift parameter will move on the chart as new candles are formed.
size	Integer	The size of the object.
lineStyle	Integer	The line style. The pull-down menu offers these choices: STYLE_SOLID STYLE_DASH STYLE_DOT STYLE_DASHDOT STYLE_DASHDOTDOT Note: Styles only work when the size is 1.
chartID	Integer	The chart identifier. Use 0 for the current chart.
windowID	Integer	The window identifier. Use 0 for the current window.



fnDrawStopSign

The **fnDrawStopSign** is used to draw a stop sign on a price chart. It is positioned by price and candle.

After the **fnDrawStopSign** function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

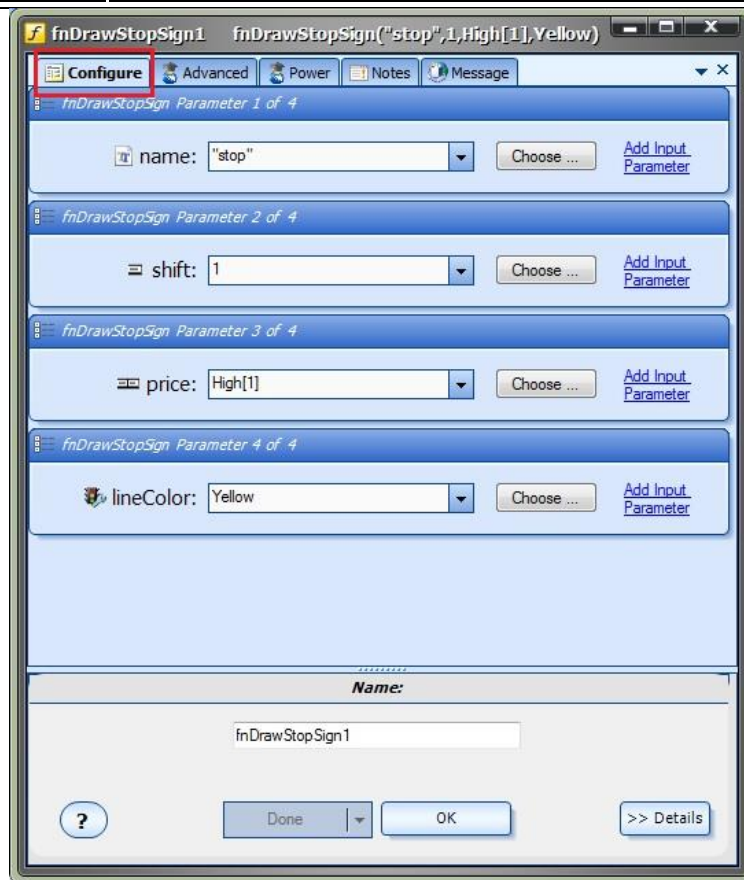
Each Chart Object function offers a number of parameters on the **Configuration** Tab, and also a number of parameters on the **Advanced** Tab.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the **Advanced** Tab usually do not need to be changed from their default values.

Parameters on the Configuration Tab

Parameter Name	Data type	Description
name	String	The name of the object. Each object on a chart must have a unique name. Note: String parameters require enclosing quotes. Objects names can viewed from the MetaTrader platform by going to: Charts->Objects->Object List
shift	Integer	The candle shift used to position the object's vertical location on the price chart.
price	Double	The price value used to position the object's horizontal location on the price chart.
lineColor	Color	The color of the object.



Parameters on the *Advanced* Tab

Parameter Name	Data type	Description
redraw	Integer	<p>The interval when to redraw the object. The pull-down menu offers these choices:</p> <p>REDRAW_ONCE: draw the object once only.</p> <p>REDRAW_BAR: redraw the object on each new bar.</p> <p>REDRAW_TICK: redraw the object on each new tick.</p> <p>Note: The object function must be connected on the drawing so that it executes on each tick for the redraw options to work correctly.</p> <p>Objects that use the shift parameter will move on the chart as new candles are formed.</p>
size	Integer	The size of the object.
lineStyle	Integer	<p>The line style. The pull-down menu offers these choices:</p> <p>STYLE_SOLID</p> <p>STYLE_DASH</p> <p>STYLE_DOT</p> <p>STYLE_DASHDOT</p> <p>STYLE_DASHDOTDOT</p> <p>Note: Styles only work when the size is 1.</p>
chartID	Integer	The chart identifier. Use 0 for the current chart.
windowID	Integer	The window identifier. Use 0 for the current window.



Chart Objects on the Drawing Pad

The Chart Object functions are dragged, dropped and connected from the [Toolbox](#) onto the [Drawing Pad](#) just like any other [function](#) in VTS.

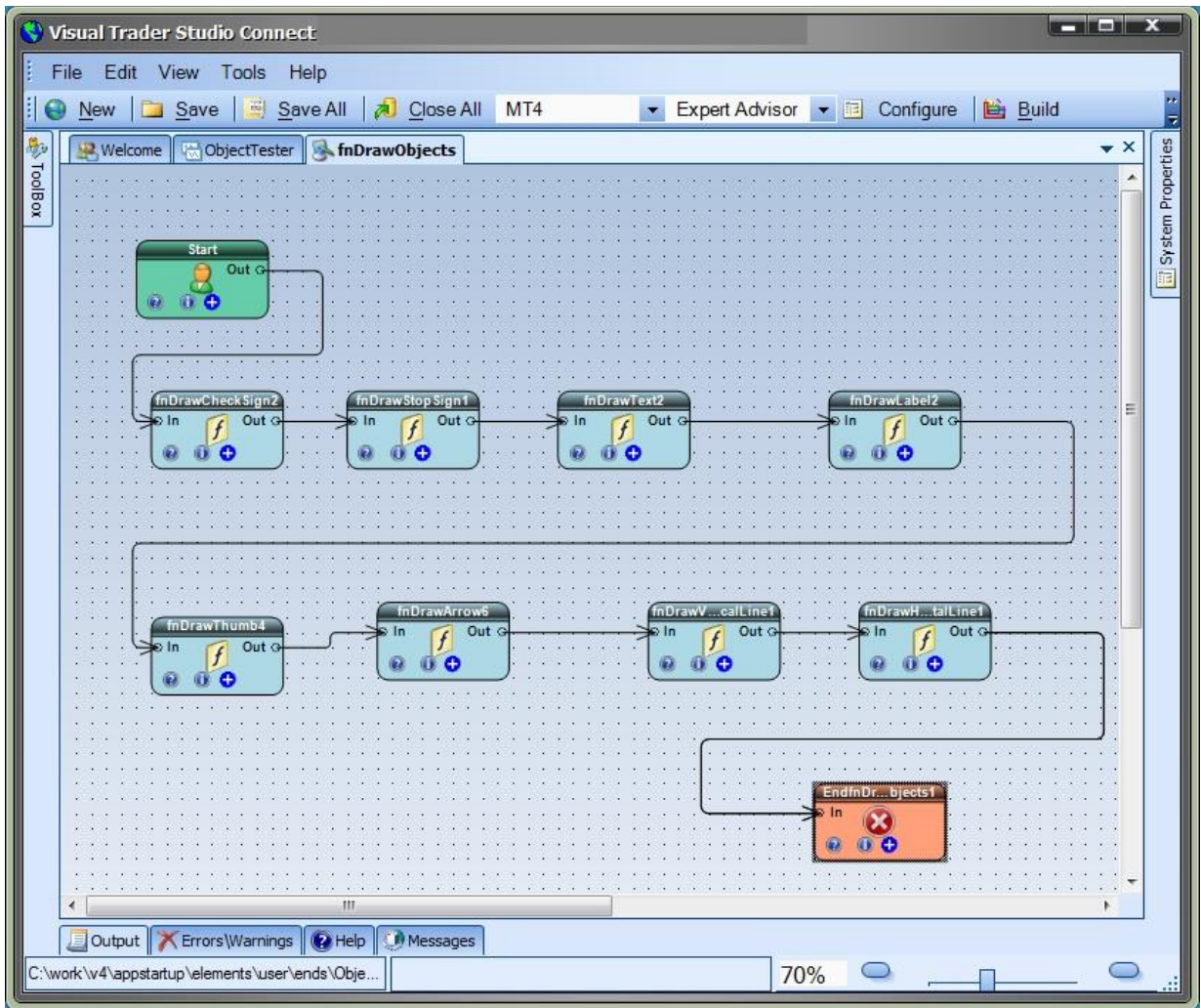
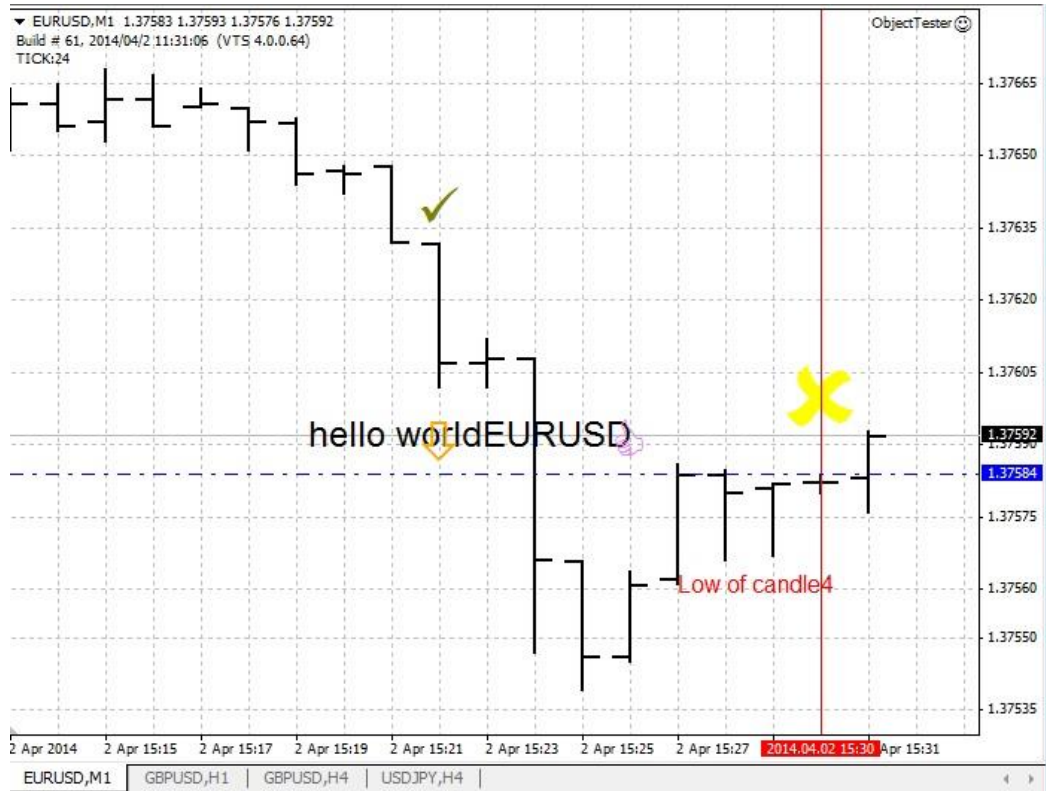


Chart Objects on a MetaTrader Price Chart

This price chart shows each of the Chart Object types.



Market Score Plug-in

Requires VTS-Connect minimum version 4.0.0.68

Use the **Market Score** plug-in to find the best market for your trading strategy. The **Market Score** plug-in calculates a score for your trading strategy on each market (or currency pair) that you choose. You can choose as many currency pairs as you like. **Market Score** finds the currency pair with highest score.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the Market Score Plug-in

You must enter your License key to enable the **Market Score Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

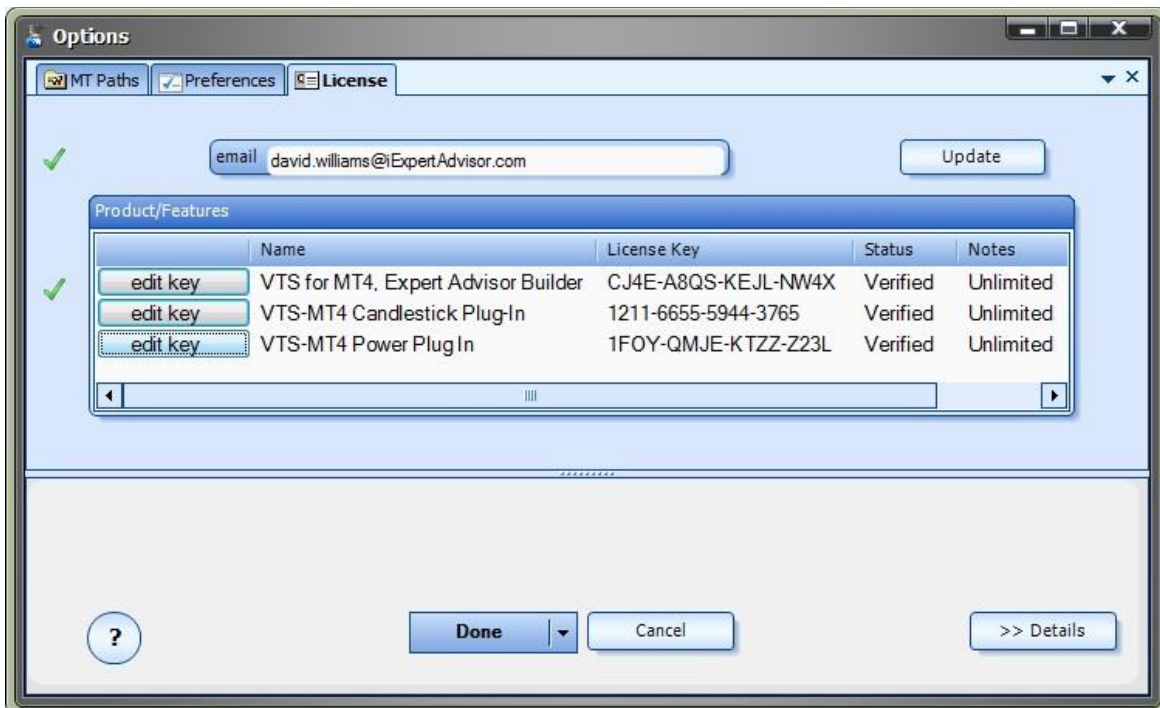
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The **email** address is the email address used to purchase [VTS](#).

The **License Key** is the key listed in the Members Area.

The **Update** button is used to verify the email address and license key.

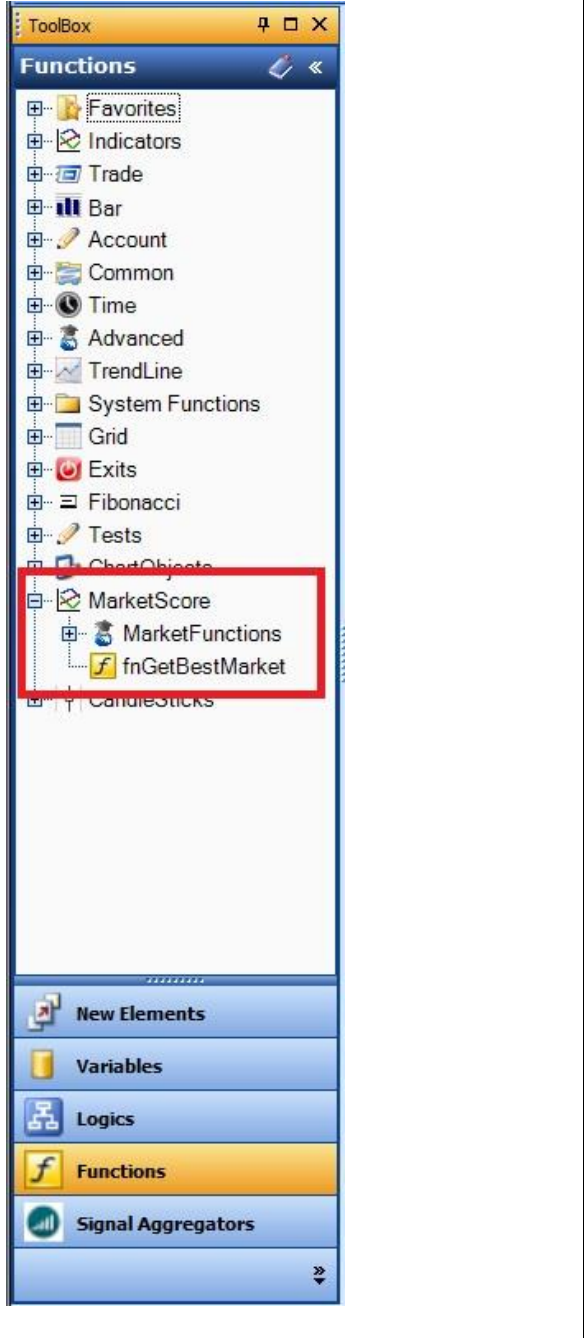
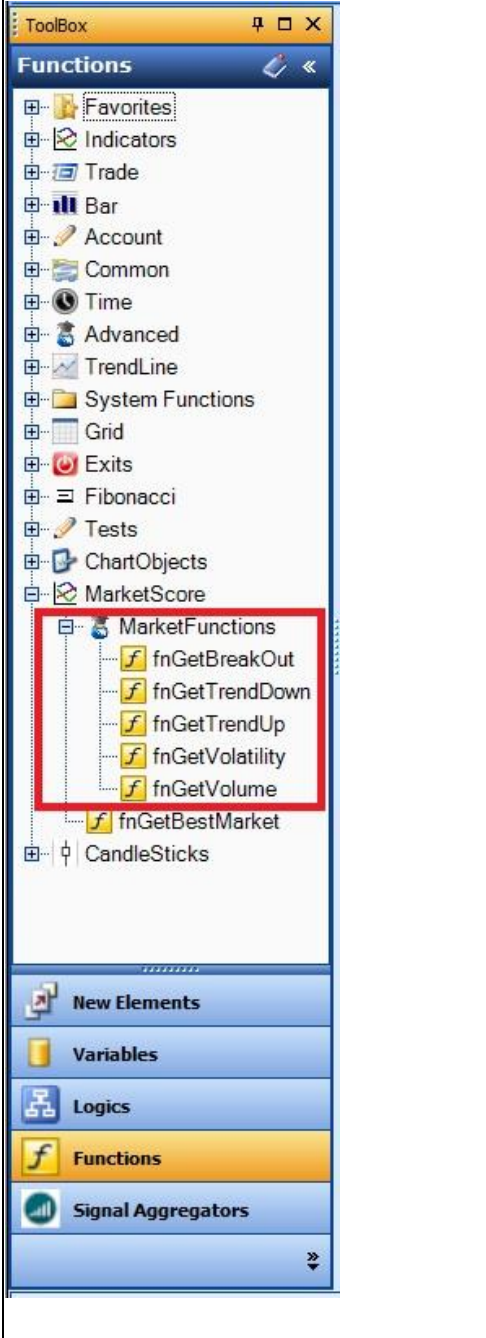
The **edit key** button is used edit the key value.



Market Score Functions in the Toolbox

Once enabled, the **Market Score** functions are available in the [Toolbox](#) Function tab under the **MarketScore** menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.

The main Market Score function <code>fnGetBestMarket</code>	The Market Functions used by <code>fnGetBestMarket</code>
 <p>The screenshot shows the 'ToolBox' window with the 'Functions' tab selected. The 'MarketScore' folder is expanded, and the 'fnGetBestMarket' function is highlighted with a red rectangular box. Below the toolbox are buttons for 'New Elements', 'Variables', 'Logics', 'Functions', and 'Signal Aggregators'.</p>	 <p>The screenshot shows the 'ToolBox' window with the 'Functions' tab selected. The 'MarketFunctions' folder is expanded, and the entire folder is highlighted with a red rectangular box. The sub-functions listed are: 'fnGetBreakOut', 'fnGetTrendDown', 'fnGetTrendUp', 'fnGetVolatility', 'fnGetVolume', and 'fnGetBestMarket'. Below the toolbox are buttons for 'New Elements', 'Variables', 'Logics', 'Functions', and 'Signal Aggregators'.</p>

Market Score Functions

The **Market Score** function library uses a single main function to find the best market: **fnGetBestMarket**

The function [fnGetBestMarket](#) uses any of the available [Market Functions](#) to calculate the market score for all markets.

The built-in **Market Functions** are:

fnGetVolatility

fnGetVolume

fnGetTrendUp

fnGetTrendDown

fnGetBreakOut

Note: Any number of **Market Functions** can be [added](#).

fnGetBestMarket

The function **fnGetBestMarket** is used to find the single best market (or currency pair) for a given strategy.

After the **fnGetBestMarket** function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

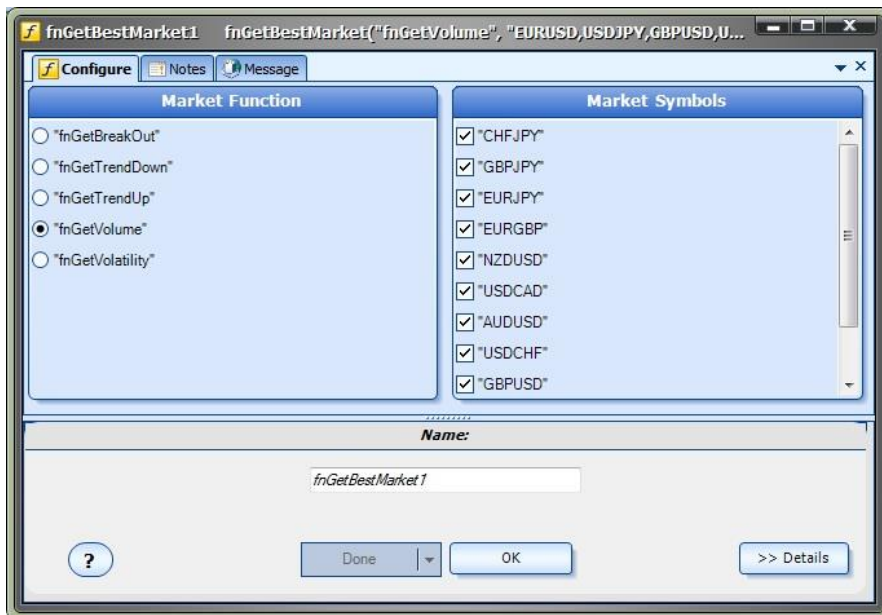
The [Function Configuration](#) window allows you to select values for each parameter.

The **Market Function** section allows you to select a single market function.

(To add more functions see [Adding Market Functions](#))

The **Market Symbols** section allows you to any number of currency pairs.

(To add more symbols see [Adding Symbols](#))



Market Functions

[fnGetVolatility](#)

[fnGetVolume](#)

[fnGetTrendUp](#)

[fnGetTrendDown](#)

[fnGetBreakOut](#)

Note: Any number of **Market Functions** can be [added](#).

fnGetBreakOut

The *market function* **fnGetBreakOut** calculates the degree to which a market is breaking out of a current price pattern.

Notes:

A *market function* is any VTS function that accepts a symbol as a parameter and returns a positive number.

The [fnGetBestMarket](#) function calls a *market function* for all selected symbols and returns the symbol with the highest score.

Although a *market function* can be dragged and dropped onto a drawing, **typically this is not done**. Instead the *market function* is selected in *Function* section of the [fnGetBestMarket](#) function

fnGetTrendDown

The *market function* **fnGetTrendDown** calculates the degree to which a market is trending down.

Notes:

A *market function* is any VTS function that accepts a symbol as a parameter and returns a positive number.

The [fnGetBestMarket](#) function calls a *market function* for all selected symbols and returns the symbol with the highest score.

Although a *market function* can be dragged and dropped onto a drawing, **typically this is not done**. Instead the *market function* is selected in *Function* section of the [fnGetBestMarket](#) function

fnGetTrendUp

The *market function* **fnGetTrendUp** calculates the degree to which a market is trending up.

Notes:

A *market function* is any VTS function that accepts a symbol as a parameter and returns a positive number.

The [fnGetBestMarket](#) function calls a *market function* for all selected symbols and returns the symbol with the highest score.

Although a *market function* can be dragged and dropped onto a drawing, **typically this is not done**. Instead the *market function* is selected in *Function* section of the [fnGetBestMarket](#) function

fnGetVolatility

The *market function* **fnGetVolatility** calculates volatility of a market.

Notes:

A *market function* is any VTS function that accepts a symbol as a parameter and returns a positive number.

The [fnGetBestMarket](#) function calls a *market function* for all selected symbols and returns the symbol with the highest score.

Although a *market function* can be dragged and dropped onto a drawing, **typically this is not done**. Instead the *market function* is selected in *Function* section of the [fnGetBestMarket](#) function

fnGetVolume

The *market function* **fnGetVolume** calculates volume of a market.

Notes:

A *market function* is any VTS function that accepts a symbol as a parameter and returns a positive number.

The [fnGetBestMarket](#) function calls a *market function* for all selected symbols and returns the symbol with the highest score.

Although a *market function* can be dragged and dropped onto a drawing, **typically this is not done**. Instead the *market function* is selected in *Function* section of the [fnGetBestMarket](#) function

Adding Market Functions

A *market function* can be any VTS function that accepts a symbol as a parameter and returns a positive number.

The [fnGetBestMarket](#) function calls a *market function* for all selected symbols and returns the symbol with the highest score.

Although a *market function* can be dragged and dropped onto a drawing, **typically this is not done**. Instead the *market function* is selected in *Function* section of the [fnGetBestMarket](#) function.

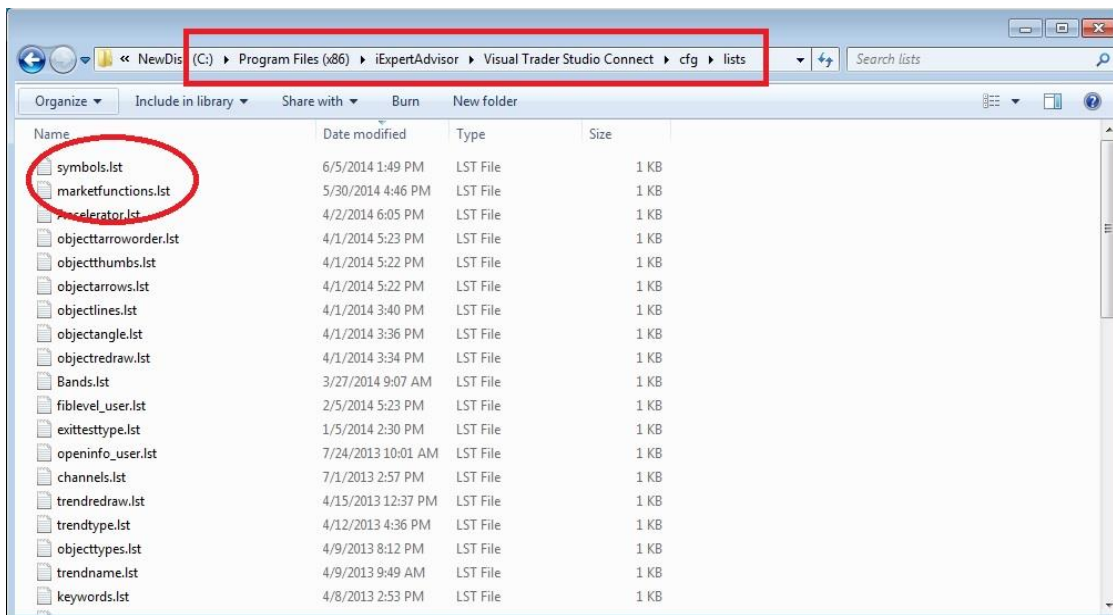
Any MQL function in VTS can be used as a market function. The function must have a single [string](#) parameter and return a [double](#) value.

To have the function appear in *Function* section of the [fnGetBestMarket](#) function, the function name is added to the **marketfunction.lst** file.

The **marketfunction.lst** file is a text file that lists all of the available market functions. It can be edited with any text editor, including NotePad.

The **marketfunction.lst** file is located at:

C:\Program Files (x86)\iExpertAdvisor\Visual Trader Studio Connect\cfg\list



The original contents of the **marketfunction.lst** file:

"fnGetVolatility"

"fnGetVolume"

"fnGetTrendUp"

"fnGetTrendDown"

"fnGetBreakOut"

To add a new function, enter the name of the function, in quotes, on a new line and save the file. For example:

"MyNewMqlFunction"

Adding Symbols

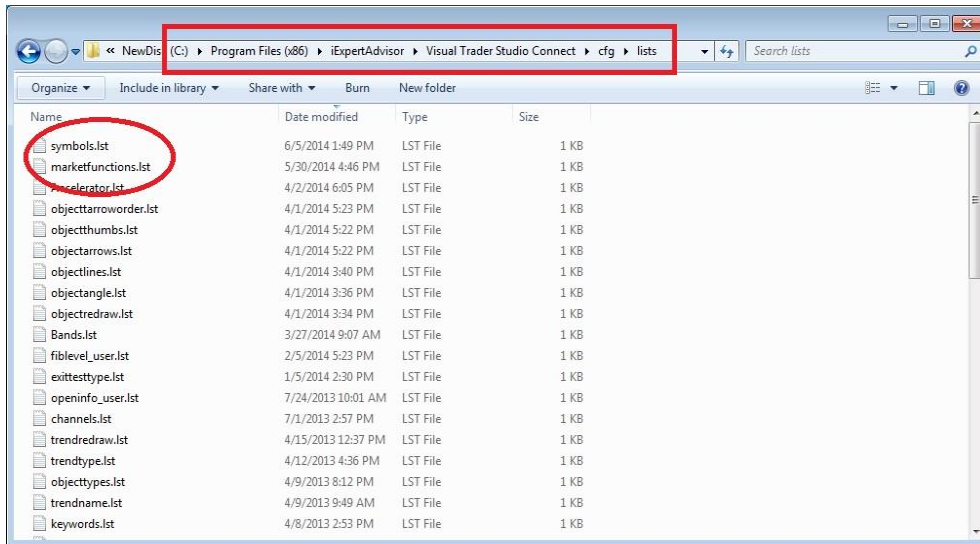
Any Symbol supported by your MetaTrader broker can be added to the **Market Score** Plug-in.

To have the function appear in *Symbols* section of the [fnGetBestMarket](#) function, the function name must be added to the **symbols.lst** file.

The **symbols.lst** file is a text file that lists all of the available market functions. It can be edited with any text editor, including NotePad.

The **symbols.lst** file is located at:

C:\Program Files (x86)\iExpertAdvisor\Visual Trader Studio Connect\cfg\list



The original contents of the **symbols.lst** file:

```
CHART
"EURUSD"
"USDJPY"
"GBPUSD"
"USDCHF"
"AUDUSD"
"USDCAD"
"NZDUSD"
"EURGBP"
"EURJPY"
"GBPJPY"
"CHFJPY"
```

To add a new symbol, enter the name of the symbol, in quotes, on a new line and save the file. For example:

```
"mEURUSD"
```

Market Score on the Drawing Pad

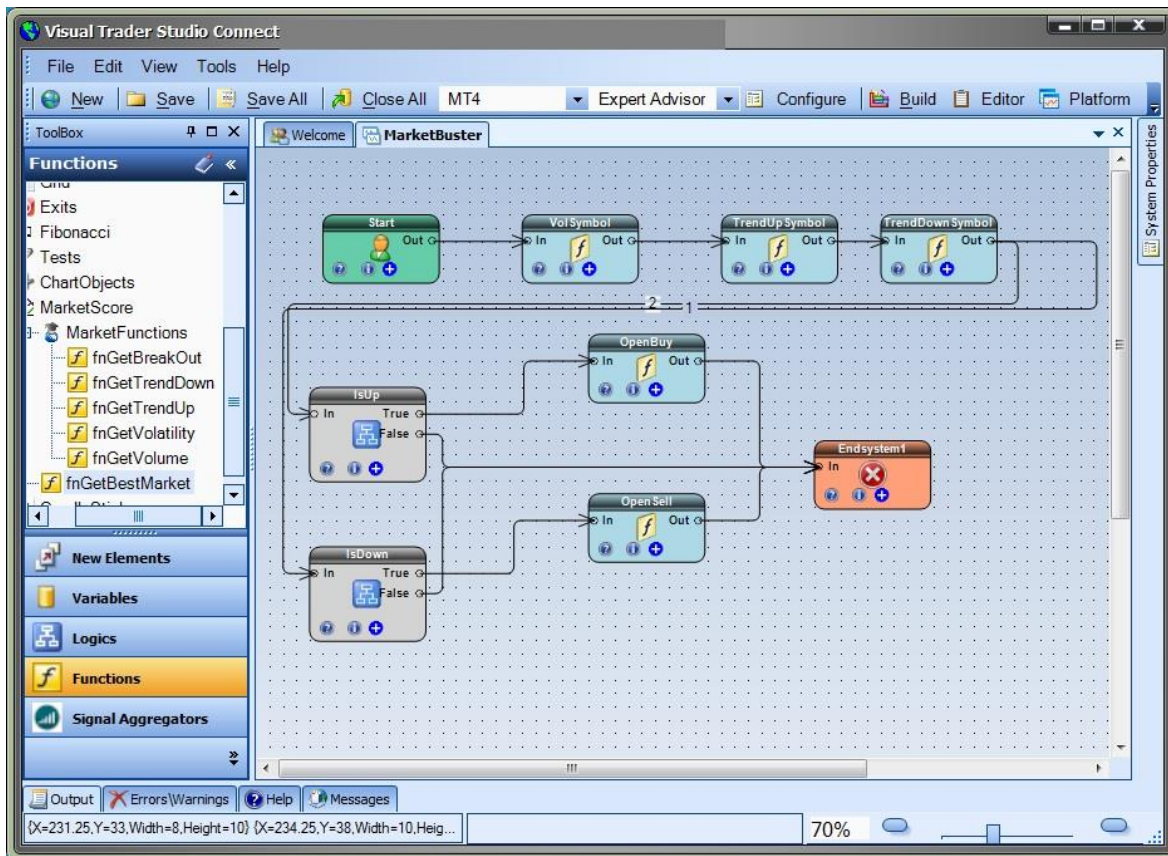
The image below shows how the **Market Score** plug-in is used.

The [FunctionElements](#) VolSymbol, TrendUpSymbol and TrendDownSymbol are [fnGetBestMarket](#) functions renamed for clarity.

The variables that hold the return values for the functions are named _VolSymbol, _TrendUpSymbol and _TrendDownSymbol. (**Note the preceding underscore** _). These variables will hold the symbol with the highest market score.

The [LogicElements](#) IsUp and IsDown use the variables to make a trading decision.

The [OpenBuy](#) and [OpenSell](#) functions use the selected variable to open an order for the Symbol with the highest score.



Order History Plug-in

Requires VTS-Connect minimum version 4.0.0.69

Use the **Order History** plug-in to get information about your closed trades. With the **Order History** plug-in, you can get any information about your closed trades. Keep emotions out of your trading by creating rock-solid rules about when to keep trading in the face of multiple winning or losing trades.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the Order History Plug-in

You must enter your License key to enable the **Order History Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

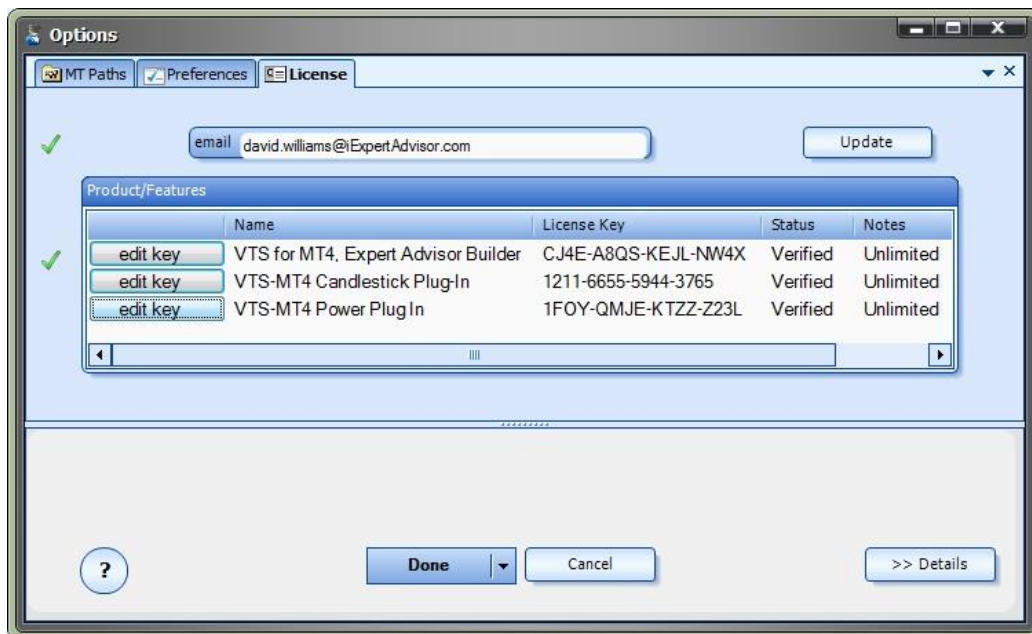
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The **email** address is the email address used to purchase [VTS](#).

The **License Key** is the key listed in the Members Area.

The **Update** button is used to verify the email address and license key.

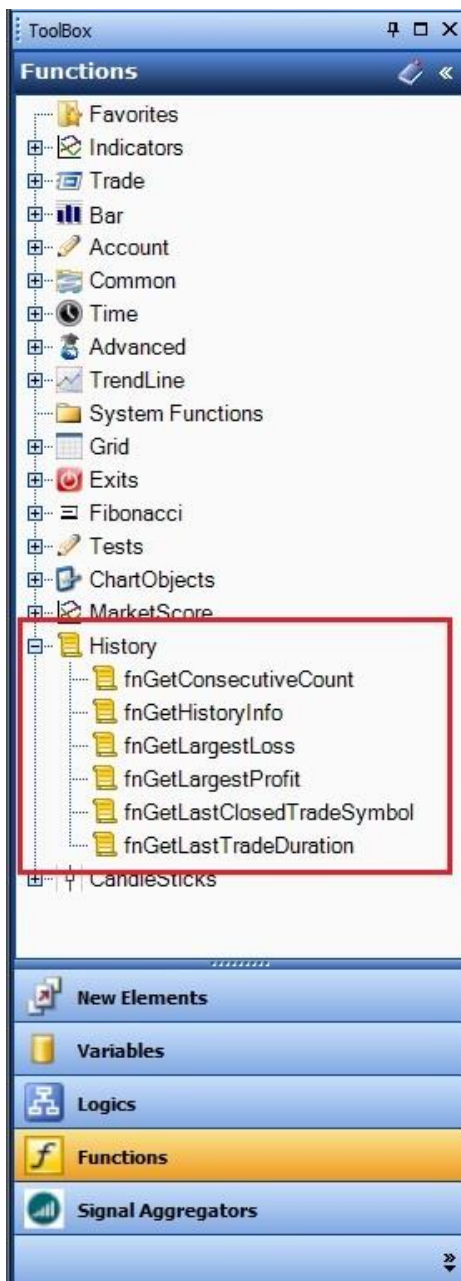
The **edit key** button is used edit the key value.



Order History Functions in the Toolbox

Once enabled, the **Order History** functions are available in the [Toolbox](#) Function tab under the **History** menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.



Order History Functions

The **Order History** function library includes these functions:

The [fnGetHistoryInfo](#) function is used to get information about a closed order.

The [fnGetLastTradeDuration](#) is used to get the duration, in minutes, of the last closed trade.

The [fnGetLastClosedTradeSymbol](#) is used to get the currency symbol of the last closed trade.

The [fnGetConsecutiveCount](#) function is used to get the number of the last consecutive winning or losing trades.

The [fnGetLargestProfit](#) function is used to find the largest profit of a closed trade within the last X trades or minutes.

The [fnGetLargestLoss](#) function is used to find the largest loss of a closed trade within the last X trades or minutes.

fnGetHistoryInfo

The **fnGetHistoryInfo** is used to get information about the last closed trade on the account.

After the **fnGetHistoryInfo** function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

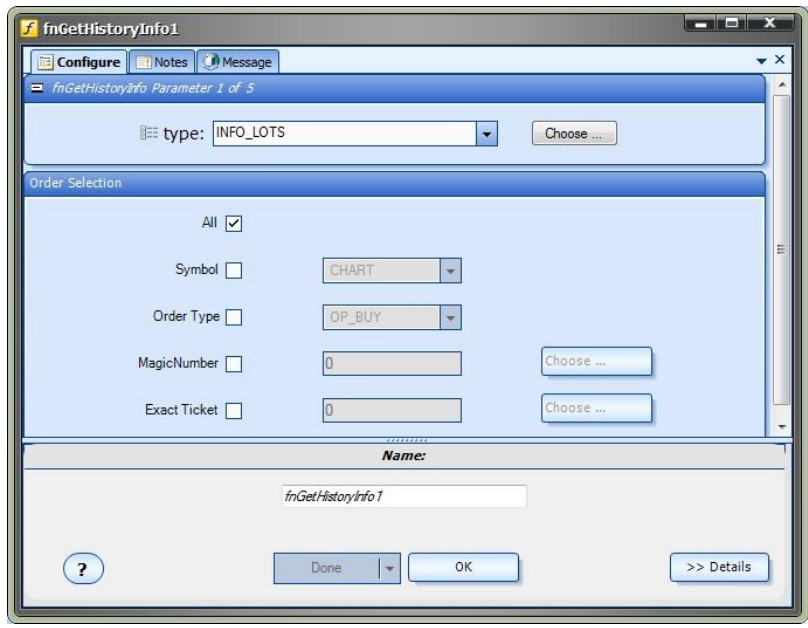
The [Function Configuration](#) window allows you to select values for each parameter.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the Configuration Tab

The following table provides information about each [parameter](#) of the **fnGetHistoryInfo** function.

InfoType	<p>The type of information requested (that matches the Order Selection criteria):</p> <p>INFO_LOTS: Get the lot size of the last closed order.</p> <p>INFO_MAGICNUMBER : Get the magic number of the last closed order.</p> <p>INFO_OPENPRICE : Get the open price of the last closed order.</p> <p>INFO_CLOSEPRICE : Get the closed price of the last closed order.</p> <p>INFO_OPENTIME : Get the open time of the last closed order.</p> <p>INFO_CLOSETIME : Get the close time of the last closed order.</p> <p>INFO_PROFIT : Get the profit of the last closed order. (this value can be negative)</p> <p>INFO_STOPLOSS : Get the stoploss of the last closed order.</p> <p>INFO_TAKEPROFIT : Get the takeprofit of the last closed order.</p> <p>INFO_SWAP : Get the swap value of the last closed order.</p> <p>INFO_TICKET : Get the ticket of the last closed order.</p> <p>INFO_ORDERTYPE : Get the order type of the last closed order.</p>
Order Selection	<p>Determines what orders of the Account are queried.</p> <p>All: all open orders on the account.</p> <p>Symbol: only open orders matching the selected symbol.</p> <p>Order Type: only open orders of the selected order type.</p> <p>MagicNumber: only open orders with a matching magicnumber.</p> <p>Exact Ticket: only open orders with the matching order ticket.</p> <p>Note: Selecting Exact Ticket disables all other selection criteria.</p>



fnGetLastTradeDuration

The *fnGetLastTradeDuration* is used to get the duration, **in minutes**, of the last closed trade on the account that matches the selection criteria..

After the *fnGetLastTradeDuration* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

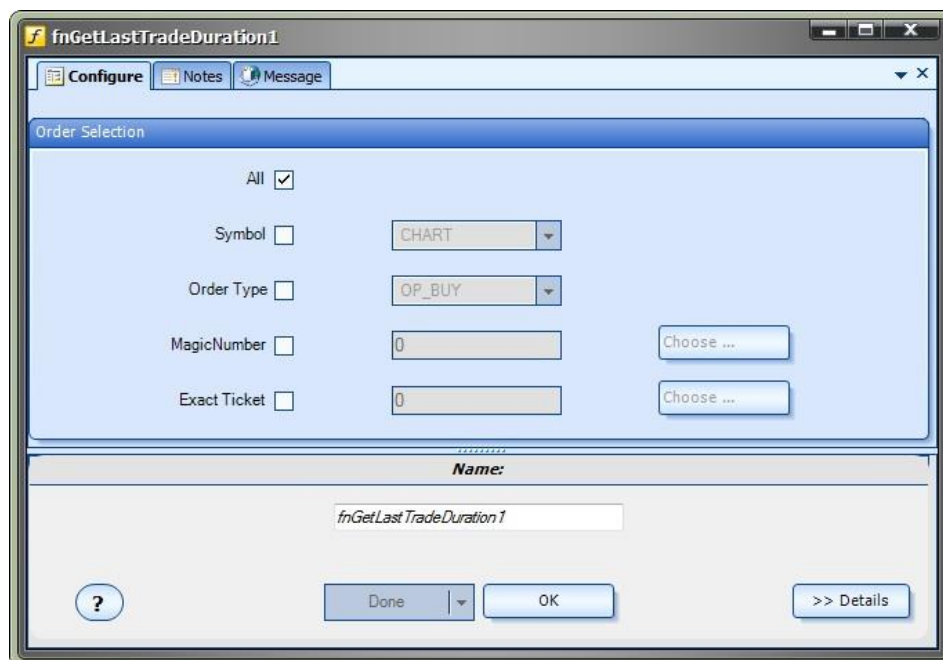
The [Function Configuration](#) window allows you to select values for each parameter.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the Configuration Tab

The following table provides information about each [parameter](#) of the *fnGetLastTradeDuration* function.

Order Selection	Determines what orders of the Account are queried. All: all open orders on the account. Symbol: only open orders matching the selected symbol. Order Type: only open orders of the selected order type . MagicNumber: only open orders with a matching magicnumber . Exact Ticket: only open orders with the matching order ticket . Note: Selecting Exact Ticket disables all other selection criteria.
------------------------	---



fnGetLastClosedTradeSymbol

The *fnGetLastClosedTradeSymbol* is used to get the currency symbol of the last closed trade on the account that matches the selection criteria..

After the *fnGetLastClosedTradeSymbol* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

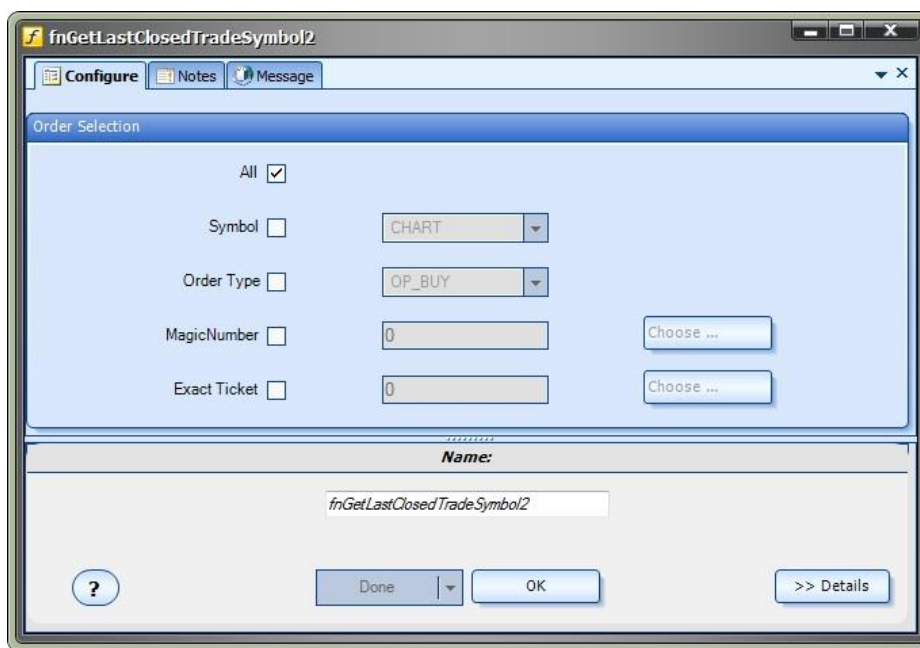
The [Function Configuration](#) window allows you to select values for each parameter.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the Configuration Tab

The following table provides information about each [parameter](#) of the *fnGetLastClosedTradeSymbol* function.

Order Selection	Determines what orders of the Account are queried. All: all open orders on the account. Symbol: only open orders matching the selected symbol. Order Type: only open orders of the selected order type . MagicNumber: only open orders with a matching magicnumber . Exact Ticket: only open orders with the matching order ticket . Note: Selecting Exact Ticket disables all other selection criteria.
------------------------	---



fnGetConsecutiveCount

The **fnGetConsecutiveCount** is used to get the currency symbol of the last closed trade on the account that matches the selection criteria.

For example:

If the last closed trade was profitable, the number of consecutive WINS is 1, and the number of consecutive losses is 0.

If the last 2 closed trades were profitable, the number of consecutive WINS is 2, and the number of consecutive losses is 0.

After the **fnGetConsecutiveCount** function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

The [Function Configuration](#) window allows you to select values for each parameter.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the Configuration Tab

The following table provides information about each [parameter](#) of the **fnGetConsecutiveCount** function.

type	The type of trades: WINS: returns the number of consecutive winning trades (profit greater than or equal to zero). LOSSES: returns the number of consecutive losing trades (profit less than zero).
Order Selection	Determines what orders of the Account are queried. All: all open orders on the account. Symbol: only open orders matching the selected symbol. Order Type: only open orders of the selected order type . MagicNumber: only open orders with a matching magicnumber . Exact Ticket: only open orders with the matching order ticket . Note: Selecting Exact Ticket disables all other selection criteria.



fnGetLargestProfit

The *fnGetLargestProfit* is used to get the largest profit of a closed trade that matches the selection criteria.

After the *fnGetLargestProfit* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

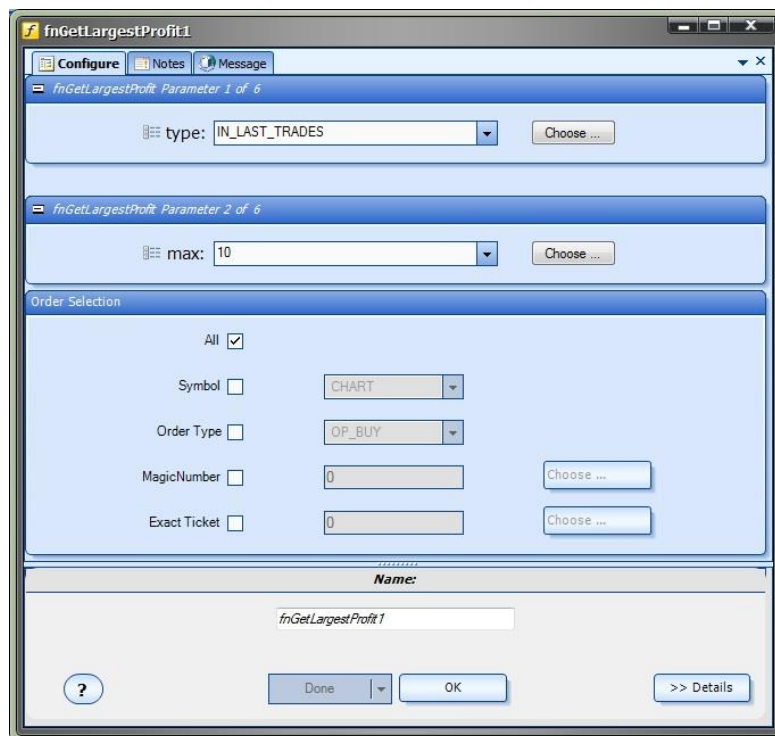
The [Function Configuration](#) window allows you to select values for each parameter.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the Configuration Tab

The following table provides information about each [parameter](#) of the *fnGetLargestProfit* function.

type	The type of trades: IN_LAST_TRADES: Considers the last max number of closed trades to calculate the largest profit. IN_LAST_MINUTES: Considers the closed trades in the last max minutes to calculate the largest profit.
max	The number of minutes or trades to consider when calculating the largest profit. The default value is 10.
Order Selection	Determines what orders of the Account are queried. All: all open orders on the account. Symbol: only open orders matching the selected symbol. Order Type: only open orders of the selected order type . MagicNumber: only open orders with a matching magicnumber . Exact Ticket: only open orders with the matching order ticket . Note: Selecting Exact Ticket disables all other selection criteria.



fnGetLargestLoss

The **fnGetLargestLoss** is used to get the largest profit of a closed trade that matches the selection criteria.

After the **fnGetLargestLoss** function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the [Element](#).

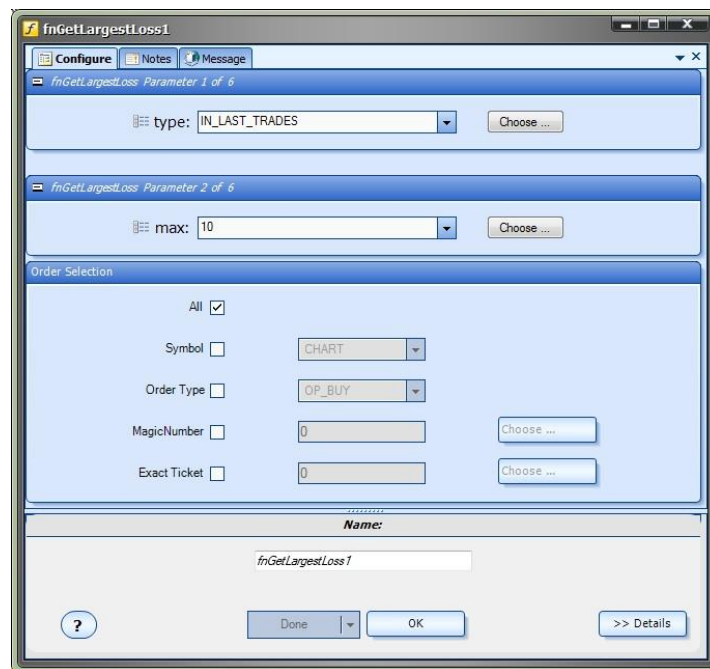
The [Function Configuration](#) window allows you to select values for each parameter.

Parameters on the **Configuration** Tab should be inspected and set each time the function is added to a drawing.

Parameters on the Configuration Tab

The following table provides information about each [parameter](#) of the **fnGetLargestLoss** function.

type	The type of trades: IN_LAST_TRADES : Considers the last max number of closed trades to calculate the largest loss. IN_LAST_MINUTES : Considers the closed trades in the last max minutes to calculate the largest loss.
max	The number of minutes or trades to consider when calculating the largest loss. The default value is 10.
Order Selection	Determines what orders of the Account are queried. All : all open orders on the account. Symbol : only open orders matching the selected symbol. Order Type : only open orders of the selected order type . MagicNumber : only open orders with a matching magicnumber . Exact Ticket : only open orders with the matching order ticket . Note : Selecting Exact Ticket disables all other selection criteria.

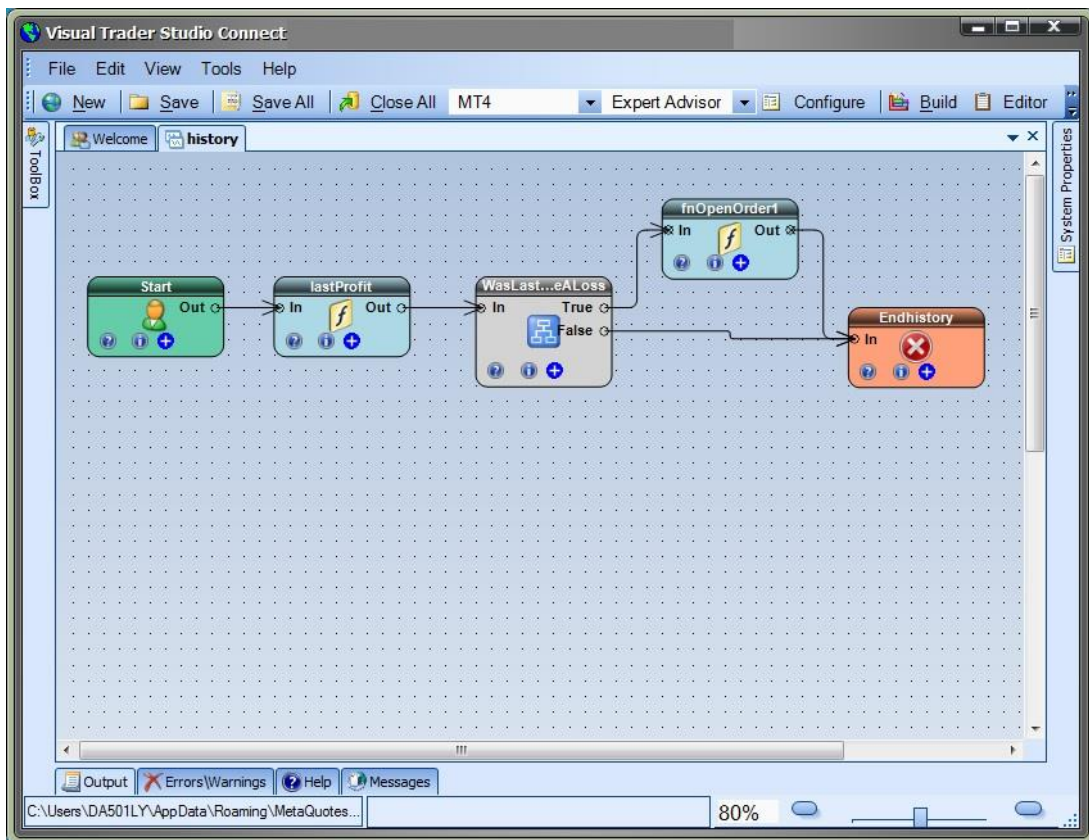


Order History on the Drawing Pad

The **Order History** functions are dragged, dropped and connected from the [Toolbox](#) onto the [Drawing Pad](#) just like any other [function](#) in VTS.

This drawing uses the [fnGetHistoryInfo](#) function, saved as **lastProfit**, to get the profit of the last closed trade.

If the last trade resulted in a loss, the [fnOpenOrder](#) function is executed to open another order.



EA Secure Plug-in

Requires VTS-Connect minimum version 4.0.0.72

Use the **EA Secure** plug-in to add a license key to your Expert Advisors: Users can not run your EA without entering a license key. With the EA Secure plug-in, you can control the account number, the type of account and the length of time that users can run your Expert Advisor.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor Builder* is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the EA Secure Plug-in

You must enter your License key to enable the **EA Secure Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

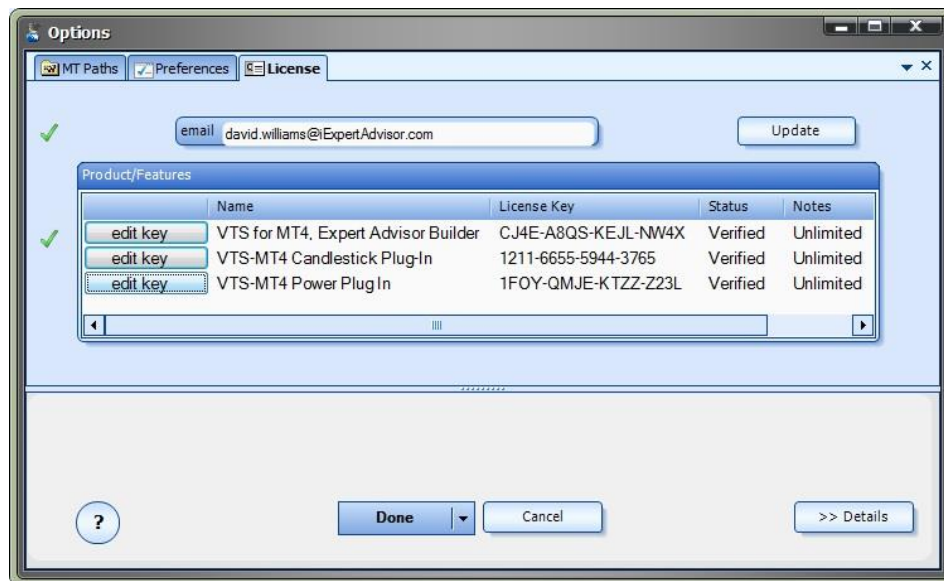
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The **email** address is the email address used to purchase [VTS](#).

The **License Key** is the key listed in the Members Area.

The **Update** button is used to verify the email address and license key.

The **edit key** button is used edit the key value.



EA Secure Plug-in

EA Secure configuration is accessed through the Security [System Manager](#).

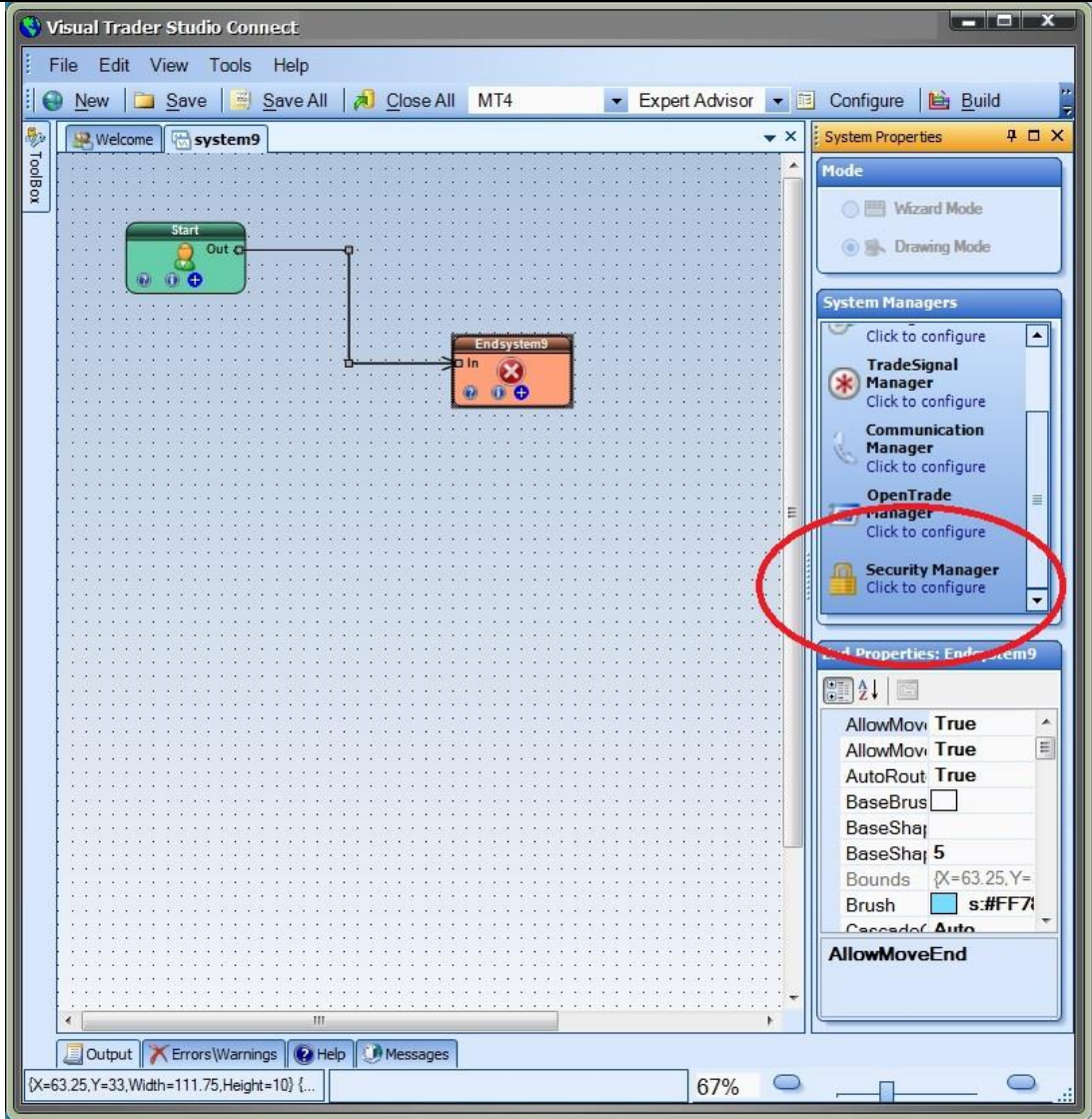
[System Managers](#) are found on the right side of the VTS main application.

Depending on your screen resolution, you may need to scroll down to view the **Security Manager** icon.

To open the **Security Manager**, double-click the icon.

Enable EA Secure	Use this checkbox to quickly enable or disable the security manager. When the security manager is disabled, the user is not required to enter a license key when starting the EA.
Use Client's MT Account Number	Use this checkbox to force the client's MetaTrader account to match the number entered in the textbox. With this feature, the EA will only run on the matching account number even if the correct secret license key is entered.
Secret Key	Enter the secret license key in this textbox. You can enter any text you like. Use English language letters and numbers only - this is an MQL limitation.

Auto-Generate Key	Use this button to auto generate a unique secret key.
Use Wildcard Key	Use this checkbox to enable wildcard functionality. The wildcard key is used to enable the EA on any account. It is typically used during testing and support.
Wildcard Key	Enter the wildcard secret license key in this textbox. You can enter any text you like. Use English language letters and numbers only - this is an MQL limitation.
Apply License To Real Account	<p>Typically, security is enabled for all account types. However, if you wish to allow users to run the EA on a demo account, the strategy tester, or even a live account, uncheck the appropriate checkbox.</p> <p>To allow the EA to always run on a <i>Real</i> account, without entering a licence key, uncheck the <i>Real Account</i> checkbox.</p>
Apply License To Demo Account	<p>Typically, security is enabled for all account types. However, if you wish to allow users to run the EA on a demo account, the strategy tester, or even a live account, uncheck the appropriate checkbox.</p> <p>To allow the EA to always run on a <i>Demo</i> account, without entering a licence key, uncheck the <i>Demo Account</i> checkbox.</p>
Apply License To Strategy Tester	<p>Typically, security is enabled for all account types. However, if you wish to allow users to run the EA on a demo account, the strategy tester, or even a live account, uncheck the appropriate checkbox.</p> <p>To allow the EA to always run on the <i>Strategy Tester</i>, without entering a licence key, uncheck the <i>Strategy Tester</i> checkbox.</p>
Use Expiration Date	Use this checkbox to add an expiration date to the license.
DateTime Picker	Use the date-time picker to select an expiration date. After the expiration date, the EA will fail to load and display a license failure message.
License Failure Message	<p>This is the message that is displayed on the price chart when the license key fails for any reason.</p> <p>You can enter any text you like. Use English language letters and numbers only - this is an MQL limitation.</p>



EA Secure Configuration

To open the **EA Secure** configuration, double-click [the Security System Manager](#).

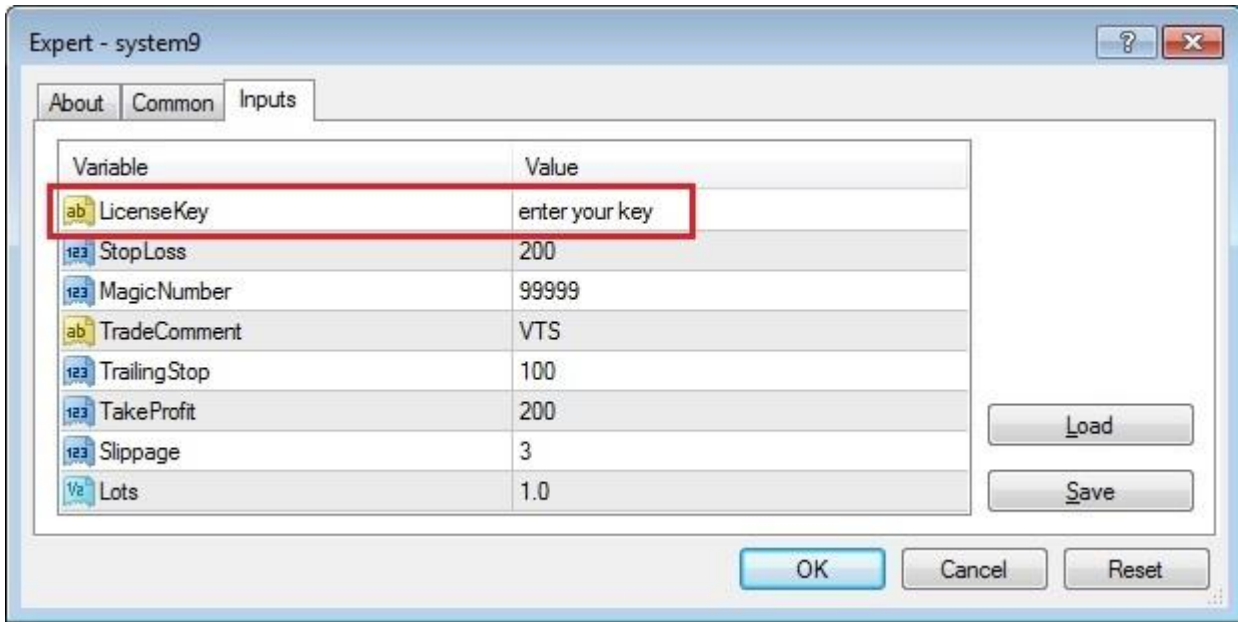
When the EA Secure feature is enabled, users are forced to enter a license key to run the Expert Advisor.

The following table provides information about each option of [the Security System Manager](#).

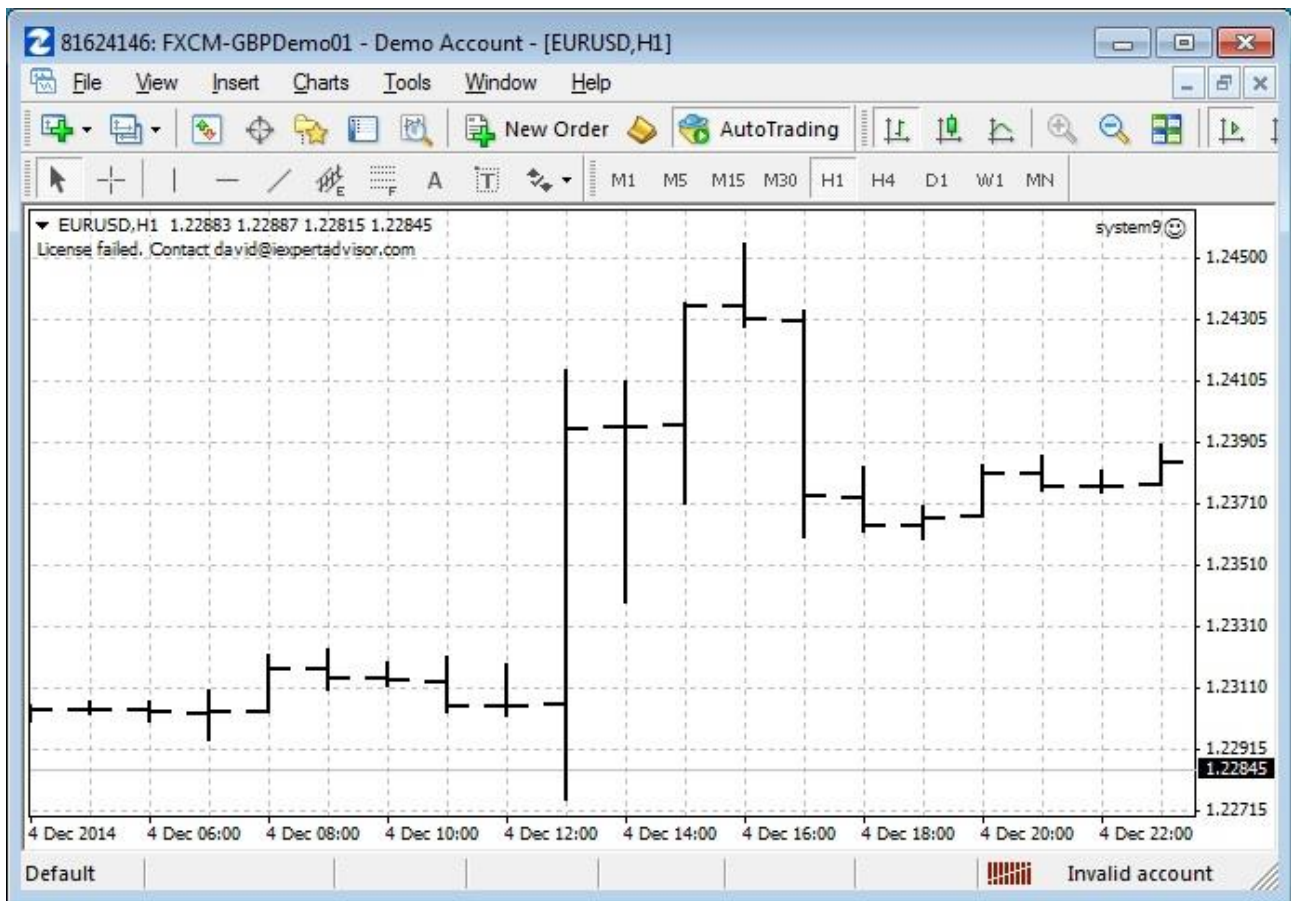
The screenshot shows the 'Security (Manager type Security)' configuration window. It features a 'Configure' tab and a checked 'Enable EA Secure' option. The 'Create Secret Key' section includes a checked 'Use Client's MT Account Number' checkbox with the value '12345678' and a text field containing '27247628', along with an 'Auto-Generate key' button. The 'Create Wildcard Key' section has a checked 'Use Wildcard Key' checkbox and a text field with 'wildcard5'. The 'Apply License To' section contains three checked checkboxes: 'Real Account', 'Demo Account', and 'Strategy Tester'. The 'License Expiration' section has a checked 'Use Expiration Date' checkbox and a date dropdown set to 'Thursday, November 19, 2015'. The 'License Failure Message' section contains a text area with the message 'License failed. Contact david@expertadvisor.com'. At the bottom, there is a 'Name:' label, a text field with 'Security', and buttons for '?', 'Done', 'Cancel', and '>> Details'.

EA Secure On Price Charts

When the user first attaches the EA to a price chart, the first Input is the **LicenseKey**. The user must enter the secret license key or the EA will not run.



If the license key fails validation for any reason, the [License Failure Message](#) is displayed on the price chart and the EA exits on each tick.



If the license key passes validation and an [Expiration Date](#) is configured, the expiration date is displayed on the price chart on each tick. If the [Expiration Date](#) is not configured and the license key passes validation, no message is shown and the EA executes normally.

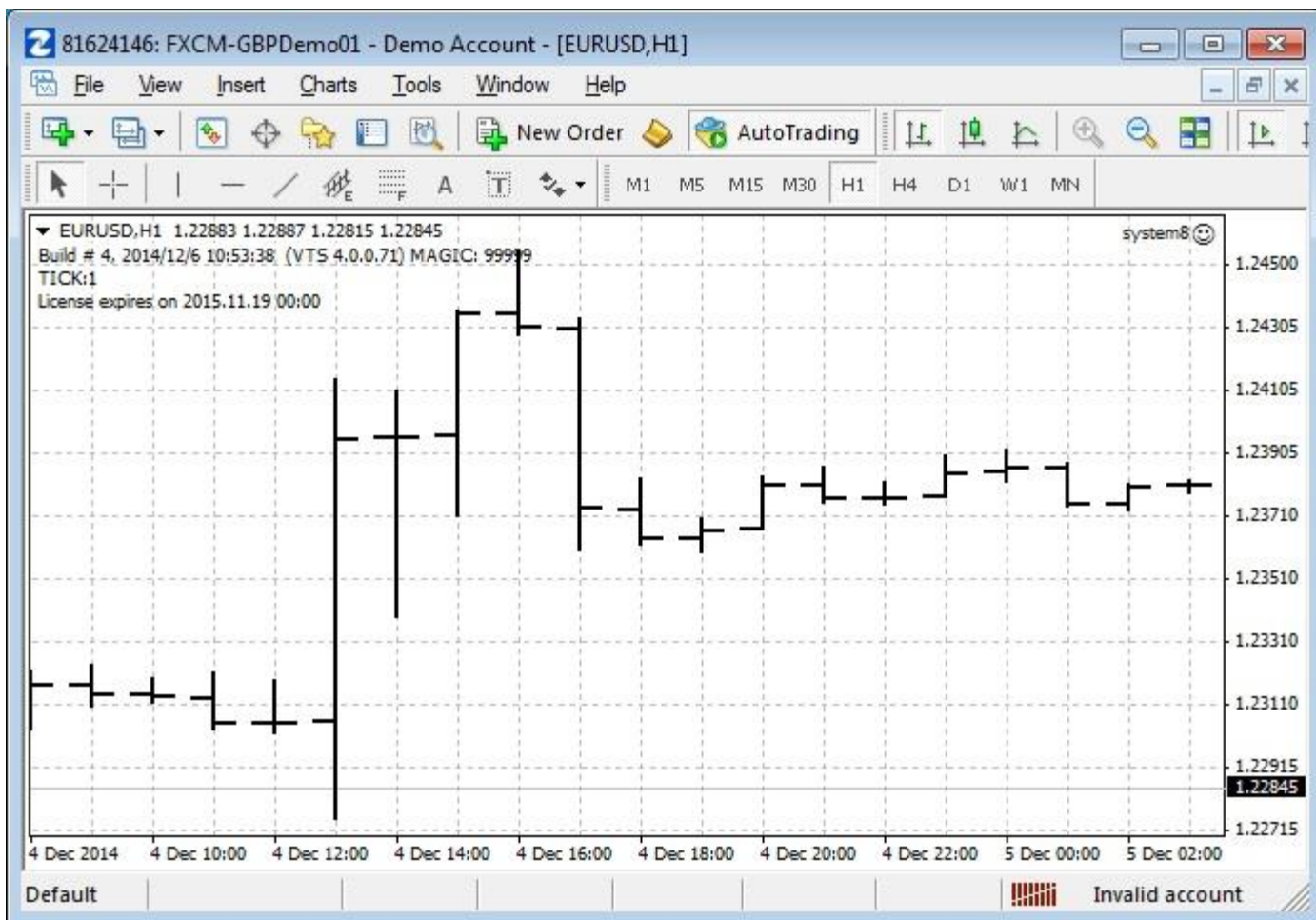


Chart Execute Plug-in

Requires VTS-Connect minimum version 4.0.0.73

Use the **Chart Execute** plug-in to *execute* (or *run*) any MQL function within your Expert Advisor directly on the price chart. When you enable the *Chart Execute* feature in your VTS EA and attach the EA to a MetaTrader price chart, a button is drawn on the chart. When you click the button, a window is displayed that allows you to choose any function (and its configurable parameters) and execute that function just once or on every price change.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor Builder* is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the Chart Execute Plug-in

You must enter your License key to enable the **Chart Execute Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The **email** address is the email address used to purchase [VTS](#).

The **License Key** is the key listed in the Members Area.

The **Update** button is used to verify the email address and license key.

The **edit key** button is used edit the key value.



Chart Execute in the System Manager

Chart Execute configuration is accessed through the **Chart Execute System Manager**.

System Managers are found on the right side of the VTS main application.

Depending on your screen resolution, you may need to scroll down to view the **Chart Execute Manager** icon.

To open the **Chart Execute Manager**, double-click the icon.

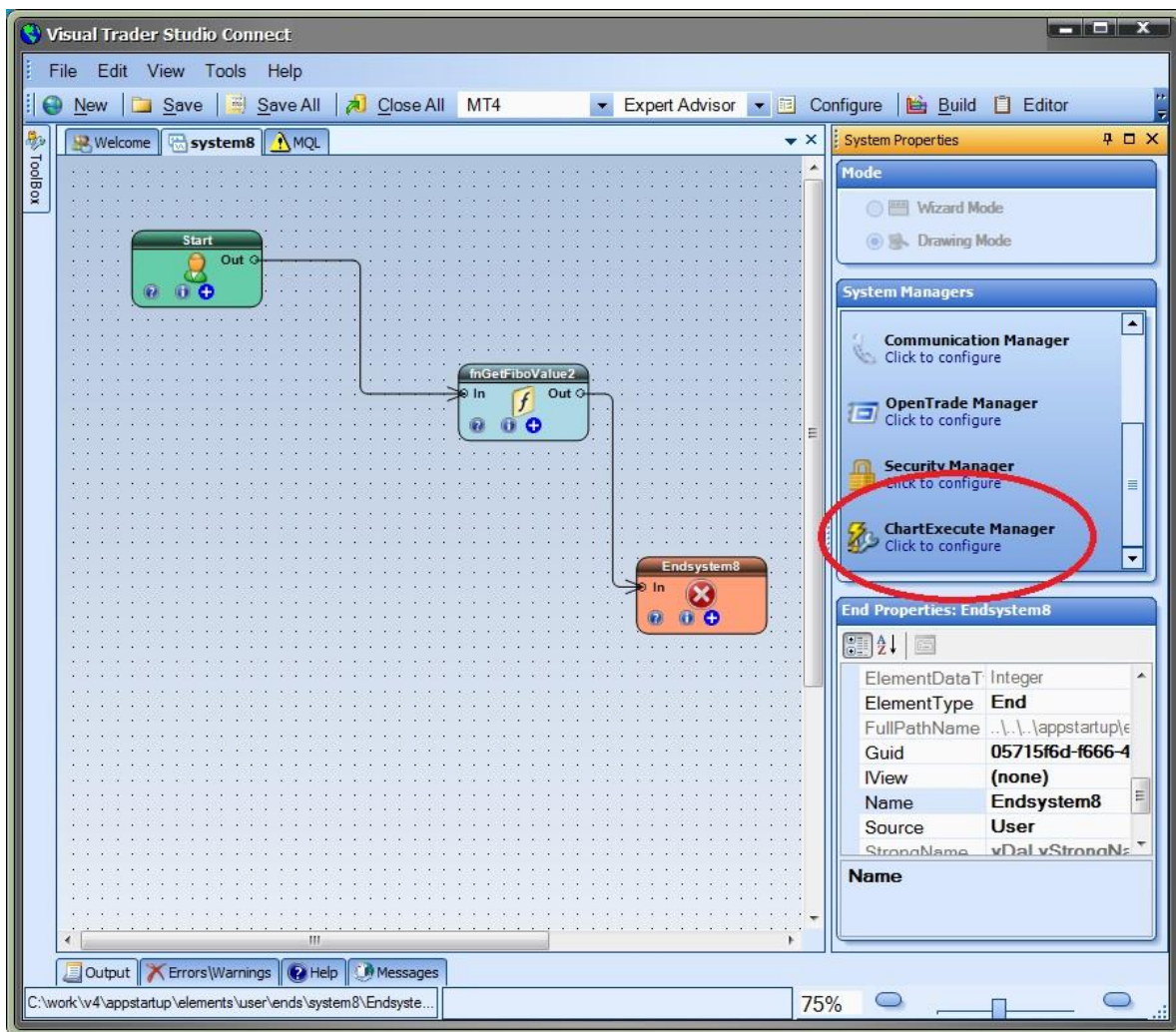


Chart Execute Configuration

To open the **Chart Execution** configuration window, double-click the [Chart Execute Manager](#).

To enable **Chart Execution**, check the *Enable Chart Execute* checkbox.

After **Chart Execution** has been enabled, individual functions are selected by checking the *Show* checkbox.

Any checked functions will appear on the *Chart Execute Window* when the *Chart Execute Button* is clicked on the MetaTrader price chart.

If **Chart Execution** is not enabled or no functions are selected, the *Chart Execute Button* will not be displayed on the MetaTrader price chart.

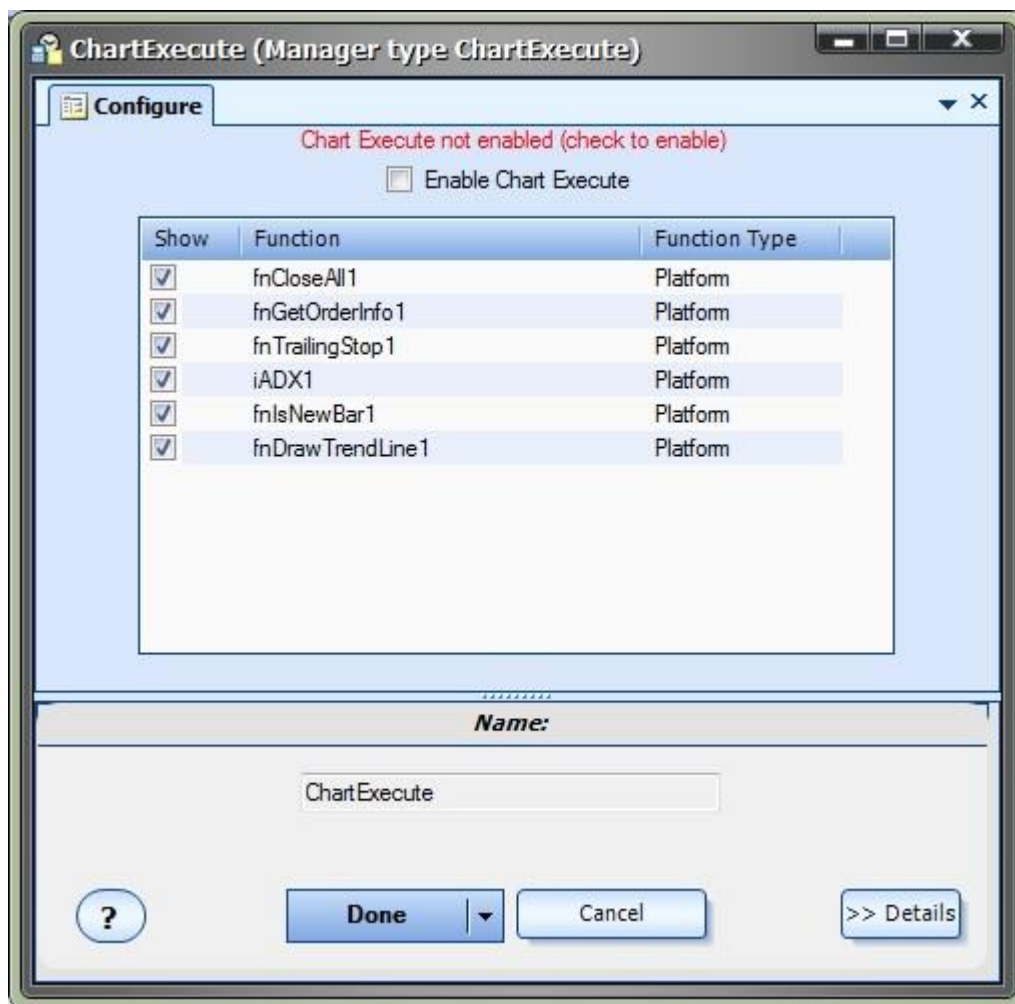
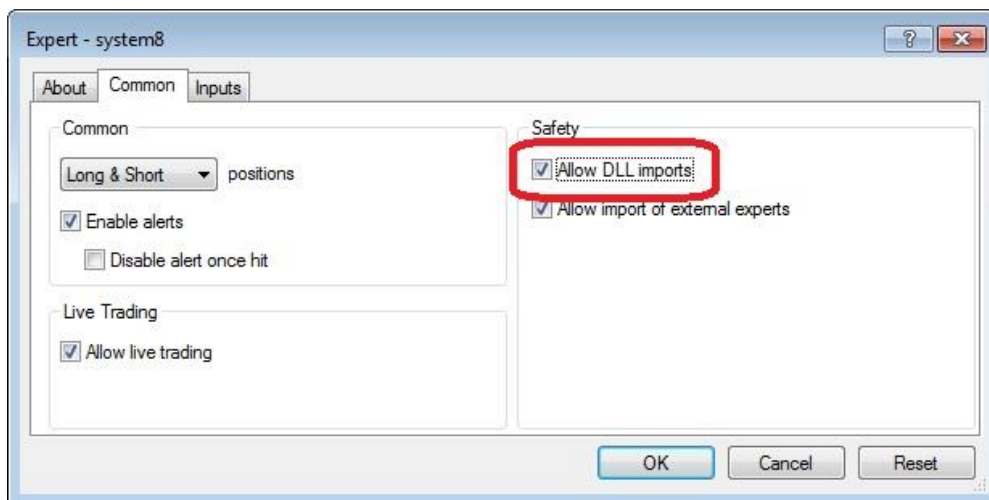


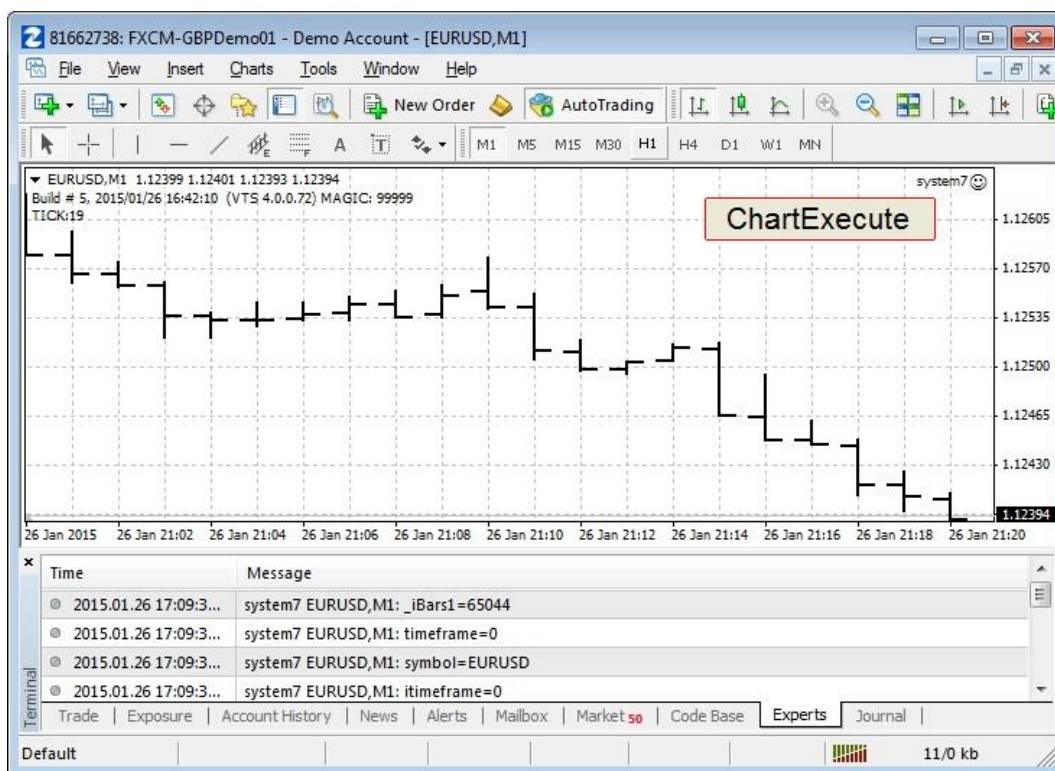
Chart Execute On Price Charts

When an EA with **Chart Execute** enabled is attached to a MetaTrader price chart, the "Chart Execute" button is displayed on the top right corner of the MetaTrader price chart.

NOTE: When attaching the EA to the chart, DLL's imports must be allowed. On the *Common* tab, check the box **Allow DLL imports**.



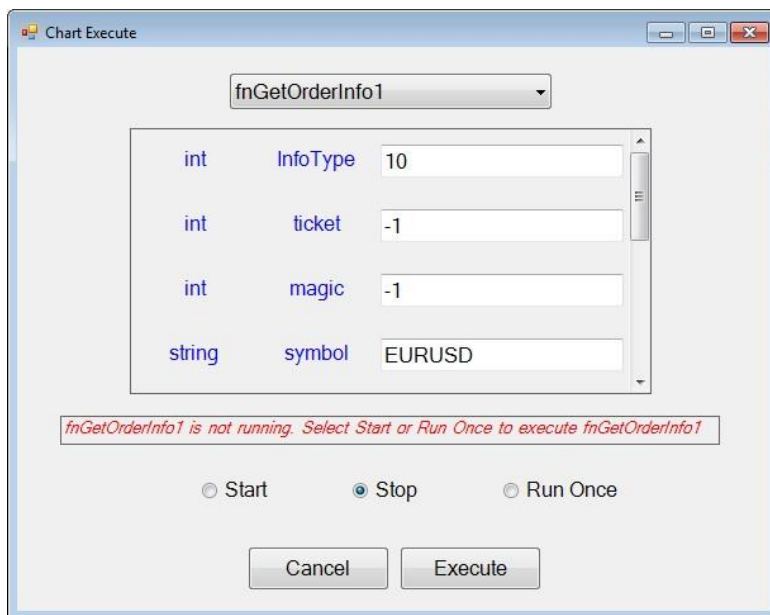
The "Chart Execute" button is displayed in the top right corner of the price chart.



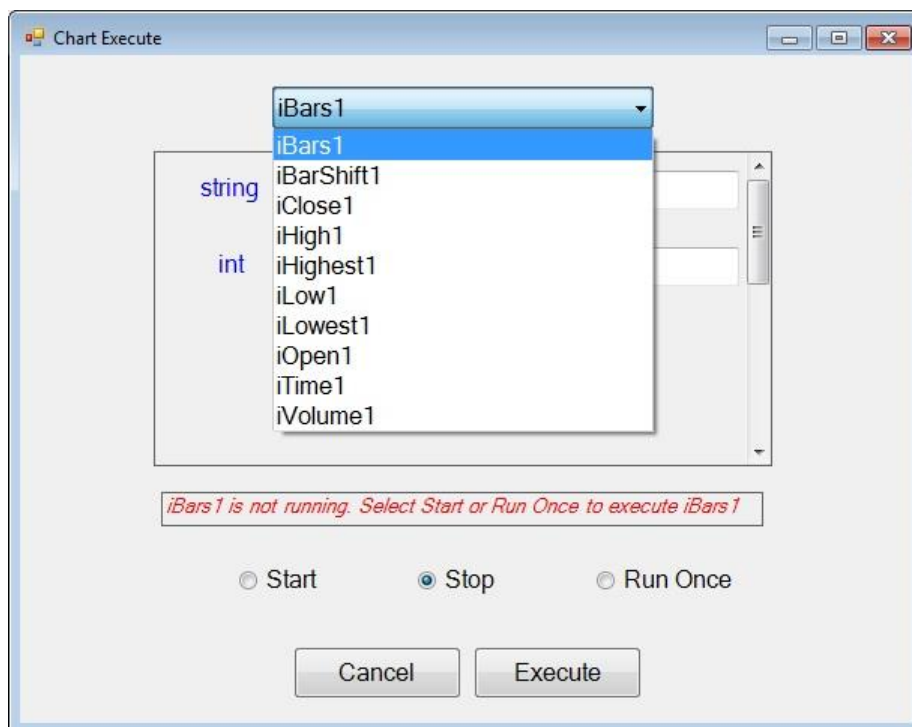
Using Chart Execute

When an EA has been [configured to use Chart Execute](#) and the EA has been attached to a MetaTrader price chart, the ["Chart Execute" button is displayed](#) in the right hand corner of the price chart.

Clicking the "Chart Execute" button will display the following window:

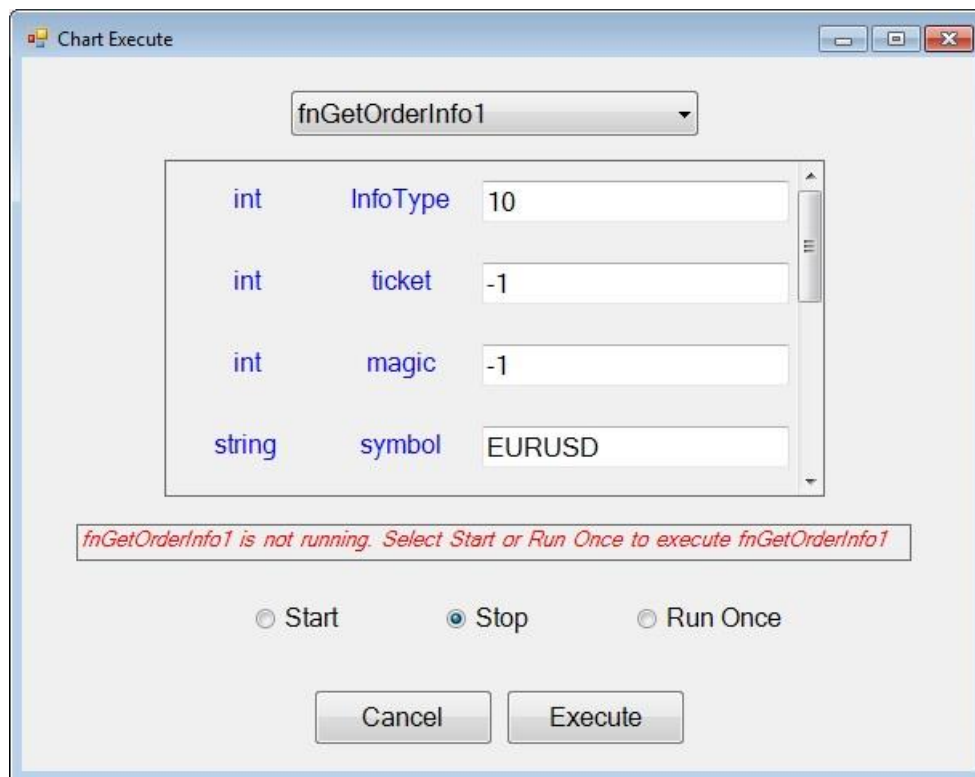


The pull-down menu at the top shows all functions of the EA that have been checked (to "show") in [VTS Chart Execute configuration](#) window:



When a function is selected, the parameters for that function are displayed and are available to be edited.

Note: Some functions do not have any parameters, in this case the parameter list is blank.



There are 3 execute **options** available:

The **Start** option will execute the selected function on every incoming price change (every *tick*). The function will continue to execute until it is *Stopped* or the EA is detached from the chart.

The **Stop** option will stop a function that is executing on every tick.

The **Run Once** option will execute the function just once. The function executes immediately when the *Execute* button is clicked.

The window is closed when **Cancel** or **Execute** is clicked.

When **Cancel** is clicked, no action takes place. The window simply closes.

When **Execute** is clicked, the selected option (Start, Stop, Run Once) is executed.

Notes:

Only one function can be managed at a time. To start each function, the "Chart Execute" button must be selected and used to start (or stop) each function individually.

Selecting **Start** on an already started function has no effect.

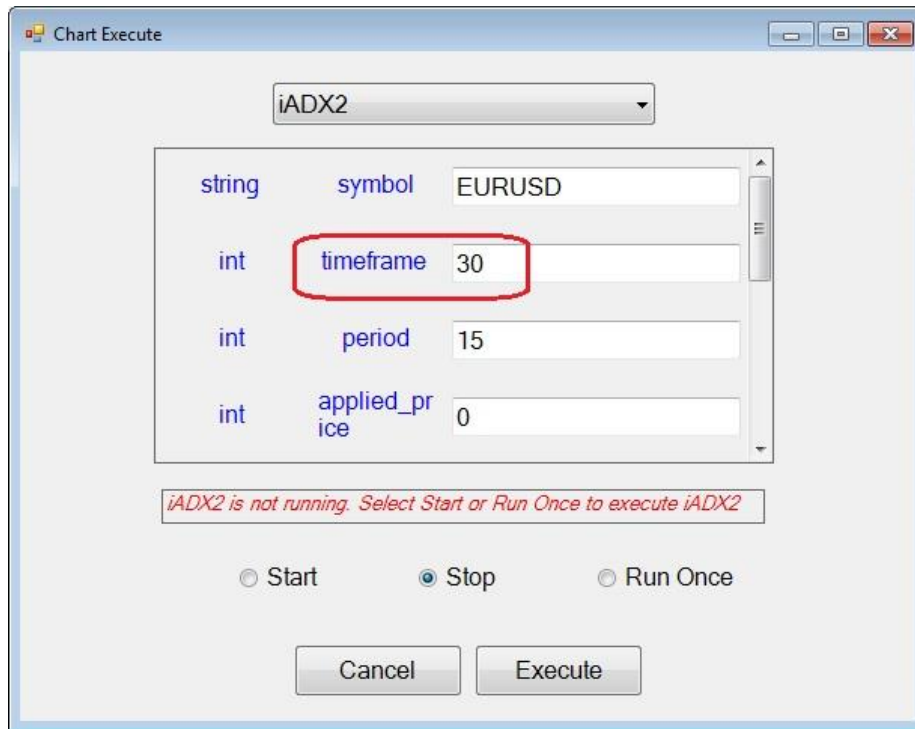
Selecting **Stop** on an already stopped function has no effect.

Chart Execute Notes

Function Parameter Values

The parameter values that appear in the "Chart Execute" window may look different than the parameter values you see in VTS. The values will often appear as their actual numerical value.

For example, for the **timeframe** parameter, instead of the value PERIOD_M30, you will see the value 30. If you are not familiar with the actual values of many of the MQL predefined variables (and most traders are not), it may be better to configure the function in VTS and run the function with the same values in "Chart Execute". By default, the parameters values in "Chart Execute" exactly match the parameter values in VTS, so if you don't change them, they will be the same.



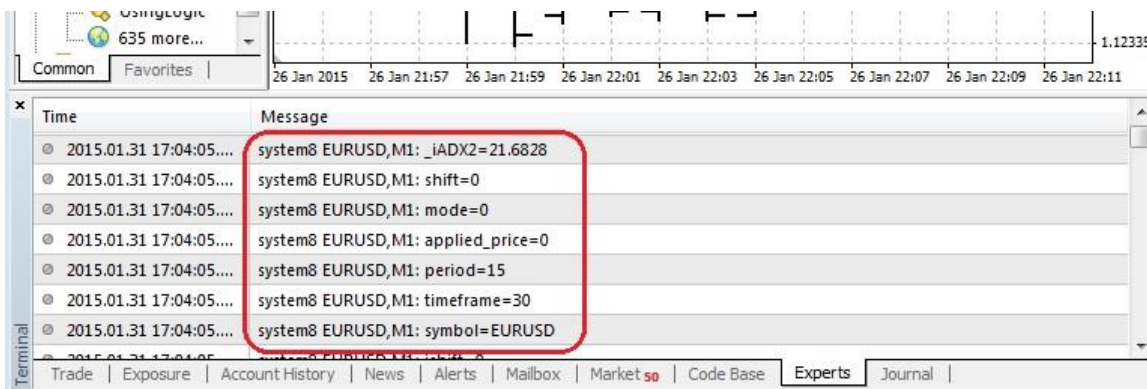
Remember, VTS is full-featured Windows application and is able to present MQL predefined variables as though they are part of VTS. "Chart Execute" is a light-weight application that is built on the fly when you build your EA, therefore the presentation is not as rich.

Function Testing

Use the "Run Once" option to test your functions for the first time.

When the function is executed, all of the parameters values and the return value of the function are displayed in the "Experts" Tab on the bottom of your MetaTrader terminal window.

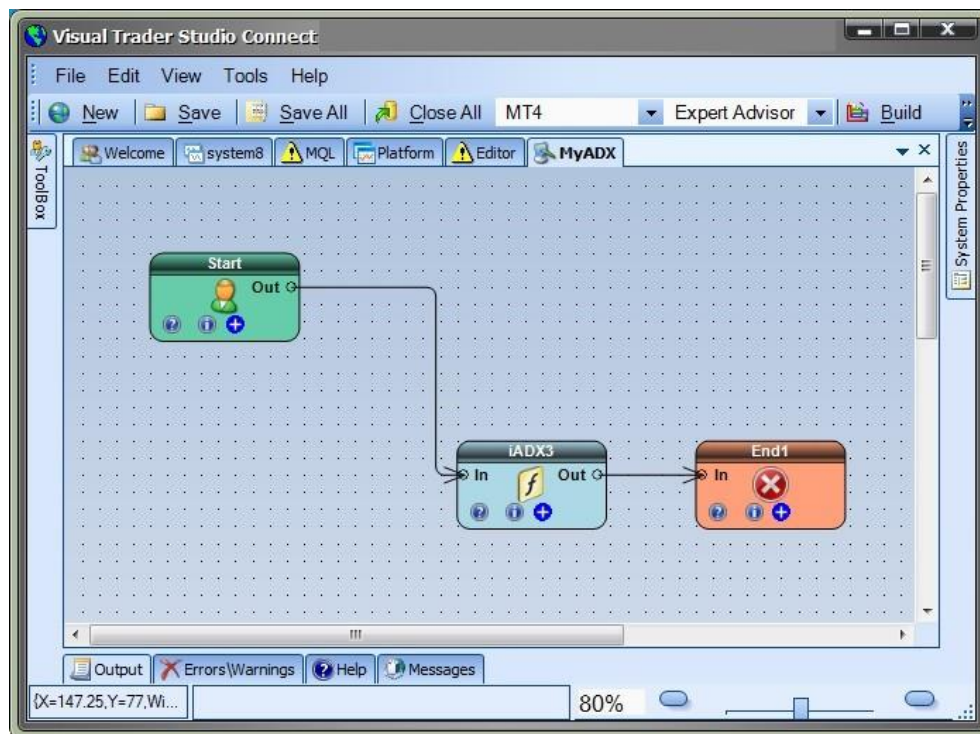
(Note: If the terminal window does not appear on the bottom of your MetaTrader platform, in your MT platform, go to View.>Terminal).



Using a VTS Drawing to Hide Function Parameters

Many times you may not want to change any parameters of a function, or perhaps just one or two parameters, but viewing them *all* is a distraction.

One method to hide the function parameters, so they do not become a distraction, is to create a new VTS [Drawing](#) with just a Start [Element](#), the function you want to execute and an [End](#) Element.

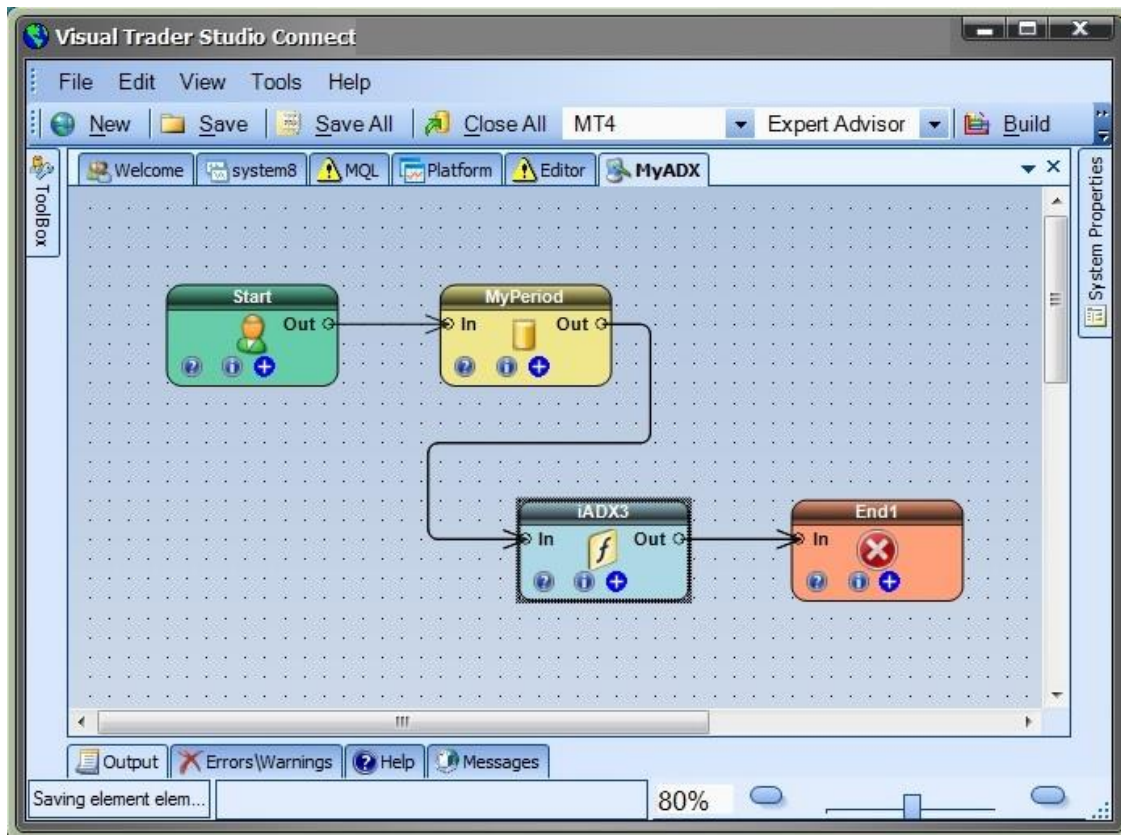


Once the drawing is attached (see [Connecting Functions on a Drawing](#) below), and [enabled in Chart Execute](#), the function "MyAdx" (the name of the drawing) will appear in the "Chart Execute" function list. And the "MyAdx" function will not have any parameters. It will simply execute the drawing above, including iADX3, with the parameters that are defined in the iADX3 function.

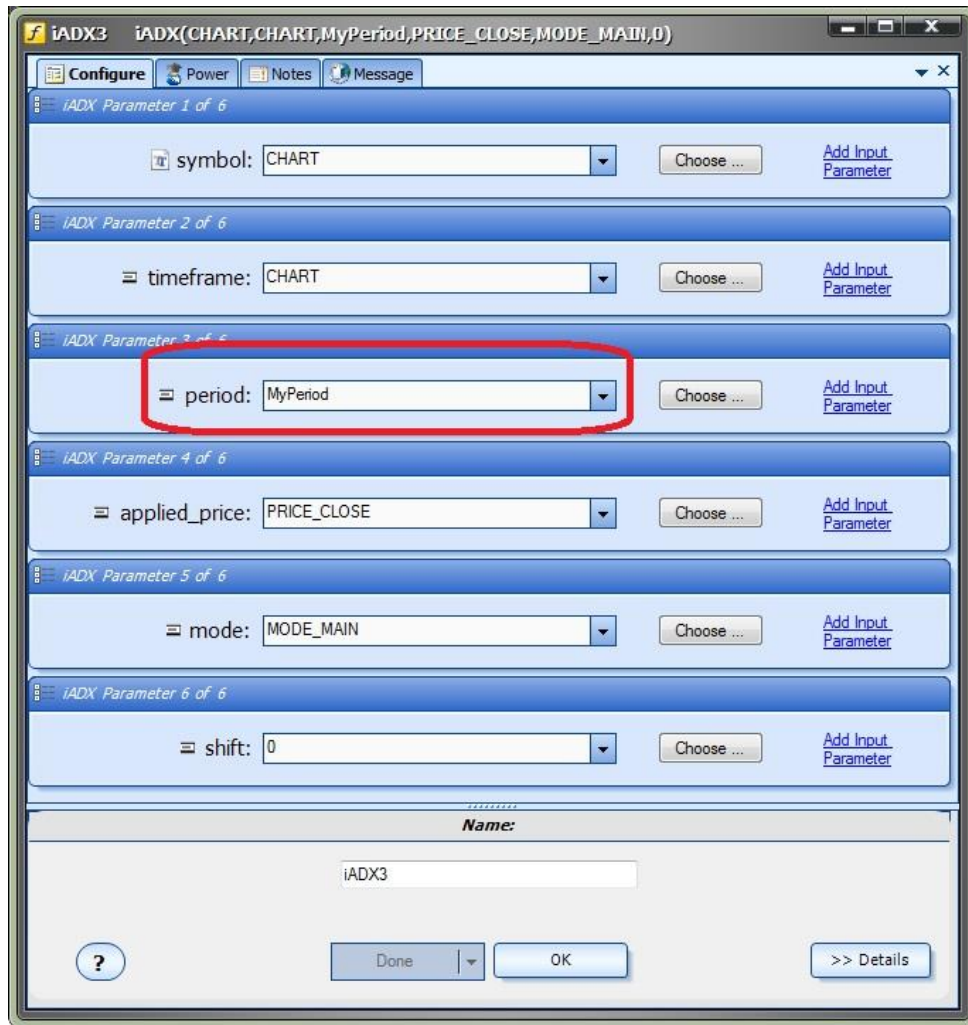
If you know you are always going to execute a function with the same parameters, this is an excellent way to insure no mistakes are made when the function is executed.

If you want the ability to change just one (or more) parameters of the iADX function, this can be done by adding a [Variable](#) Element (of [scope](#) Parameter) to the drawing, and the using the variable in the ADX function.

For example, create a [Variable](#) named "MyPeriod", set the scope to [parameter](#) and add it to the [Drawing](#).



When configuring the ADX function, set the **period** Parameter to "MyPeriod".



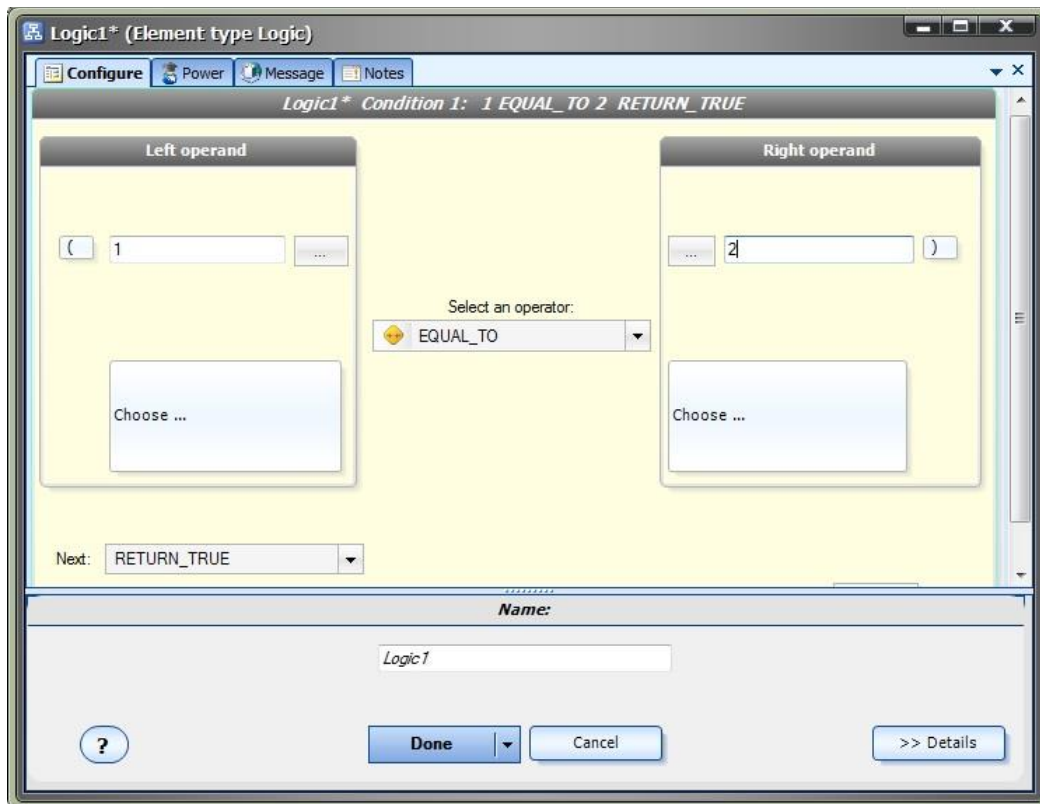
Using this method, when the "MyAdx" function is displayed in "Chart Execute", the only parameter to appear will be "MyPeriod".

Connecting Functions on a Drawing

In order for a function to be displayed in "Chart Execute", it must be connected on a Drawing in the VTS System.

There may be a situation where you want a function to be available in "Chart Execute", but the function is not needed in your EA. For this situation, you can connect the function an output of a [Logic](#) Element that will never evaluate to true, and therefore will never execute within the EA.

For example, create a Logic Element with a logic condition of "1 is equal to 2". This condition will **never** evaluate to true.

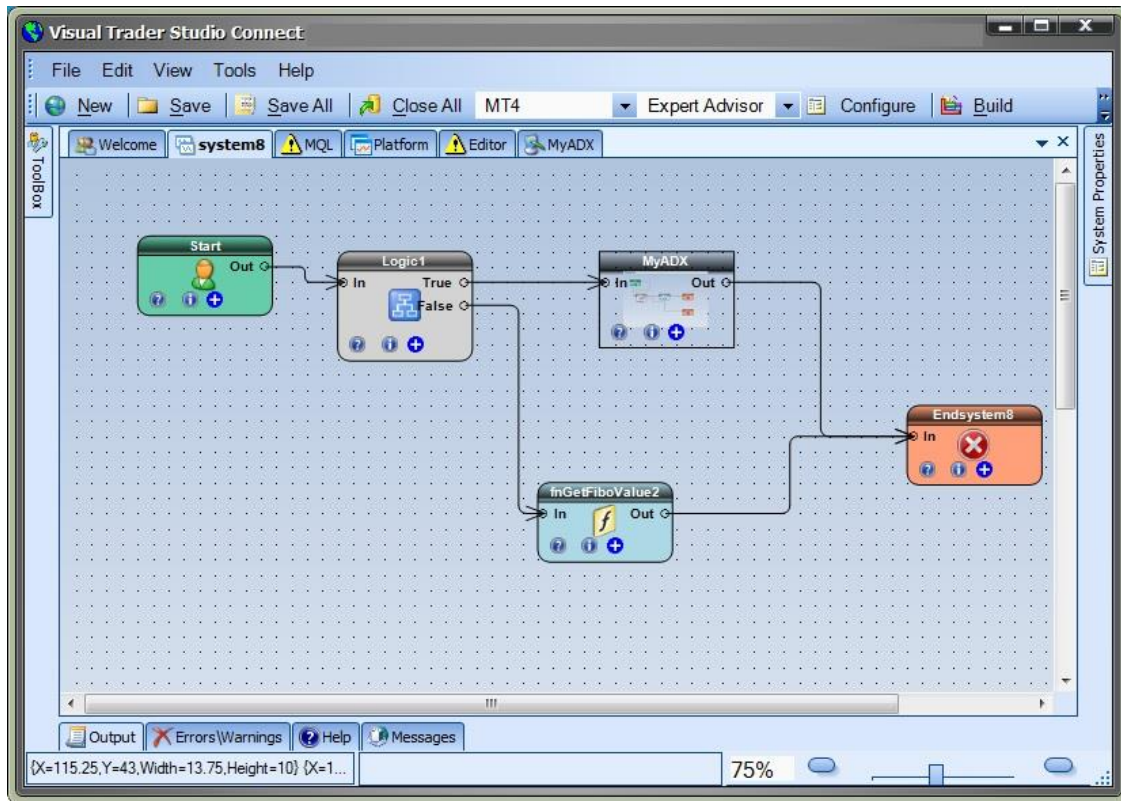


Connect the function to be used only by "Chart Execute" to the **True** output of the Logic Element. The rest of the (useful) Elements required by the EA can be connected to the **False** output, where they will execute normally when the EA runs.

In the below [Drawing](#), the MyADX function is connected to the **True** output of Logic1. The only condition within Logic1 is:

"1 is equal to 2"

which will never execute within the EA, but the MyADX function will be available to execute in "Chart Execute".



This technique of creating Drawing functions to hide the parameters is especially useful for trading operations such as opening, closing and managing open trades. The MQL functions [Ordersend](#), [OrderModify](#) and [OrderClose](#) have many parameters that most traders do not have a need to change when executing.

Money Manager Plug-in

Requires VTS-Connect minimum version 4.0.0.75

Optimize the profit potential of your EA by automatically selecting the risk-appropriate lot size based on your current account balance. The Money Manager allows you to generate lot size strategies where you define your level of allowed risk and your maximum loss. Fixed, Fixed Percentage and Fixed Ratios methods are available as well as a Monte Carlo simulation engine that allows you to view the likely outcome of your money management strategy.

The Money Manager Plug-in is available as a VTS [System Manager](#) and can be used with any VTS-generated Expert Advisor.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor Builder* is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the Money Manager Plug-in

You must enter your License key to enable the **Money Manager Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

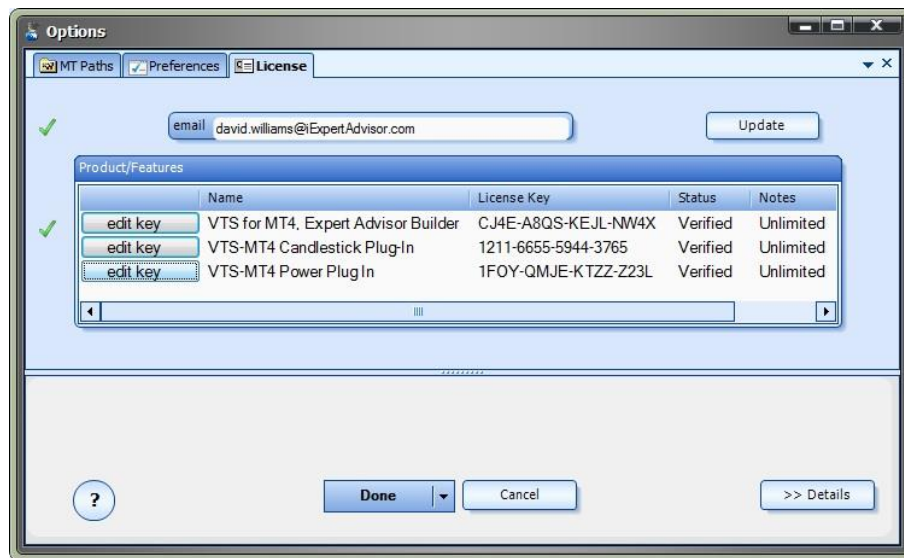
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The **email** address is the email address used to purchase [VTS](#).

The **License Key** is the key listed in the Members Area.

The **Update** button is used to verify the email address and license key.

The **edit key** button is used edit the key value.



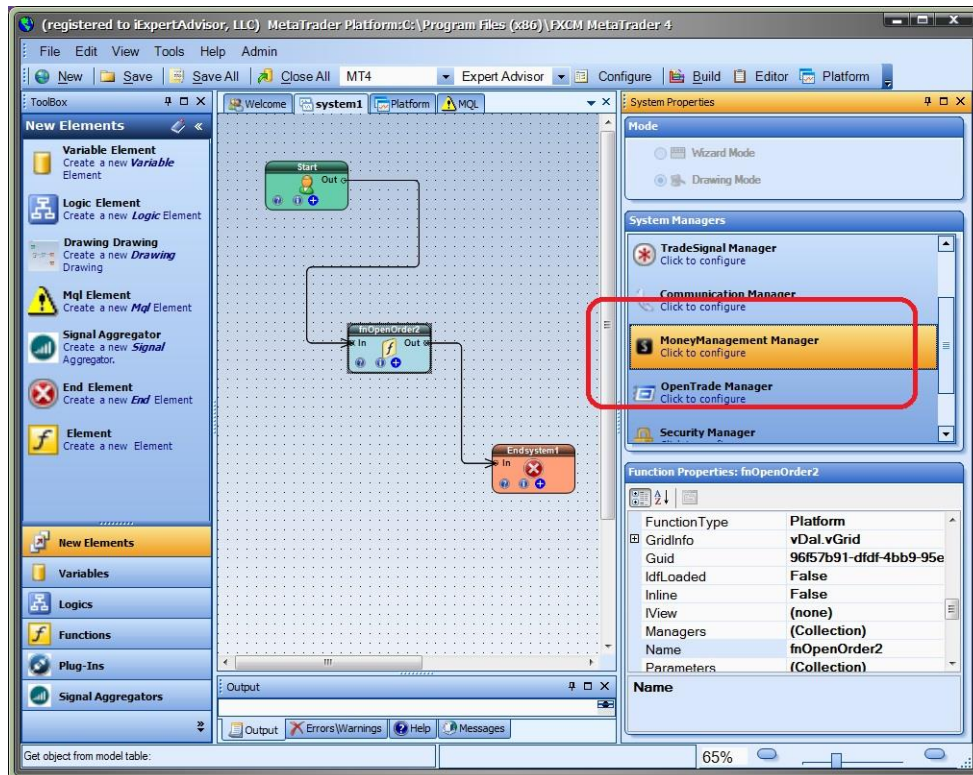
Money Manager in the System Manager

Money Manager configuration is accessed through the **Money Manager System Manager**.

[System Managers](#) are found on the right side of the VTS main application.

Depending on your screen resolution, you may need to scroll down to view the **Money Manager** icon.

To open the **Money Manager**, double-click the icon.



Money Manager Configuration

Money Manager configuration is accessed through the **Money Manager System Manager**.

There are two tabs available: Configure and Simulate.

The **Configure** area is used to define the Money Manager parameters

The **Simulate** area is used to generate Monte Carlo simulations to view the likely equity curve of the strategy.

Configure Parameters

Section	Parameter Name	Value	Description
Account			
	Account Type	Micro, Mini or Standard	The account type will effect the value of one point as well as the minimum and maximum lot sizes. For reference, the value of one point on a EURUSD chart is: Standard: \$10.00 Mini: \$1.00 Micro: \$0.10
	Start Balance	Any value greater than 0	The starting balance. This value, along with the stoploss, is used to determine the starting lot size.
Trade			
	Stoploss	Any value greater than 0 Any VTS Variable	The stoploss is used to define the maximum lot size for a given risk value. On a per trade basis, the stoploss is viewed as the maximum allowable loss.
	TakeProfit	Any value greater than 0 Any VTS Variable	The takeprofit is only used during simulation to apply profitable trades to the account balance. The takeprofit is not used to calculate lot size.
Money Management			
	Type	Fixed Fixed Percent Fixed Ratio	Fixed uses the same lot size for any account balance. Fixed Percent calculates the lot size using the stoploss (to define maximum allowable loss for a risk percentage) and the current balance. Fixed Ratio calculates the lot size using the delta* value.
	Risk % or Delta	Any value greater than 0	The percent amount of the current balance that can be lost on a single trade, or the delta* value.

* The **Fixed Ratio** method is further defined in our ebook **Automatic Alpha**, by David Williams. Here is an excerpt:

The Fixed Ratio method allows the account to grow aggressively in the beginning, assuming more risk, and gradually reduces risk as the account size grows. The main principle behind Fixed Ratio is that the relationship between the number of lots traded and the amount of profit needed to increase the number of lots traded remains fixed.

For example, if a trader's money management method is to trade one (full-sized) lot and requires \$10,000 USD of profit to increase to (2) two lots, then the Fixed Ratio method would required an additional \$20,000 USD of profit to increase to the next level, that is, to (3) three lots.

The ratio between the number of lots traded and the amount of profit required to increase

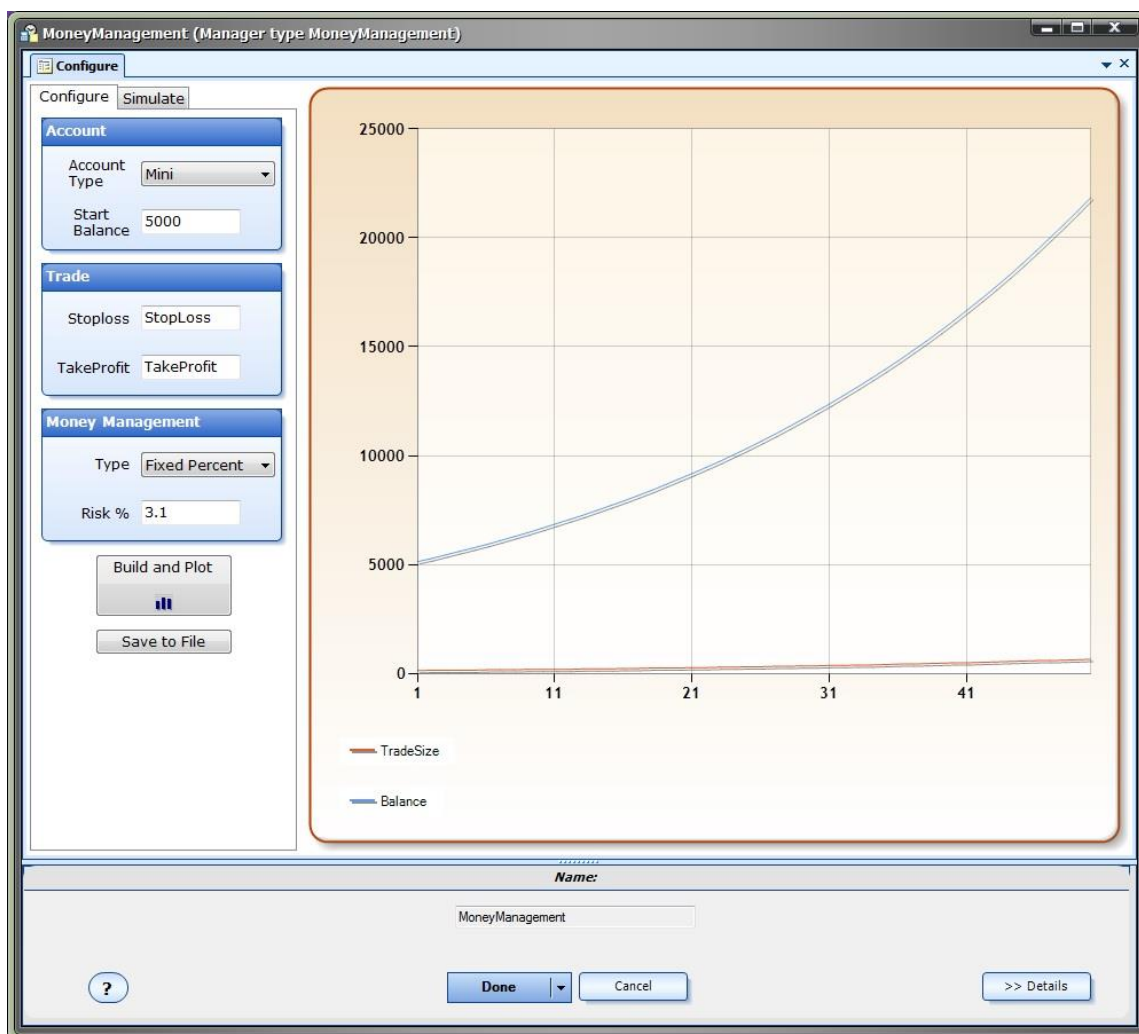
the number of lots is fixed. In Jones' method, the amount of profit required to increase the number of lots traded is referred to as Delta. Once a trader chooses a value of Delta appropriate for their risk tolerance, the entire money management scheme can be fully described.

For example, suppose a trader opens a (mini) account with \$2000 USD.
 The trader chooses to begin trading with a lot size of (1) mini-lot. (One mini-lot is the minimum lot size that can be traded).
 The trader chooses a Delta value of \$200 USD.

The account level at which the trader increases their lot size to (2) lots is \$2,200 USD.
 The account level at which the trader increases their lot size to (3) lots is \$2,600 USD.
 The account level at which the trader increases their lot size to (4) lots is \$3,200 USD.

This Fixed Ratio relationship can be expressed as:
 The New Account Level to Increase Trade Size by One Lot =
 Starting Account Value + (Delta x Number of Current Lots)

For more information, read **The Trading Game**, by Ryan Jones. He is the original developer of the Fixed Ratio method.



To generate an equity curve, click the **"Build and Plot"** button.

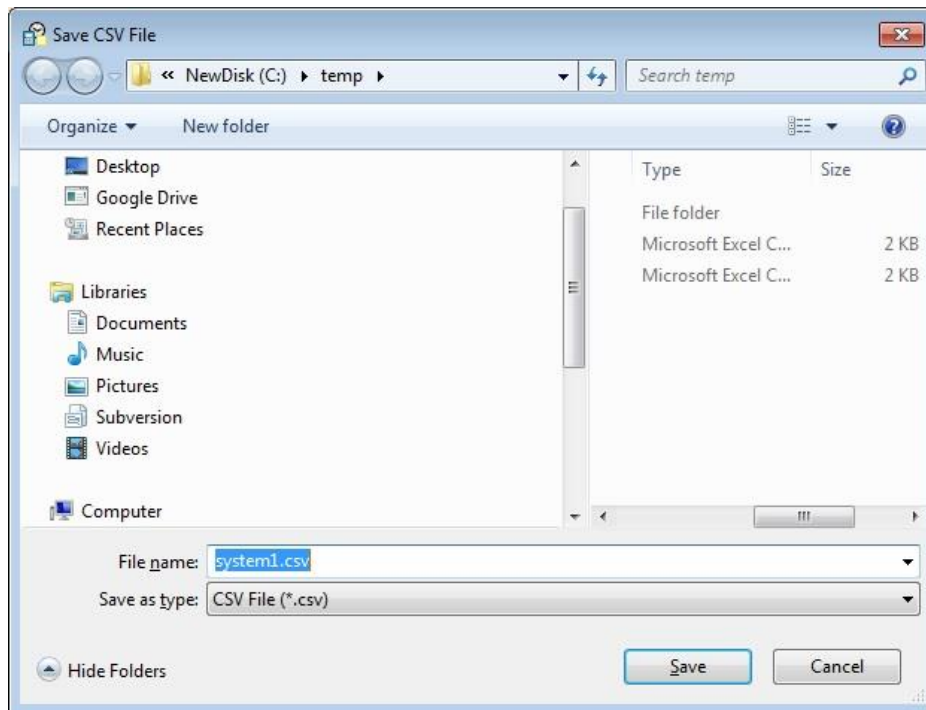
This plots the equity using winning trades only.

By default, the result of 50 trades is shown. The purpose of the equity graph is to display the *relative strength* of different money management configurations.

The **"Build and Plot"** button should be clicked after any changes have been made to any of the parameters.

A table is built that assembles the lot size for each level of the account balance.

This table can be saved as a Comma Separated File (CSV) by clicking the "Save to File" button:



Simulate Parameters

Parameter Name	Value	Description
Start Balance	Any value greater than 0	The starting balance
Number of Trades	Any value greater than 0	The number of trades to simulate*
Number of Samples	Any value greater than 0	The number of times to run the simulation*
Winning Percent	A value between 0 and 100	The winning percent of the trading system

Theory

The Monte Carlo simulation uses the winning percentage to define exactly how many trades will be profitable from the value entered in the 'Number of Trades'.

The TakeProfit value from the configuration tab is used to define the amount of a winning trade.

The StopLoss value from the configuration tab is used to define the amount of a losing trade.

Each sample distributes the wins and losses randomly throughout the number of trades. The final plot shows the average of all sampled plots.

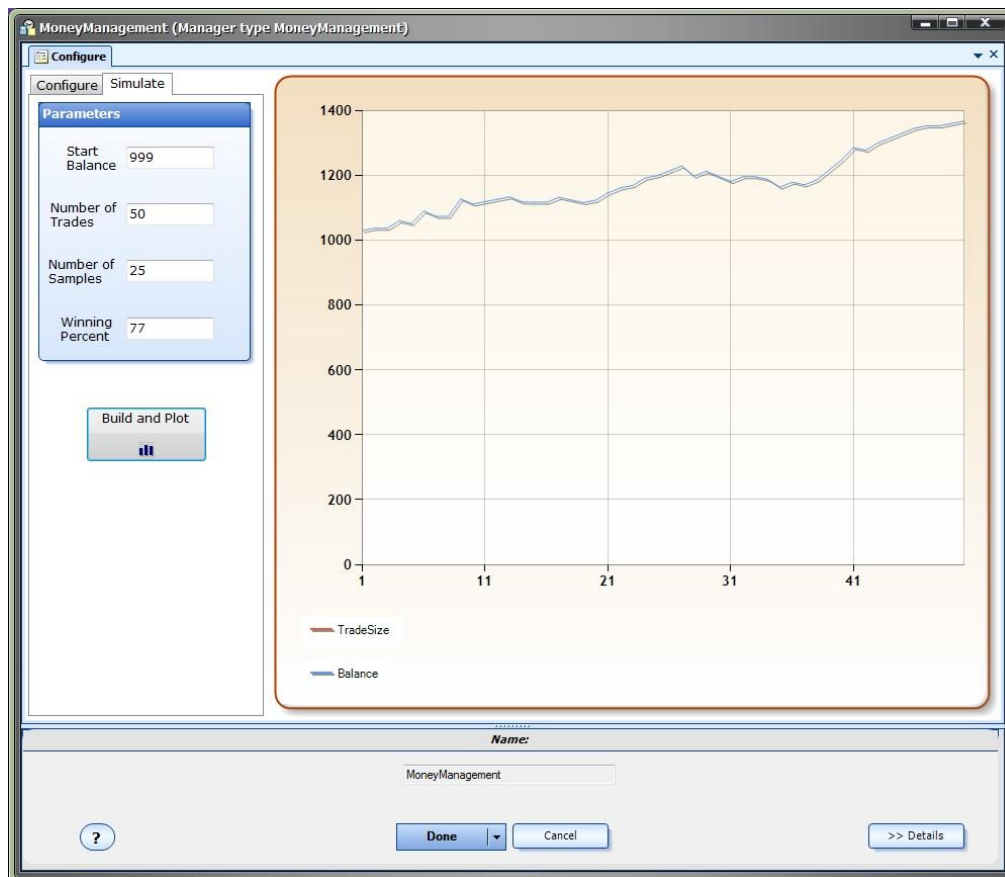
Each time you click the "Build and Plot" button, the equity plot is redrawn. It should look different each time since the distribution of wins and losses is random.

It is useful to set the **Number of Trades** to 1 and click the **Build and Plot** button several times: This displays the equity curve of a single run of the Expert Advisor, which is what most traders run.

Monte Carlo simulation is useful when you know the winning percentage of your Expert Advisor, but you do not know the exact order of the wins and losses.

*Note: Large values of **Number of Trades** and/or **Number of Samples** may result in intensive CPU utilization.

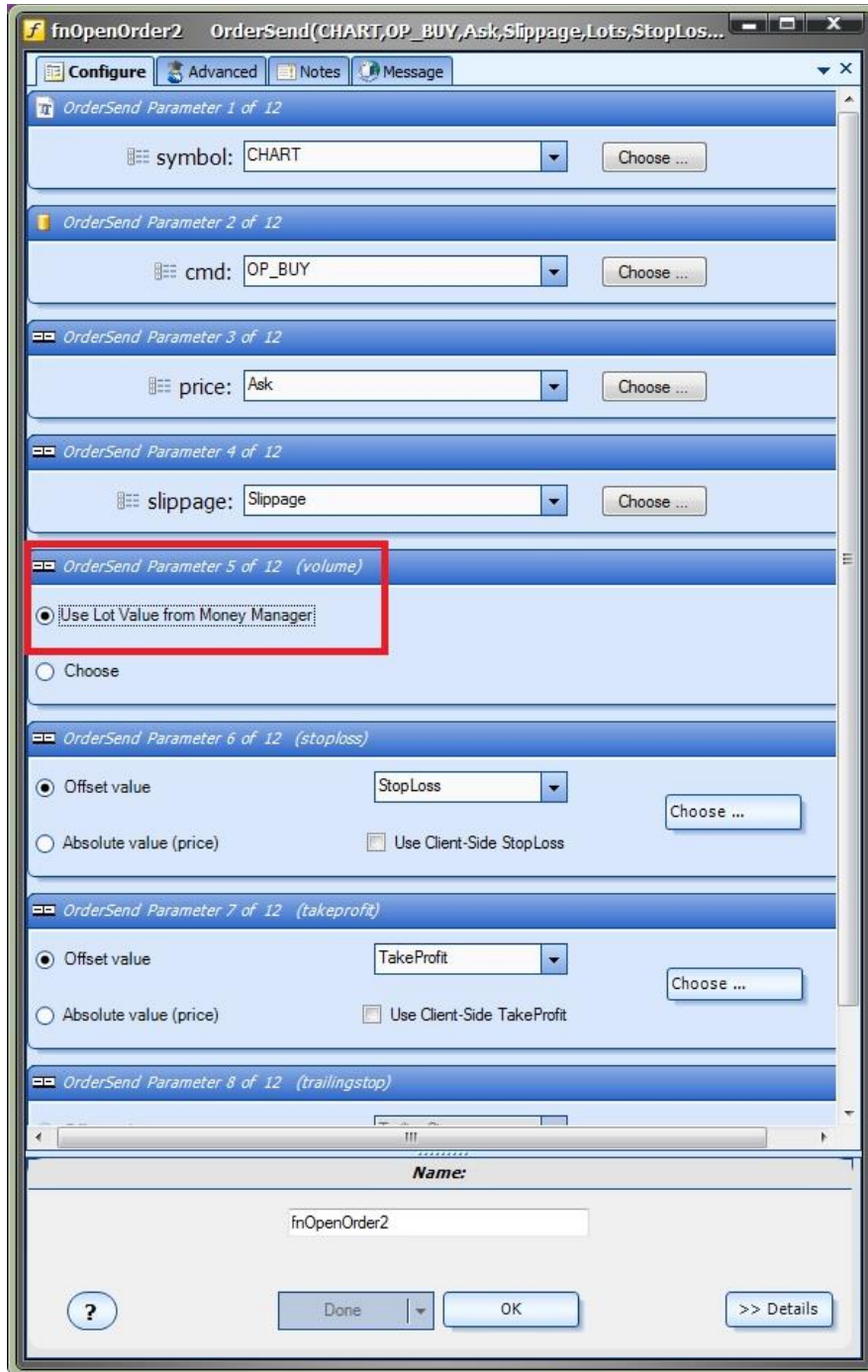
The total number of simulations will equal the **Number of Trades** multiplied by **Number of Samples**



Using the Money Manager

After you have defined and built a Money Management configuration, the Money Management lot size value can be set in the **Volume** parameter of the [fnOpenOrder](#) function.

NOTE: If any changes are made to the Money Management configuration, the VTS system must be rebuilt to rebuild the money management tables.



Money Management Table

After using the [Configuration](#) to build and plot a money management scheme, VTS builds a table values.

This table can be saved to a [CSV](#) file and opened using a program like Excel.

This table provides a view of the lot size versus balance scheme.

You can refer to this table to find the correct lot size when *manually* opening trades

This same data is used by your Expert Advisor to find the correct lot size when the EA opens a trade.

This is an example of a money management table.

Trade	Number of Lots	Profit	Account Balance
1	0.7	140	5140
2	0.8	160	5280
3	0.8	160	5440
4	0.8	160	5600
5	0.8	160	5760
6	0.9	180	5920
7	0.9	180	6100
8	0.9	180	6280
9	1	200	6460
10	1	200	6660
11	1	200	6860
12	1	200	7060
13	1.1	220	7260
14	1.1	220	7480
15	1.1	220	7700
16	1.2	240	7920
17	1.2	240	8160
18	1.3	260	8400
19	1.3	260	8660
20	1.3	260	8920
21	1.4	280	9180

22	1.4	280	9460
23	1.5	300	9740
24	1.5	300	10040
25	1.6	320	10340
26	1.6	320	10660
27	1.7	340	10980
28	1.7	340	11320
29	1.8	360	11660
30	1.8	360	12020
31	1.9	380	12380
32	1.9	380	12760
33	2	400	13140
34	2	400	13540
35	2.1	420	13940
36	2.2	440	14360
37	2.2	440	14800
38	2.3	460	15240
39	2.4	480	15700
40	2.5	500	16180
41	2.5	500	16680
42	2.6	520	17180
43	2.7	540	17700
44	2.8	560	18240
45	2.9	580	18800
46	3	600	19380
47	3	600	19980
48	3.1	620	20580
49	3.2	640	21200
50	3.3	660	21840

Package Plug-in

Requires VTS-Connect minimum version 4.0.0.77

Use the **Package** plug-in to package your drawings, EA, documents, and videos into a single sharable file. Allows you to customize the package with your logo, website, email address and affiliate ID. Ideal for sharing your trading strategy.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor Builder* is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the Package Plug-in

You must enter your License key to enable the **Package Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The **email** address is the email address used to purchase [VTS](#).

The **License Key** is the key listed in the Members Area.

The **Update** button is used to verify the email address and license key.

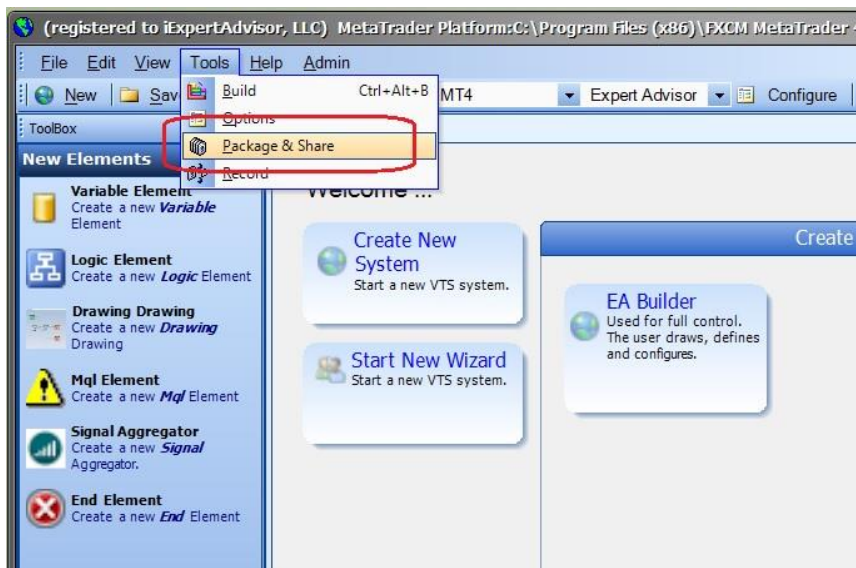
The **edit key** button is used edit the key value.



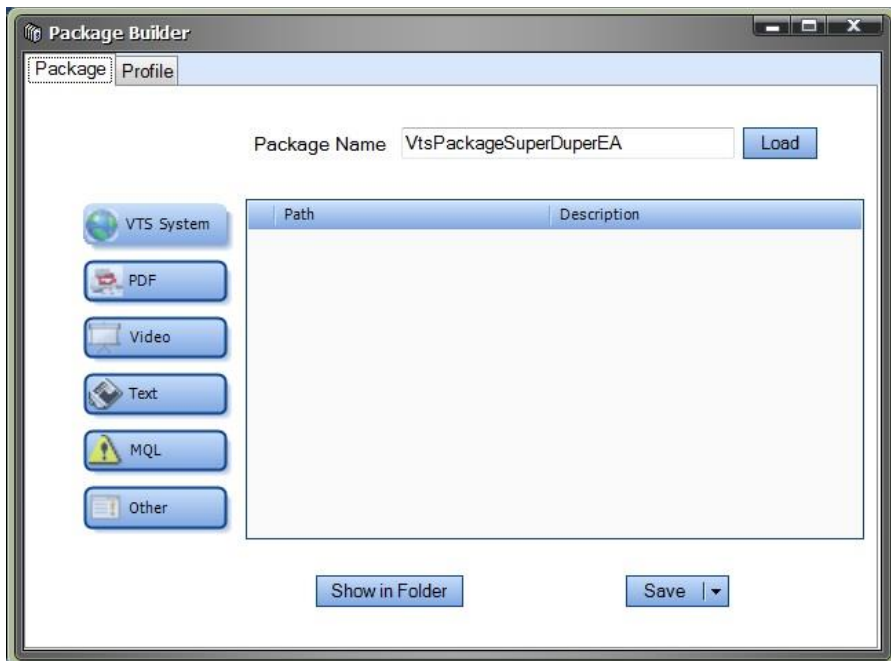
Create a Package

In VTS, the Package Plug-in is found under: **Tools->Package & Share**

Note: A VTS system must be open to use the Package Plug-in



Selecting **Tools->Package & Share** opens the **Package Builder** window:



The **"Package Name"** field will auto-populate, but it can be edited.

Clicking **"VTS System"** will package all of the VTS drawings for the open system into a zip file with the same name as the VTS system. *This is a **proprietary** ZIP file format used by VTS to package drawings and other VTS system data.*

Clicking **PDF** allows you to add a PDF file.

Clicking **Video** allows you to add a video file, the default is mp4, but any format can be selected.

Clicking **Text** allows you to add a text file.

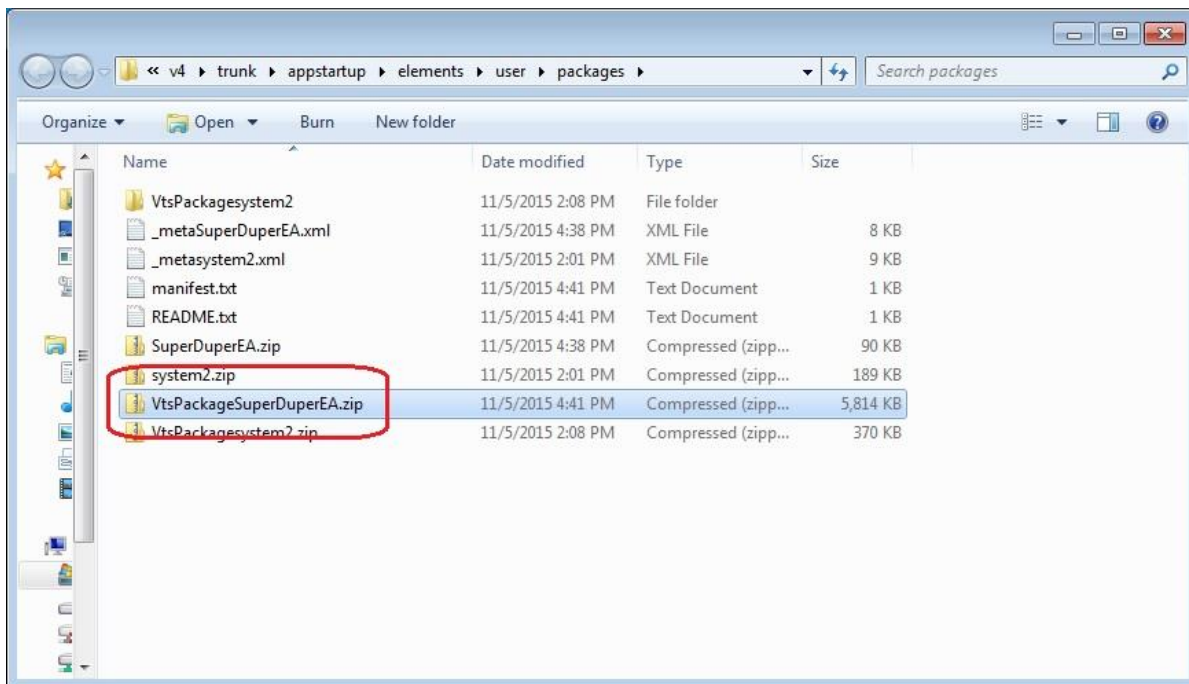
Clicking **MQL** allows you to add an mq4 file.

Clicking **Other** allows you to add any other file type to the package.

Build the Package

The **Save** button is used to build and save the package. The file name of the package is the "Package Name" with a zip extension. *This is a **standard ZIP** file that can be opened on any Windows computer.*

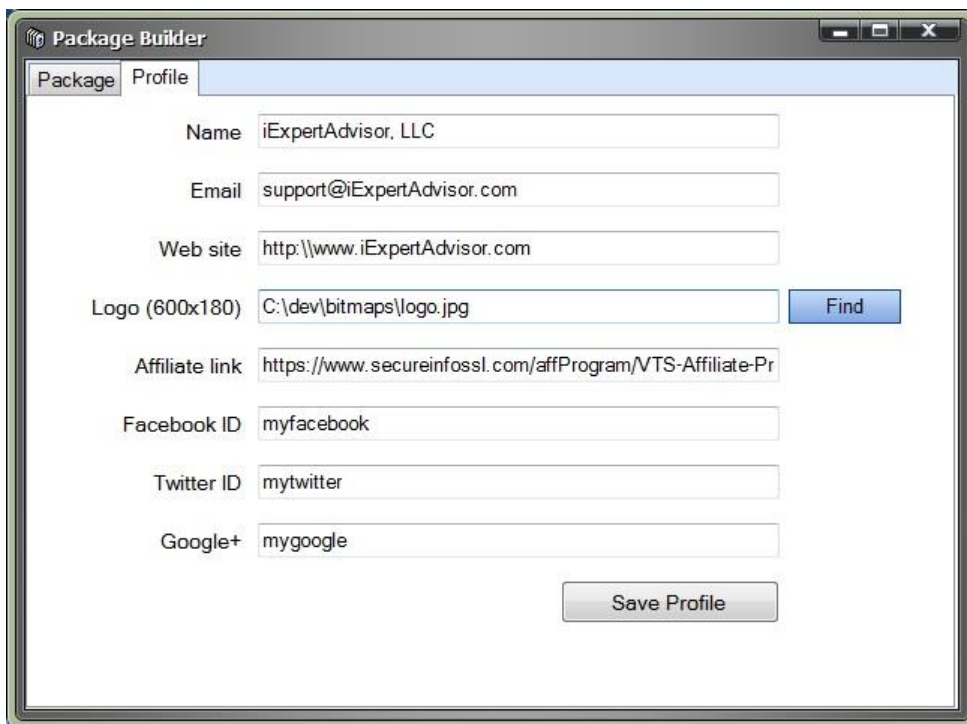
The **"Show in Folder"** button is used to open the folder where the package file is created. This is convenient for locating the package for posting or emailing.



Set Your Profile

Setting the attributes in your profile allows you to personalize (or **brand**) your VTS Package, and even include your affiliate link.

Selecting **Tools->Package & Share** opens the **Package Builder** window. Clicking the "**Profile**" tab shows the profile settings.



The screenshot shows the 'Package Builder' window with the 'Profile' tab selected. The window contains the following fields and buttons:

- Name:** iExpertAdvisor, LLC
- Email:** support@iExpertAdvisor.com
- Web site:** http://www.iExpertAdvisor.com
- Logo (600x180):** C:\dev\bitmaps\logo.jpg (with a 'Find' button)
- Affiliate link:** https://www.secureinfossl.com/affProgram/VTS-Affiliate-Pr
- Facebook ID:** myfacebook
- Twitter ID:** mytwitter
- Google+:** mygoogle
- Save Profile:** (button)

Most of these values appear on the **Package welcome page** that is opened when a user drags and drops your package into VTS.

Name: Personal or company name.

Email: Your email address.

Web site: Your website (a link is created on the **Package welcome page**)

Logo: Your logo, or any image that is about 600x180 pixels.

Affiliate link: Your iExpertAdvisor link that allows you to earn commission by tagging a customer as your own. This link is found in the Affiliate section. For more information, go here: <http://www.iexpertadvisor.com/affiliate-program/>

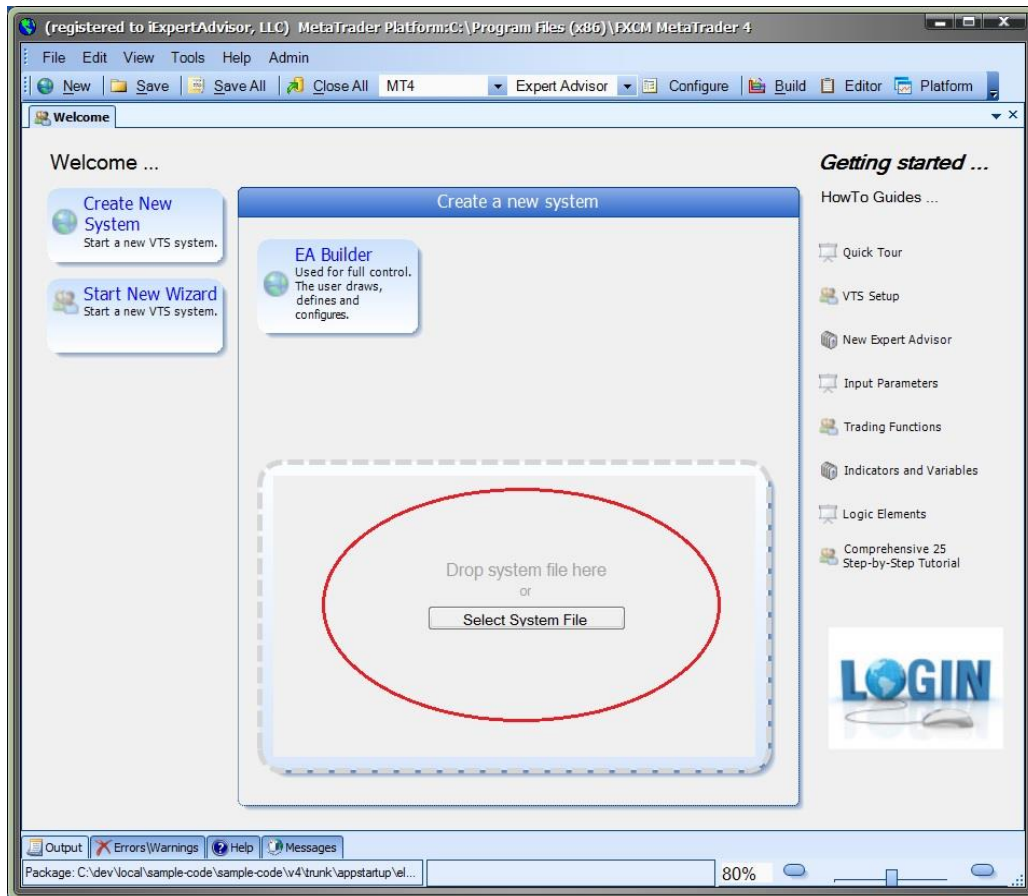
Facebook, Twitter and Google+ are currently unused but will support links in a later version.

Note: The best way to see the format of your profile on the **Package welcome page** is to create a sample package and load the package.

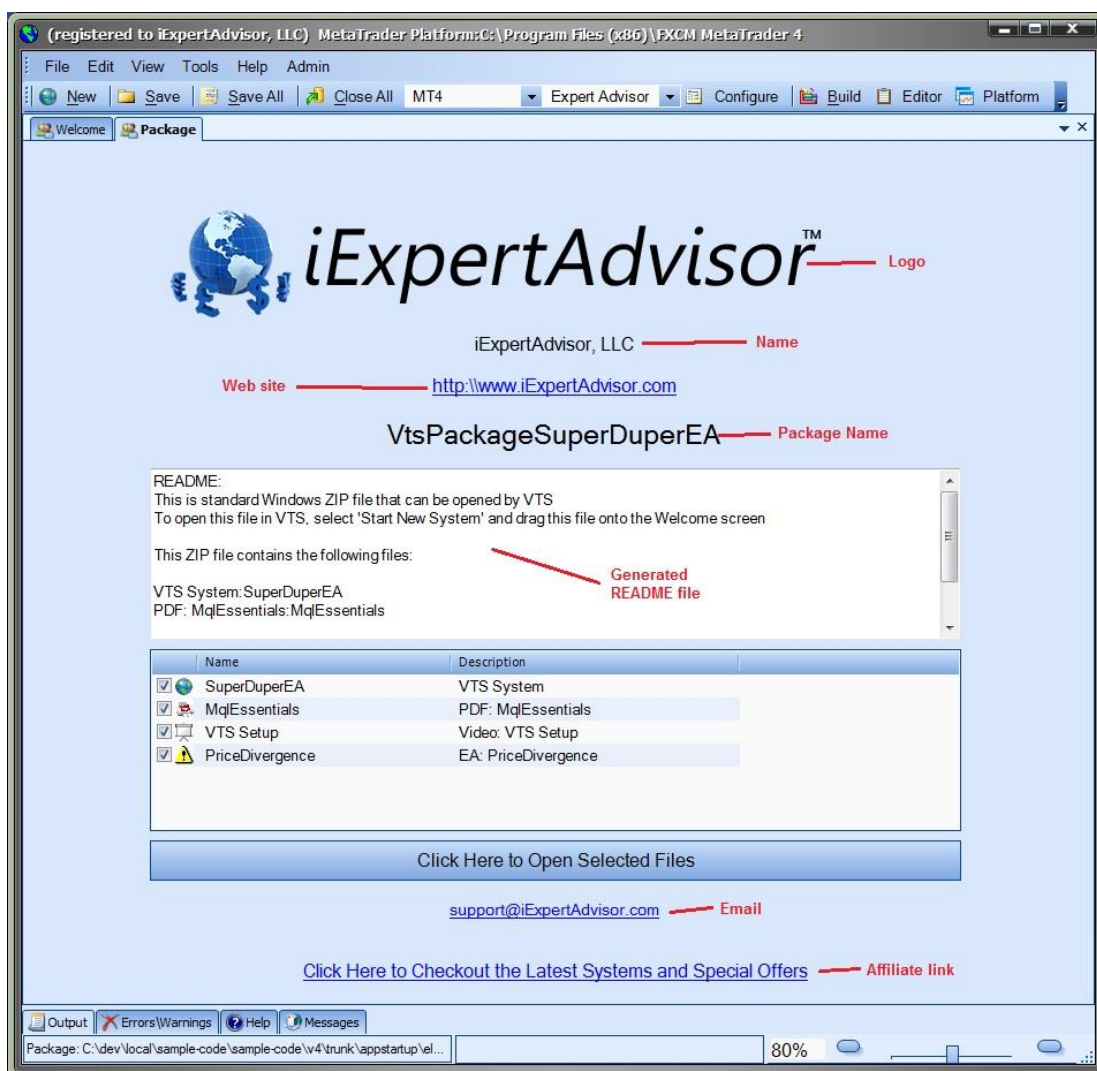
Load a Package

To load a package that has been built using the VTS **Package Plug-in**, drag the package file onto the **Welcome** page.

Note: Any VTS user can open a Package file. The **Package Plug-in** is *not* required to open a package.



When a package is dropped onto the **Welcome** page, the **Package welcome** page is displayed.



The values set in your profile appear in various locations on the **Package welcome** page. Note that some values use default iExpertAdvisor values if they are not defined in your profile.

An auto-generated README file is displayed that details the contents of the package.

A list of package items is displayed with a checkbox.

When the "**Click Here to Open Selected Files**" button is clicked, all checked files are opened. Many files will open within a VTS tab if possible. Others are opened using the default executable associated with the file.

The bottom link "**Click Here to Checkout the Latest Systems and Special Offers**" contains your affiliate link. When users click this link, the user is identified as your lead and you receive a commission for any products they purchase. For more information, go here: <http://www.iexpertadvisor.com/affiliate-program/>

Pivot Points Plug-in

Requires VTS-Connect minimum version **4.0.0.79**

The **Pivot Point Plug-in** allows an Expert Advisor to draw Pivots Points for any timeframe, including daily, weekly and monthly timeframes and read the values of the main, support and resistance pivot-point values. The values can be used for any trade entry or exit logic, as well as for stoploss and takeprofit values.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor Builder* is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the *Pivot Points Plug-in*

You must enter your License key to enable the ***Pivot Points Plug-in***. Your license key for all of your VTS products can be found in the [Members Area](#).

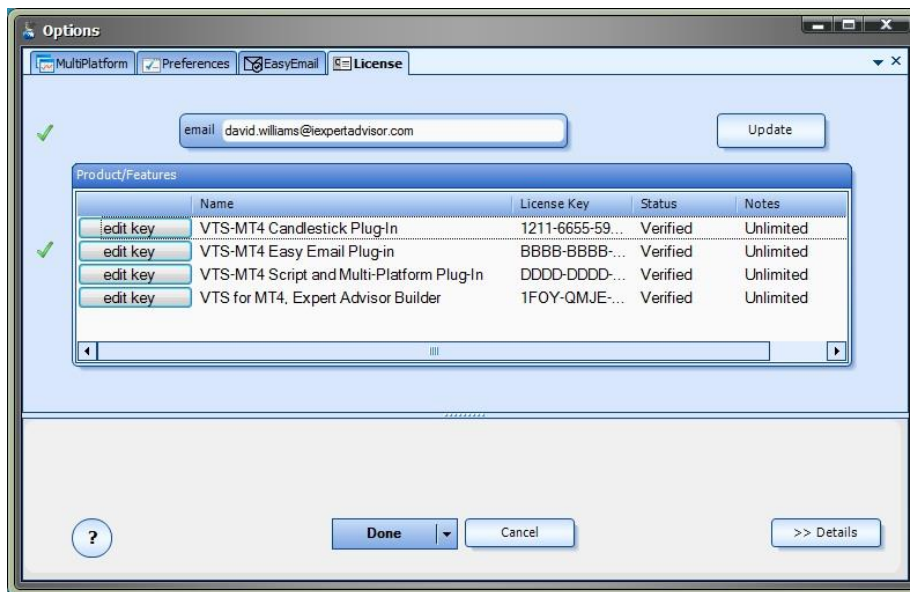
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The **email** address is the email address used to purchase [VTS](#).

The **License Key** is the key listed in the Members Area.

The **Update** button is used to verify the email address and license key.

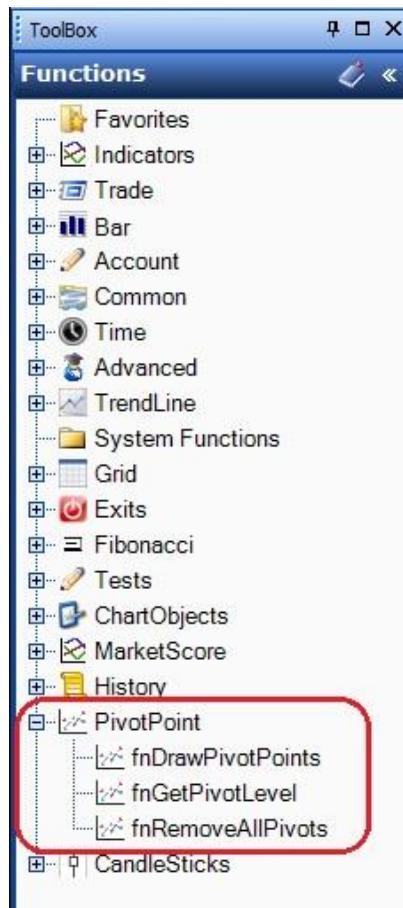
The **edit key** button is used edit the key value.



Pivot Point Functions in the Toolbox

Once enabled, the Pivot Point functions are available in the [Toolbox](#) Function tab under the **PivotPoint** menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.



Pivot Point Functions

The Pivot Point function library include these functions:

[fnDrawPivotPoints](#)

Used to programatically draw Pivot Point lines on a price chart.

[fnGetPivotLevel](#)

Used to get the value of a Pivot Point level.

[fnRemoveAllPivots](#)

Used to programatically delete all Pivot Point lines from a price chart.

fnDrawPivotPoints

The function *fnDrawPivotPoints* is used to programmatically draw a Pivot Point, as a horizontal line, on a price chart.

After the *fnDrawPivotPoints* function has been added to a [Drawing](#), it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

The definition of the pivot point calculations are:

$$\text{Pivot Point} = (\text{Previous High} + \text{Previous Low} + \text{Previous Close}) / 3$$

The pivot point can then be used to determine levels of estimated support and resistance levels for the day:

$$\text{Resistance Level 1} = (2 * \text{Pivot Point}) - \text{Previous Low}$$

$$\text{Support Level 1} = (2 * \text{Pivot Point}) - \text{Previous High}$$

$$\text{Resistance Level 2} = (\text{Pivot Point} - \text{Support Level 1}) + \text{Resistance Level 1}$$

$$\text{Support Level 2} = \text{Pivot Point} - (\text{Resistance Level 1} - \text{Support Level 1})$$

$$\text{Resistance Level 3} = (\text{Pivot Point} - \text{Support Level 2}) + \text{Resistance Level 2}$$

$$\text{Support Level 3} = \text{Pivot Point} - (\text{Resistance Level 2} - \text{Support Level 2})$$

These parameters appear on the main *configure* tab:

Parameter Name	Data type	Description
basename	string	The basename of line objects that are drawn on the price chart. The basename is combined with the bar number, shift and line type to create a unique name for each line that is drawn on the chart.
type	timeframe	The timeframe of the pivot point. Any timeframe can be used, but Daily, Weekly and Monthly and most popular. On a 1 hour price chart, a Daily pivot point line will be drawn across 24 candles.
count	count	This controls how many lines are drawn. For a Daily pivot point, a count = 1 will draw a line representing today's pivot point. A count = 2 will draw today's and yesterday's pivot points.
lineColor	color	The color of the line/s.

These parameters appear on the *advanced* tab

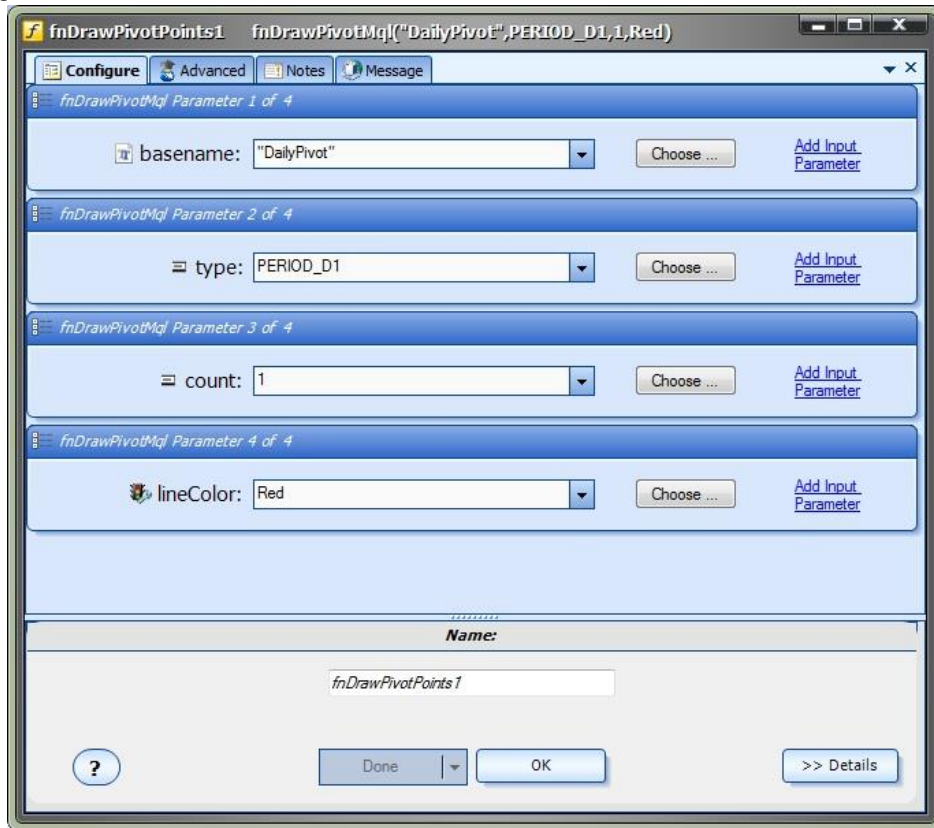
Note: In order to capture the values of any of the support or resistance lines using the function [fnGetPivotLevel](#), the lines must be drawn using the *fnDrawPivotPoints* function.

These values are used to draw the support or resistance lines:

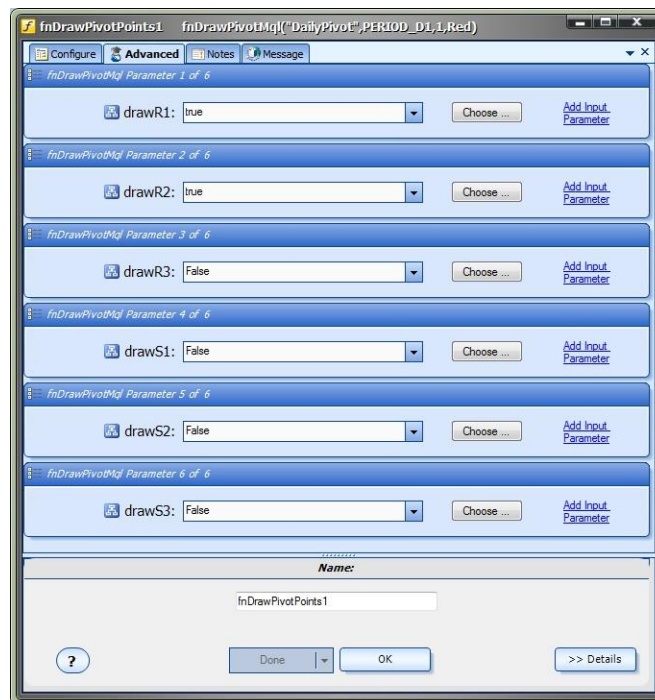
Parameter Name	Data type	Description
drawR1	Boolean	If true, the Resistance Level 1 line is drawn on the chart.
drawR2	Boolean	If true, the Resistance Level 2 line is drawn on the chart.
drawR3	Boolean	If true, the Resistance Level 3 line is drawn on the chart.
drawS1	Boolean	If true, the Support Level 1 line is drawn on the chart.
drawS2	Boolean	If true, the Support Level 2 line is drawn on the chart.
drawS3	Boolean	If true, the Support Level 3 line is drawn on the chart.

Note: It is possible that not all of the support or resistance lines will be shown on the price chart. If the value of the line is too large or too small it may not appear on the price chart depending on the scale and timeframe of the chart.

The main **configure** tab:



The **advanced** tab, used for drawing support and resistance lines:



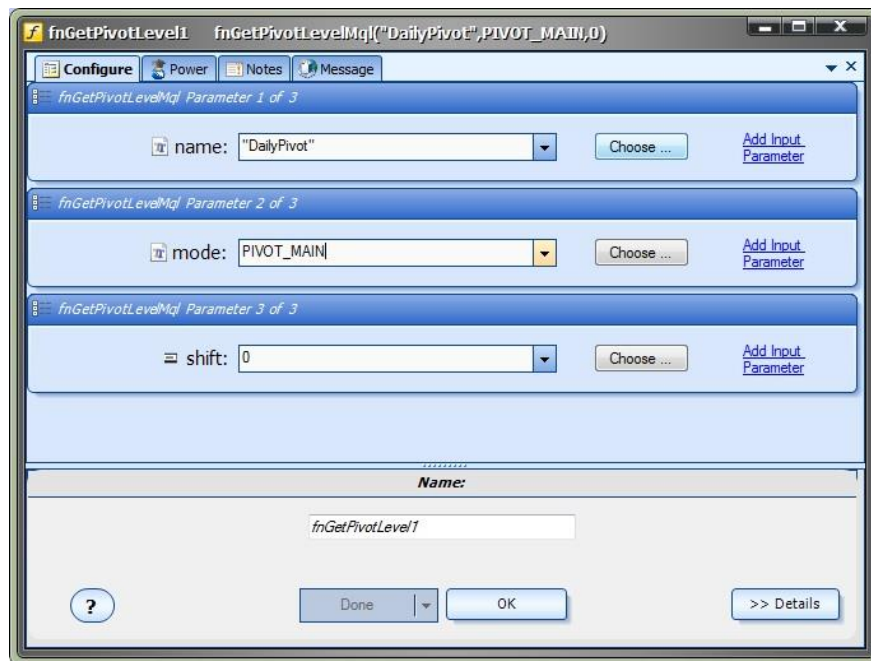
fnGetPivotLevel

The *fnGetPivotLevel* function is used to get the value of a Pivot Point line.

After the *fnGetPivotLevel* function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
name	string	The name of the pivot point line to get the value for, as defined in the basename parameter of the fnDrawPivotPoint function. Note: The name entered in the parameter must exactly match the basename parameter of the fnDrawPivotPoint function.
mode	PIVOT value (string)	Defined what line to use to capture the value: PIVOT_MAIN PIVOT_S1 PIVOT_S2 PIVOT_S3 PIVOT_R1 PIVOT_R2 PIVOT_R3 . Note: In order to capture the values of any of the support or resistance lines using the function <i>fnGetPivotLevel</i>, the lines must be drawn using the fnDrawPivotPoints function.
shift	integer	The count index of where to capture the pivot point value. The count is defined in the fnDrawPivotPoints function.



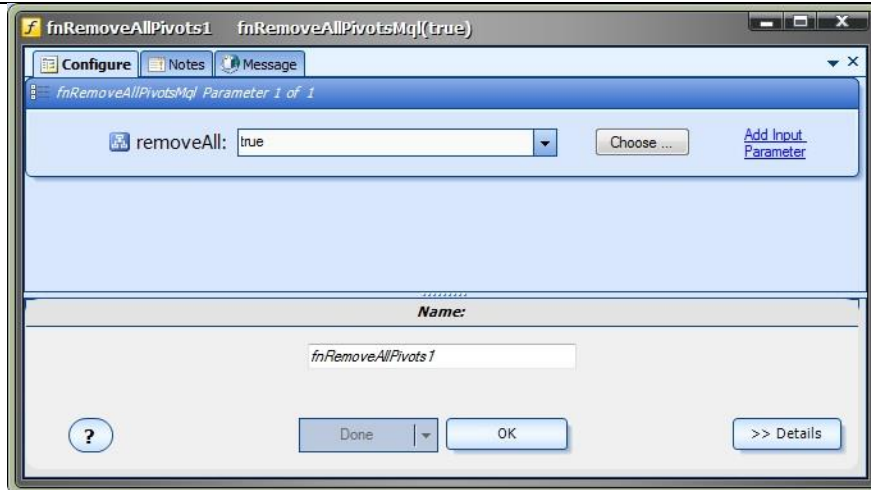
fnRemoveAllPivots

The **fnRemoveAllPivots** function is used to remove all Pivot Point lines from all price charts.

After the **fnRemoveAllPivots** function has been added to a Drawing, it is configured by clicking the (+) button along the bottom of the Element.

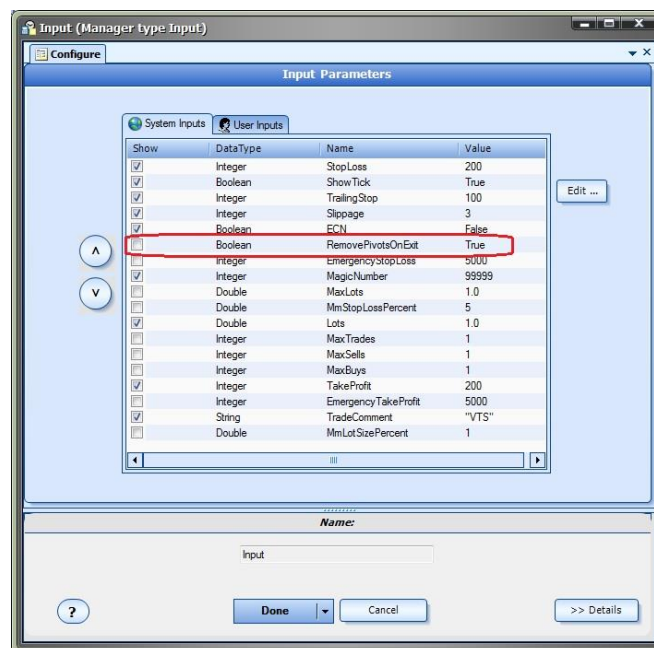
The [Function Configuration](#) window allows you to select values for each parameter.

Parameter Name	Data type	Description
removeAll	Boolean	if true, all pivot lines are removed



By default, all pivot points are removed when the EA is removed from the price chart. To change this behavior, set the Input value of **RemovePivotsOnExit** to false.

The Input value of **RemovePivotsOnExit** is found on the Input Manager "System Inputs" tab:



Using the Pivot Point Plug-in

The Pivot Point plug-in can be used simply to draw pivot points on a price chart. However the real power of the plug-in is in using the pivot point values within an Expert Advisor to make trading decisions.

One typical usage of pivot points is to use one of the resistance values as a breakout value for a BUY trade entry, and use the other pivot lines as a takeprofit target and a stoploss target.

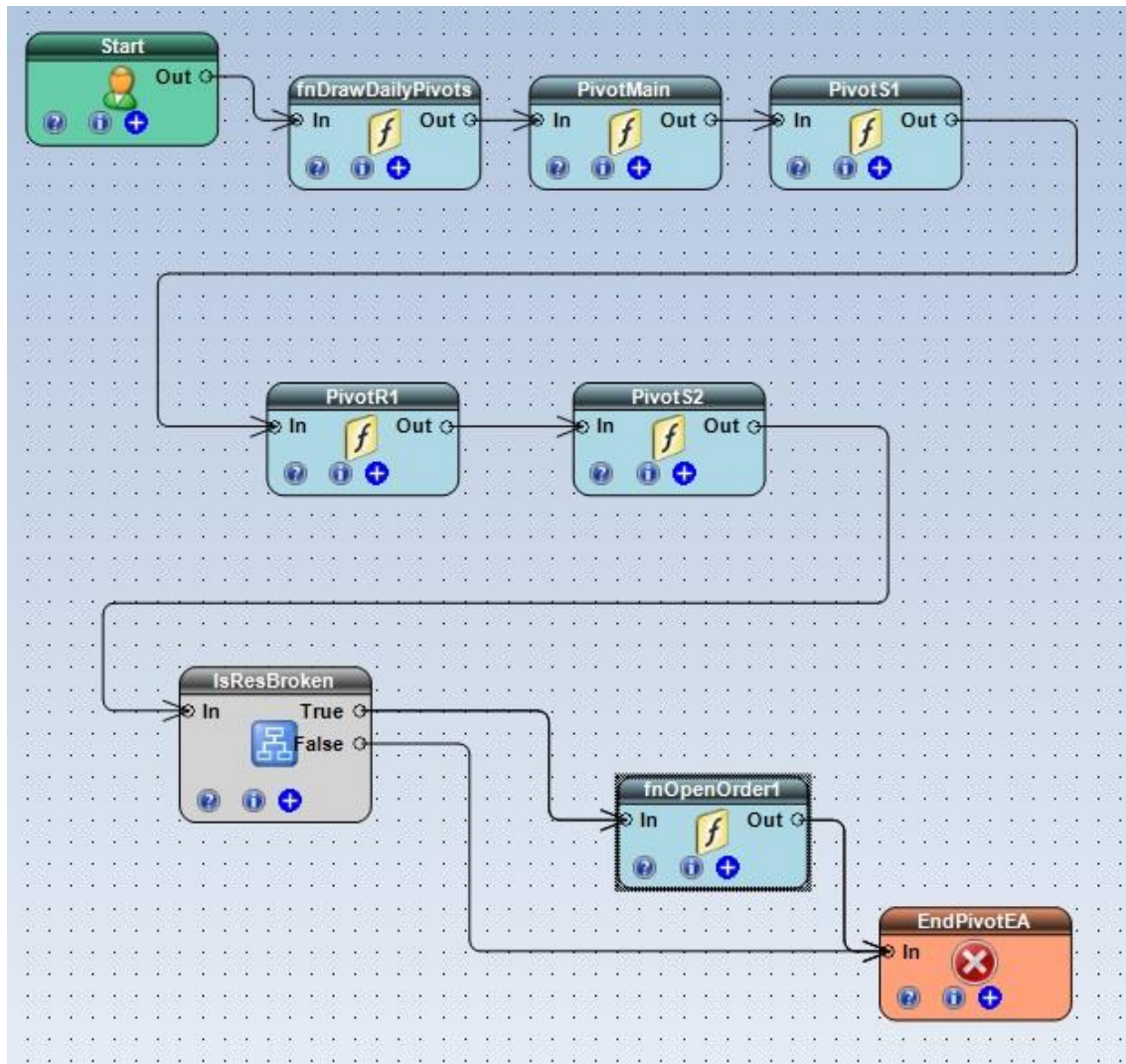
For example:

Resistance Level 1 is used as a price entry. Open a BUY order if the price goes above Resistance Level 1.

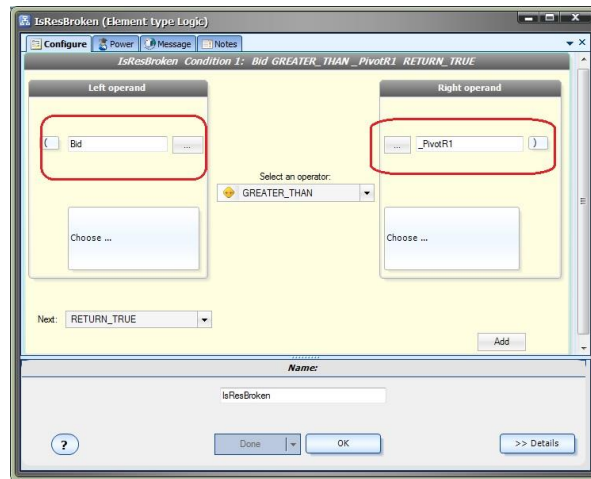
Resistance Level 2 (or 3) is the used as the target takeprofit value;

Support Level 1 (or 2 or 3) is used as the stoploss value.

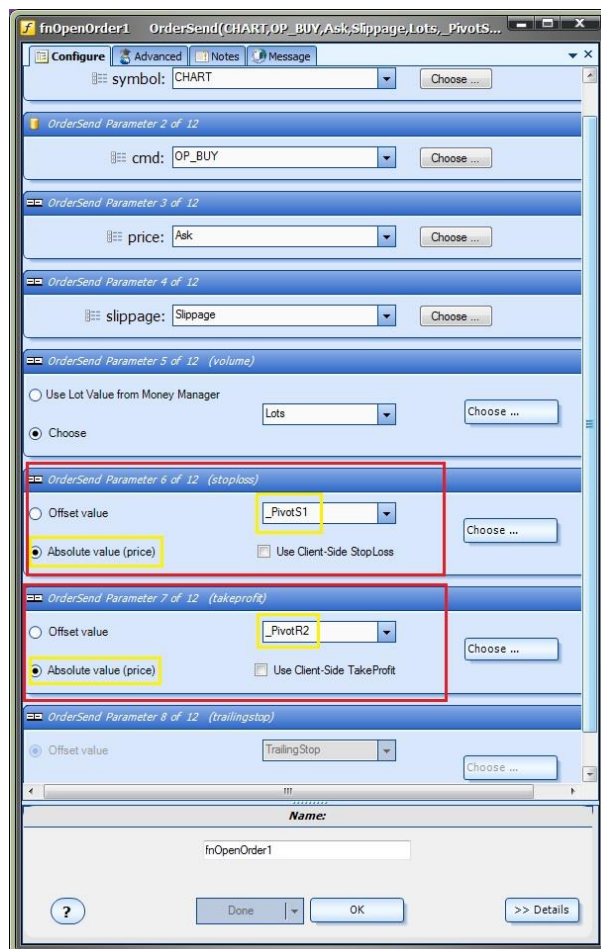
This drawing illustrates a simple trading system that draws daily pivot points, collects the values for the Main, S1, R1 and R2, and opens a BUY trade if the Bid price breaks the Support 1 line.



The logical condition checks if the Bid price breaks the Support 1 line:



The fnOpenOrder function sets the takeprofit as R2 and the stoploss as S1.

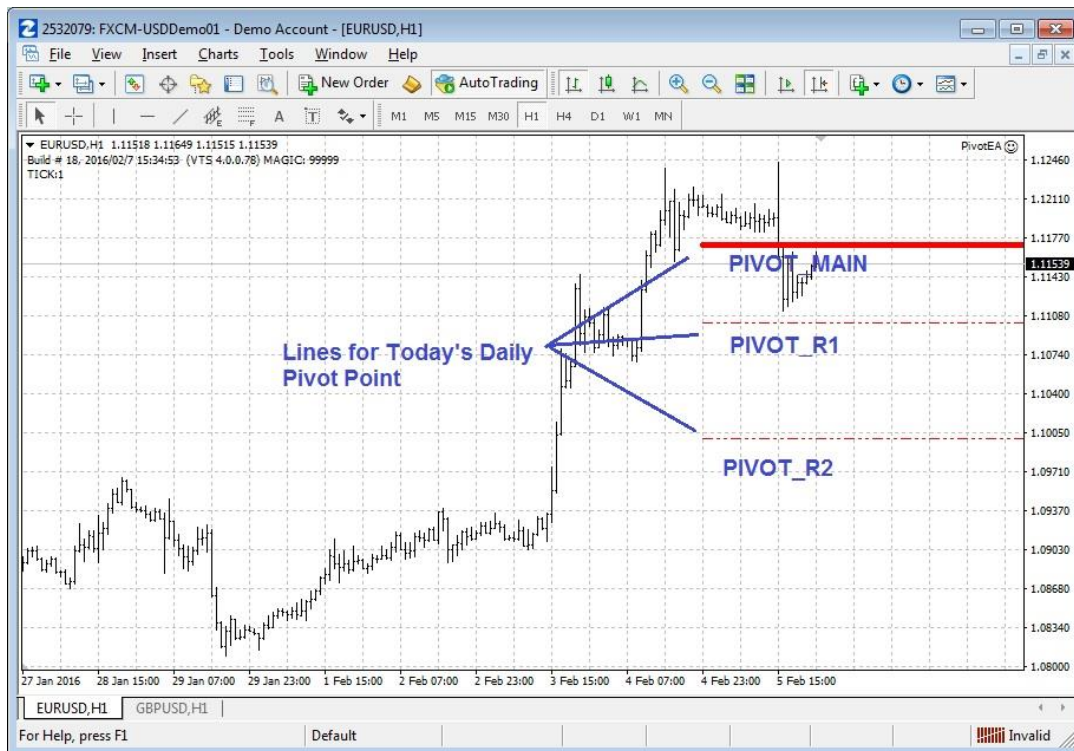


Pivot Points on a MetaTrader Price Chart

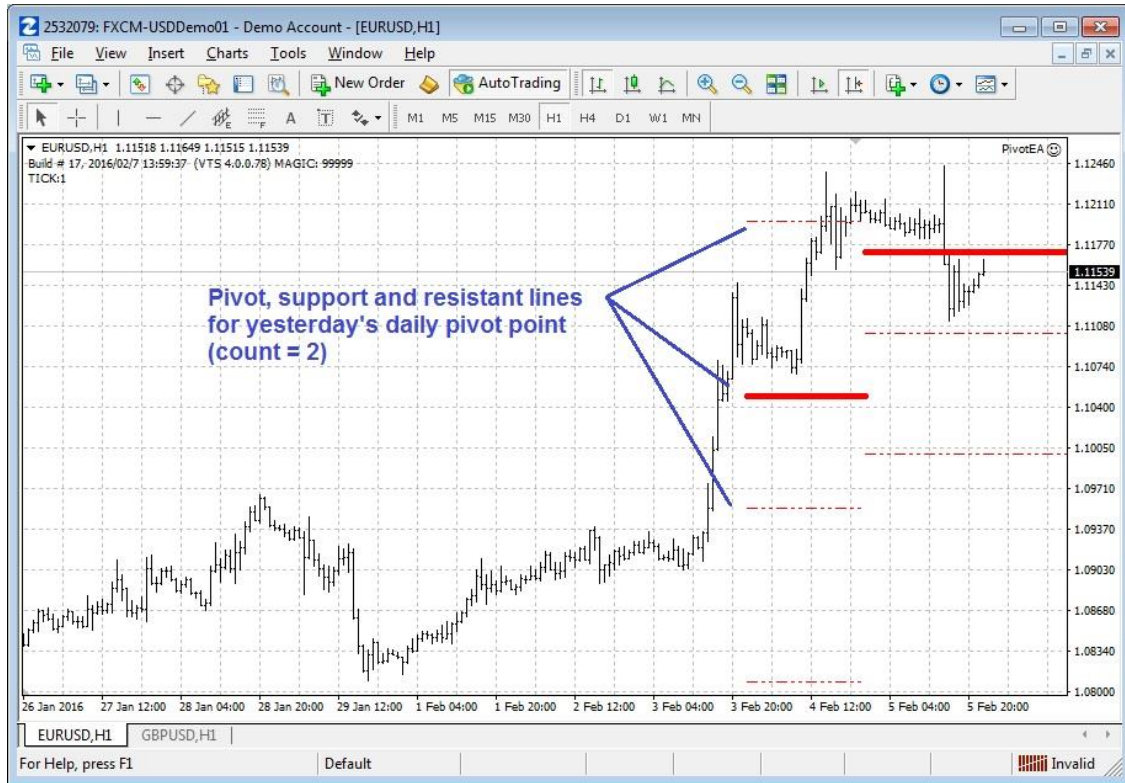
This price chart shows a Main, Resistance 1 and Resistance 2 pivot point lines.

Note that the lines Support 1 and Support 2 are not shown because their values lie outside of the chart highest price.

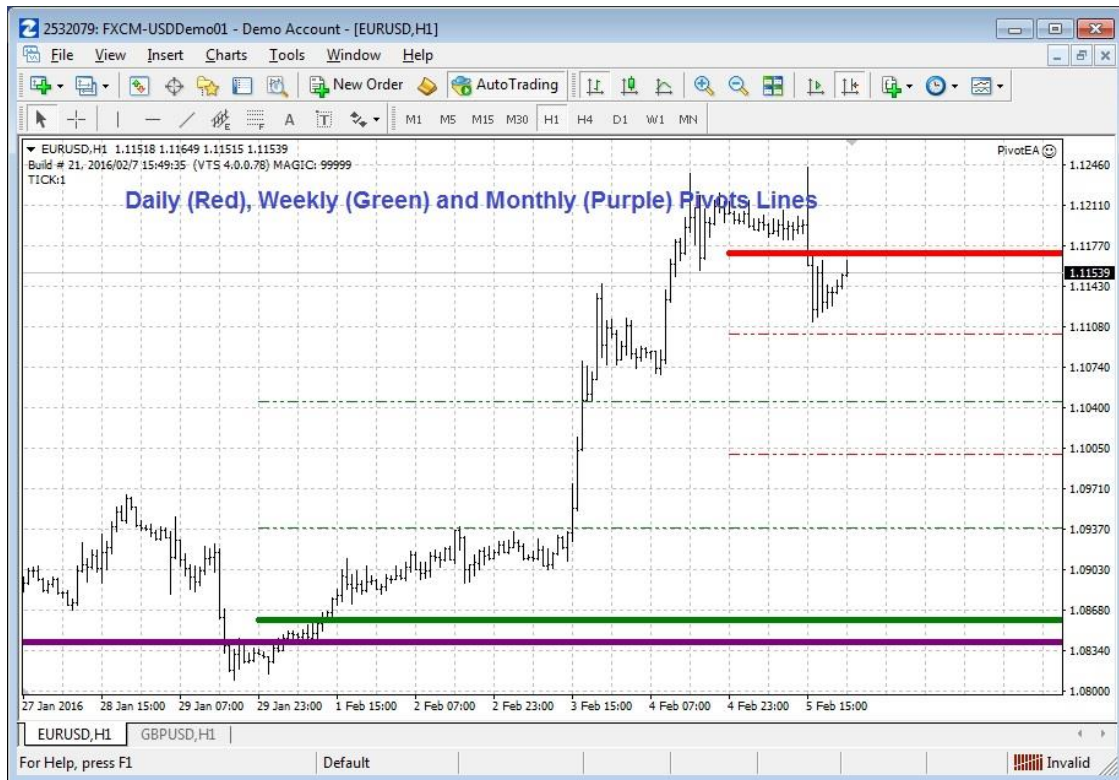
(In the case, the values are because of large upward movement in price.)



This price chart shows the pivot point lines for today and yesterday. The count parameter of `fnDrawPivotPoints` was set to 2.



This price chart shows Daily, Weekly and Monthly pivot lines.



File IO Plug-in

Requires VTS-Connect minimum version **4.0.0.81**

The **File IO Plug-in** allows an Expert Advisor to read and write any text file. The exact format of the file may be defined in any manner using the File IO Manager. The file is read from, or written to, using the functions from the Functions Toolbox *fnFileRead* and *fnFileWrite*.

What is a Plug-in?

VTS stands for **Visual Traders Studio**.

The VTS *Expert Advisor* Builder is a Windows graphical application that enables non-programmers to build complex Expert Advisors by dragging, dropping and connecting logical elements.

The VTS application contains basic functionality to build almost any Expert Advisor.

A **VTS Plug-in** allows traders to easily implement advanced trading techniques using an add-on user interface.



Enable the *File IO* Plug-in

You must enter your License key to enable the **File IO Plug-in**. Your license key for all of your VTS products can be found in the [Members Area](#).

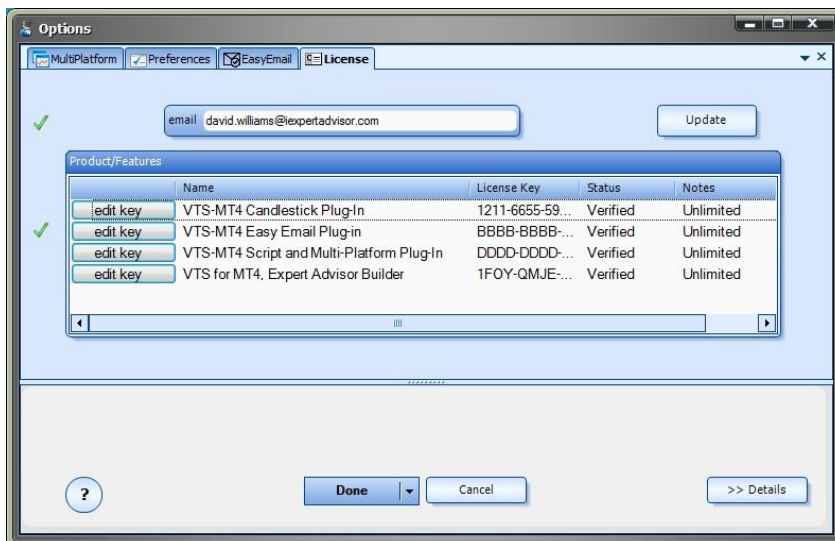
License keys are entered in Visual Traders Studio (VTS) from the License entry tab.

The **email** address is the email address used to purchase [VTS](#).

The **License Key** is the key listed in the Members Area.

The **Update** button is used to verify the email address and license key.

The **edit key** button is used edit the key value.



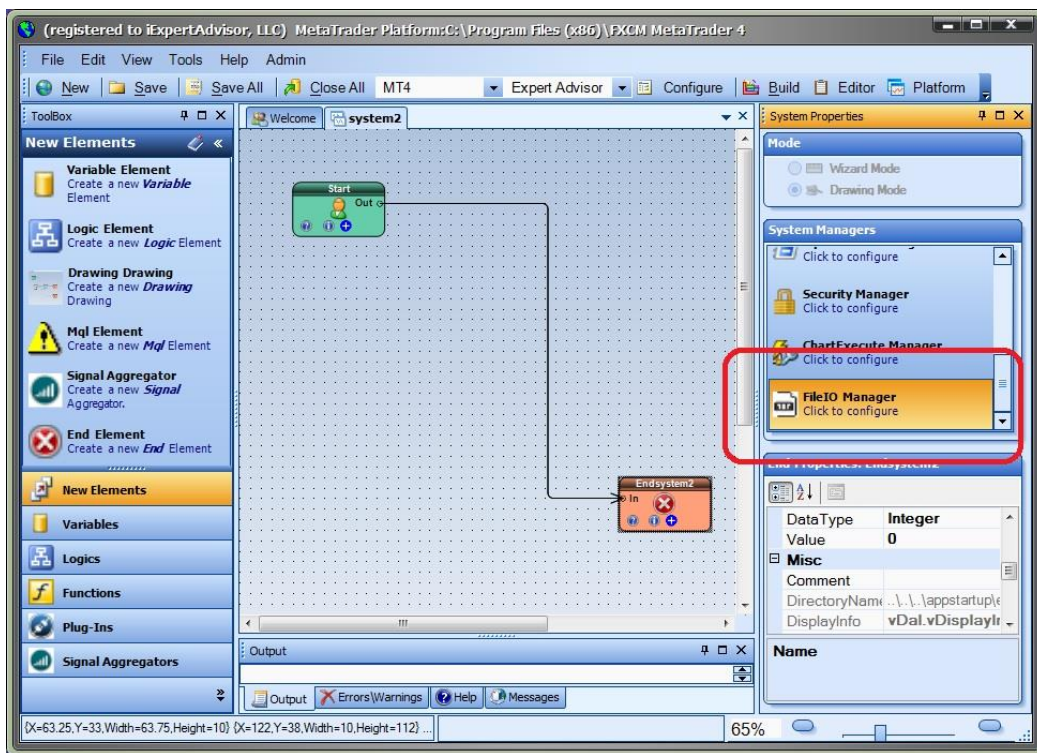
File IO in the System Manager

File IO configuration is accessed through the **File IO System Manager**.

[System Managers](#) are found on the right side of the VTS main application.

Depending on your screen resolution, you may need to scroll down to view the **FileIO** icon.

To open the **File IO** Manager, double-click the icon.



File IO Configuration

File IO configuration is accessed through the [File IO System Manager](#).

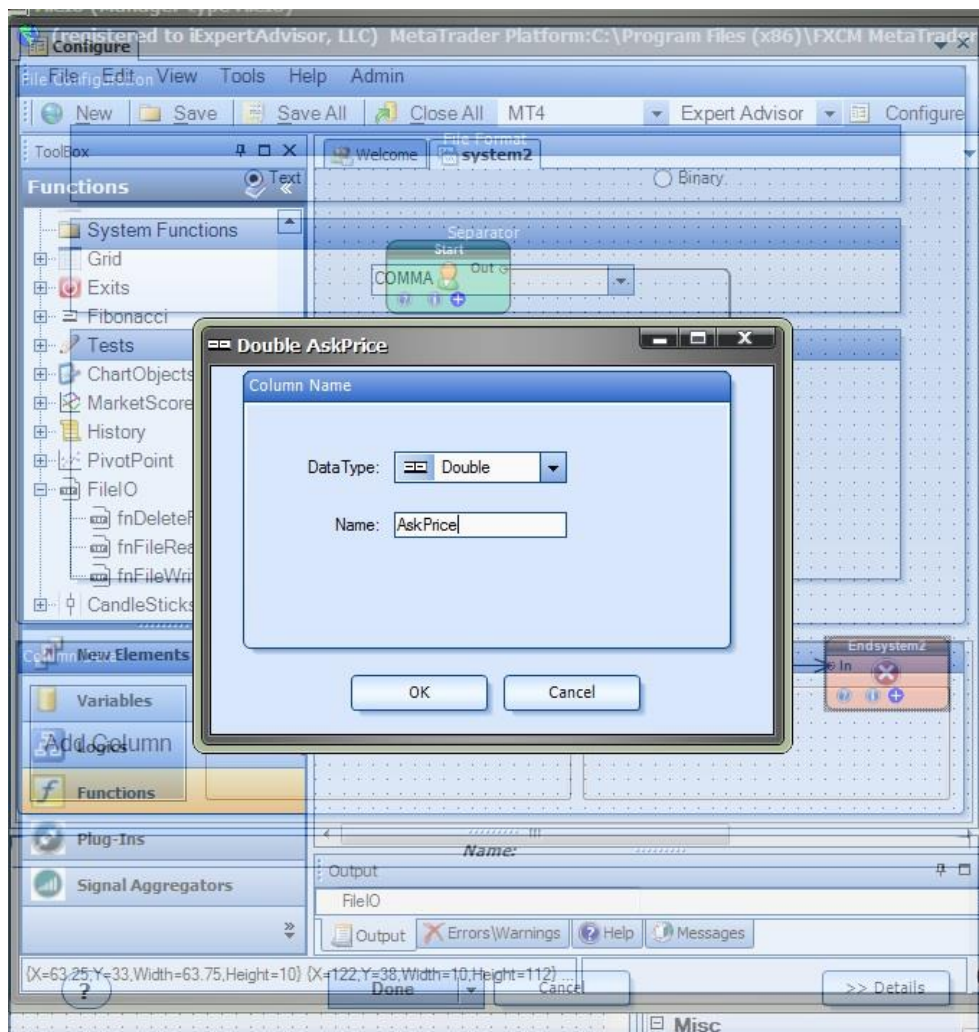
To open the **File IO** Manager, double-click the icon.

This screen is used to configure the format of the file that is read or written to using the functions [fnFileRead](#) and [fnFileWrite](#).



Name	Options	Description
File format	Text Binary	Simple text file Binary read/write mode (without string to string conversion).
Separator	COMMA TAB SPACE PIPE	Character used to separate data values
File Size, Max Type	LINE_MAXTYPE BYTE_MAXTYPE	Mac Value in bytes or lines.
File Size, Max Value	Integer number (0-max)	A value 0 is unlimited.
File Size, Max Action	OVER_WRITE ROLL_OVER	Overwrite the last line of the file. Delete the file and start over.
Add Column	Add a Data column	Add a column of any data type. As columns are added they appear to the left. Columns can be dragged to be rearranged. Columns can be edited by double-clicking. Columns can be deleted by right-clicking

Clicking the "Add Column" button displays the following window to create or edit a column:



Example:

To create a file that writes the currency symbol, bid and ask price on each tick:

```
EURUSD,1.1229300000,1.1232600000
```

The first column is a string data type and can be named anything, for example **SymbolName**

The second column is a double data type and be named anything, for example **BidPrice**

The third column is a double data type and be named anything, for example **AskPrice**

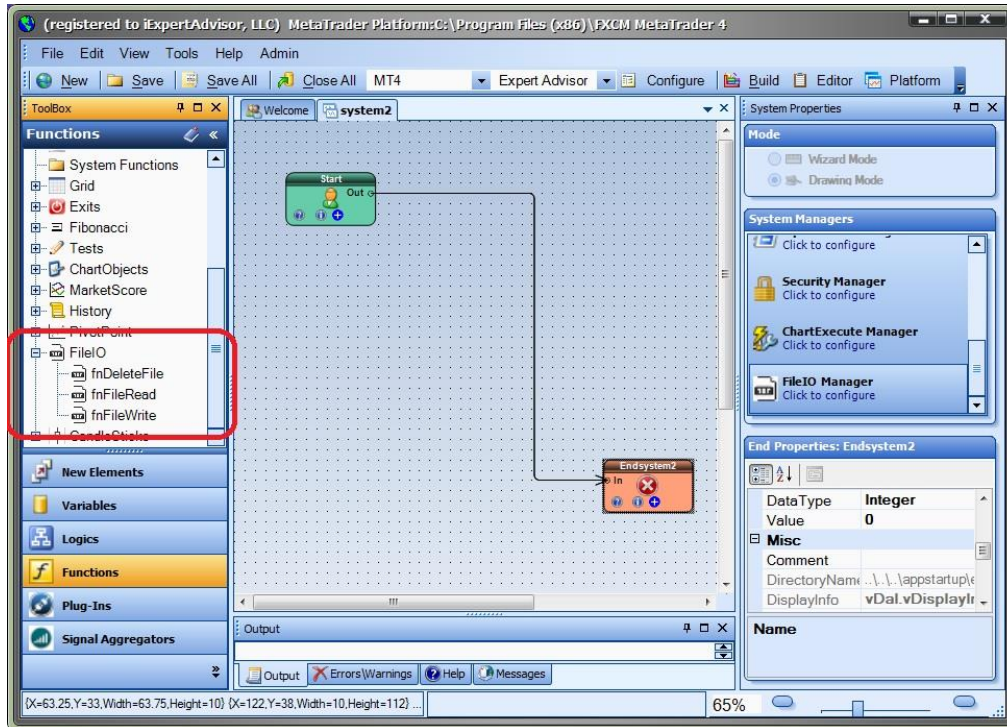
The columns are separated by a comma: the Separator value is selected as **COMMA**.

Refer to the sections on [fnFileRead](#) and [fnFileWrite](#) to read and write lines of data to this file format.

File IO Functions in the Toolbox

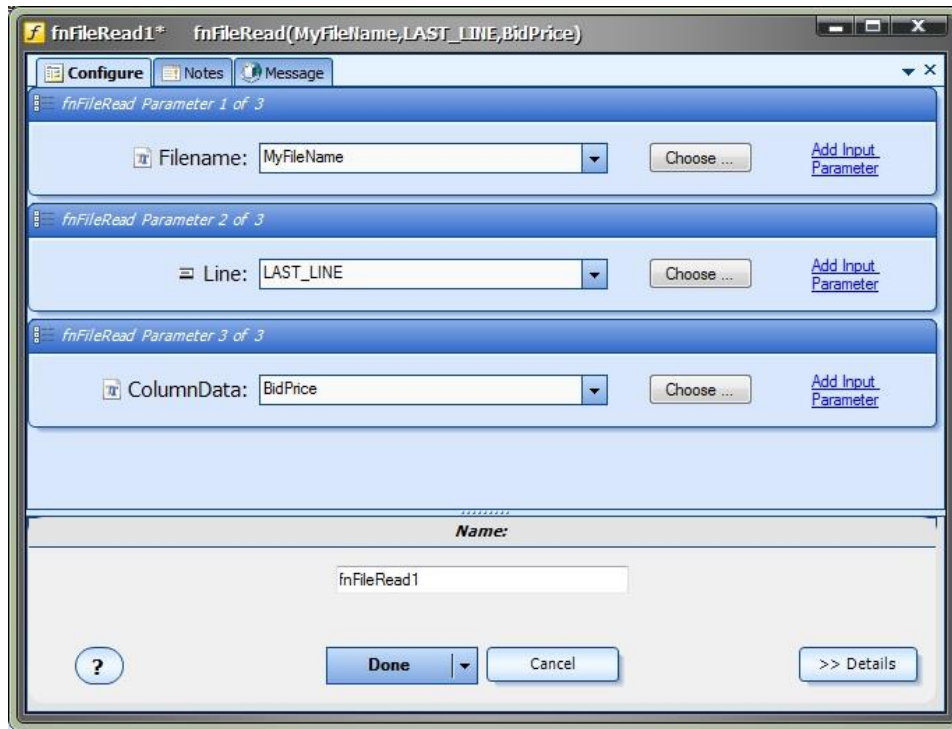
Once enabled, the File IO functions are available in the [Toolbox](#) Function tab under the **FileIO** menu.

These functions are dragged and dropped from the Toolbox onto the [Drawing Pad](#) like any other functions.



fnFileRead

The function **fnReadFile** is used to read a single data value from a file. The configuration of the file is defined in the [File IO System Manager](#)

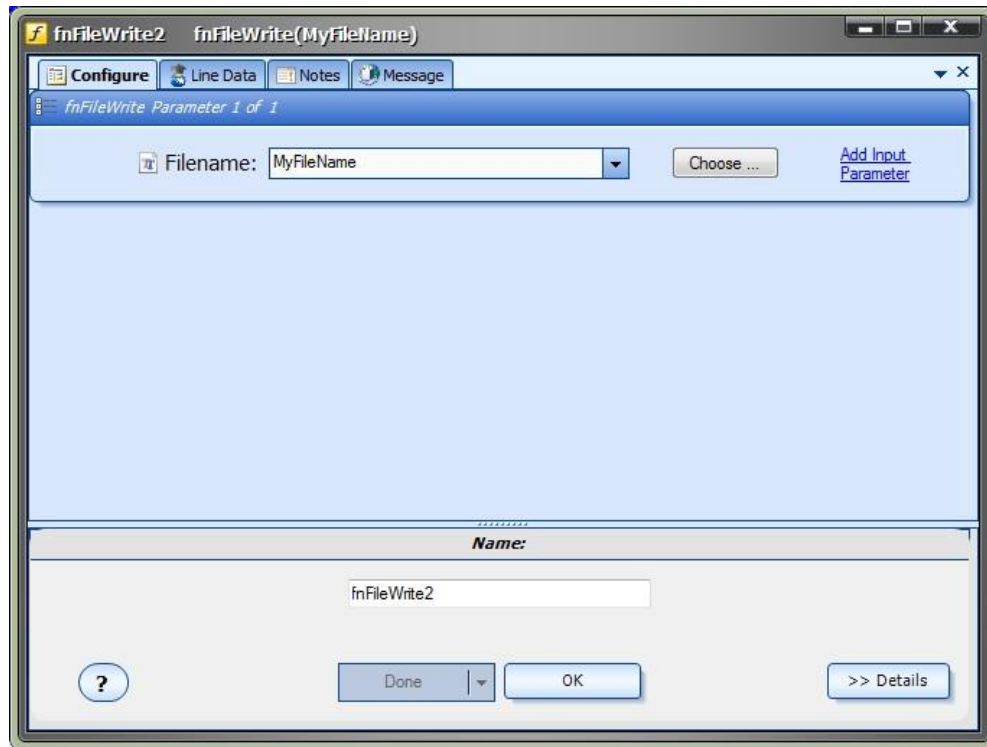


Parameter Name	Data type	Description
FileName	string	The name of the file to read. MetaTrader will only read files in the MT platform "Files" folder, for example: C:\Program Files (x86)\FXCM MetaTrader 4\MQL4\Files
Line (number)	integer	The line number of the file to read. The special value LAST_LINE always reads the last line of the file.
ColumnData	Any data type	The data column to read, as defined in the File IO Configuration . The selection of the ColumnData parameter determines the data type returned from this function. The values available for selection are determine by the data columns defined in the File IO Configuration . For example, if three data columns are defined as: SymbolName (string) BidPrice (double) AskPrice (double) Those three values will be available for the selection of the ColumnData parameter. If the "SymbolName" value is selected, the function will return a string data type. If the "BidPrice" or "AskPrice" value is selected, the function will return a double data type. The returned data type may be important if the value is used in a comparison statement of a logical condition

fnFileWrite

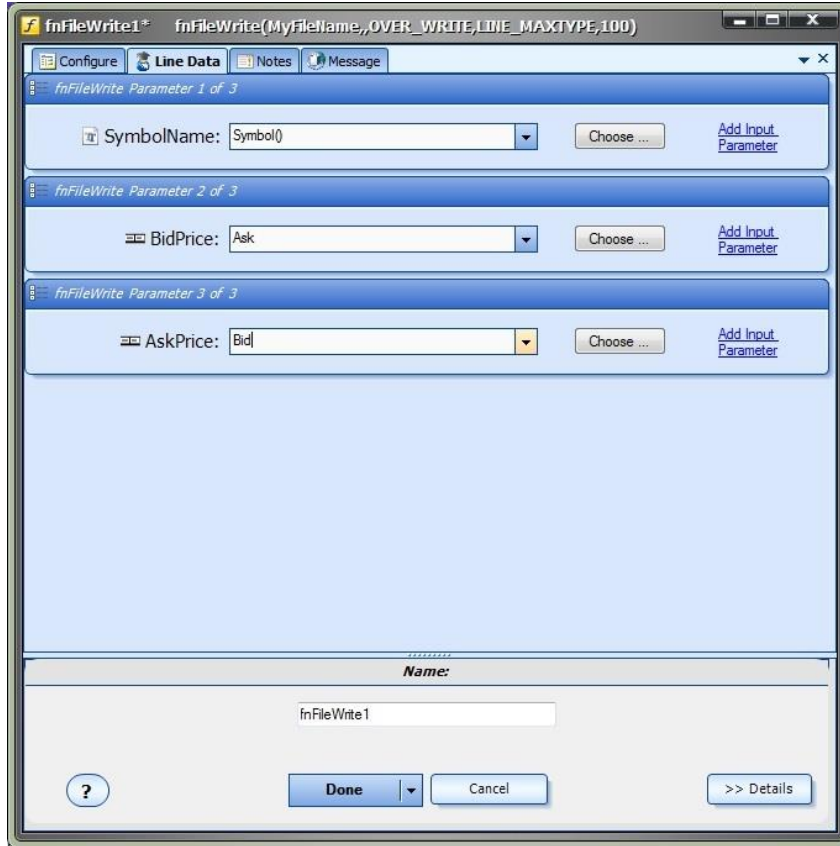
The function **fnFileWrite** is used to write a single line of to a file. The configuration of the file is defined in the [File IO System Manager](#).

The **Configure** tab contains the parameter for the file name:



Parameter Name	Data type	Description
FileName	string	The name of the file to read. MetaTrader will only read files in the MT platform "Files" folder, for example: C:\Program Files (x86)\FXCM MetaTrader 4\MQL4\Files

The **Line Data** tab contains parameters for the Column Data defined in the [File IO System Manager](#). For this example, we have added Columns named SymbolName, BidPrice and AskPrice.

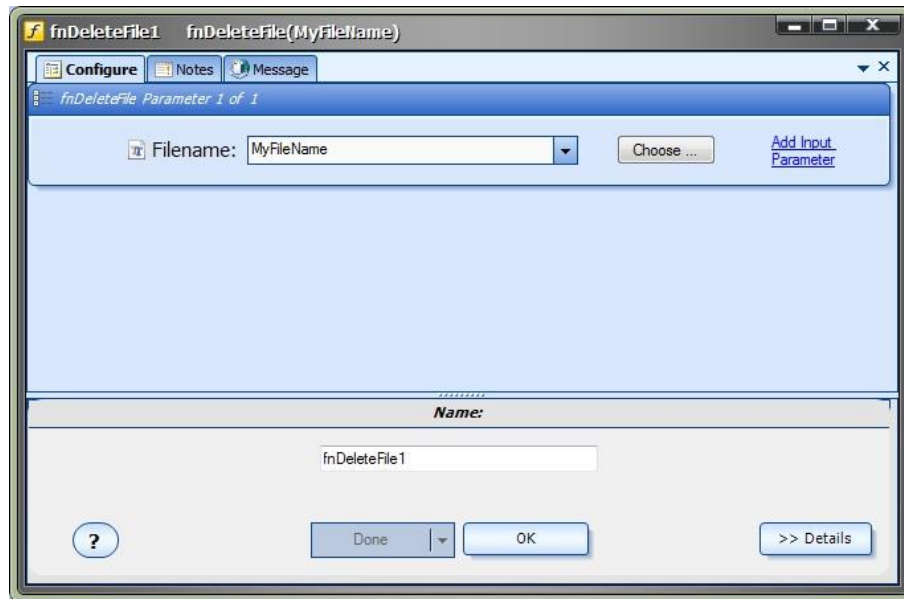


Note: There will be no parameters shown if no Columns have been added in the [File IO System Manager](#).

Parameter Name	Data type	Description
SymbolName	string	As defined by the user in in the File IO System Manager .
BidPrice	double	As defined by the user in in the File IO System Manager .
AskPrice	double	As defined by the user in in the File IO System Manager .

fnDeleteFile

The function **fnDeleteFile** is used to delete a file from the file system. (Note: a file is opened automatically when using the function [fnFileWrite](#))



MetaTrader will only read or write files in the MT platform "Files" folder.
For example: C:\Program Files (x86)\FXCM MetaTrader 4\MQL4\Files

Using the File IO Plug-in

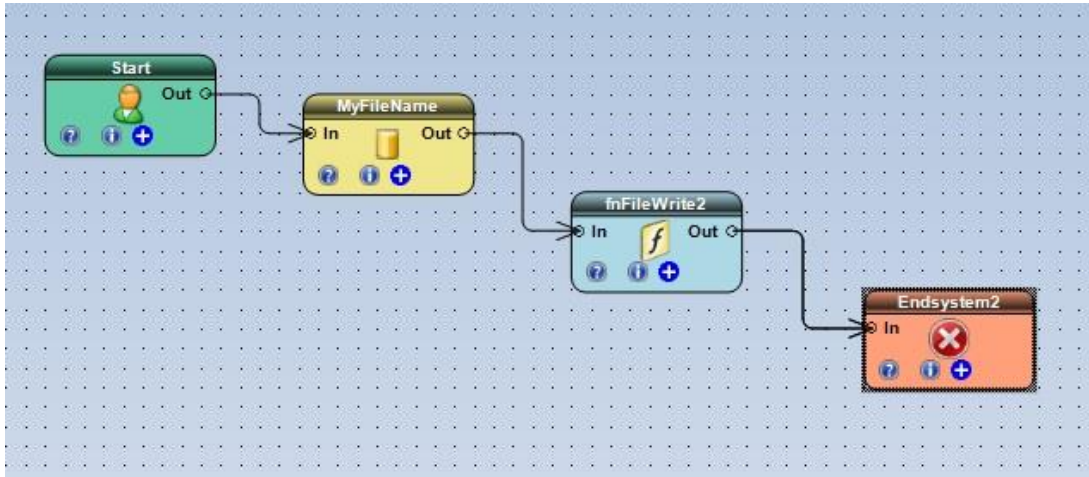
Writing Data

This drawing will write data to a file. The function **fnFileWrite2** is configured as shown in the [fnWriteFile section](#).

On each tick, the Symbol, Bid price and Ask price are written to the file. (The file name is defined in the variable MyFileName).

This is a sample of a single line:

```
EURUSD,1.1229200000,1.1232500000
```

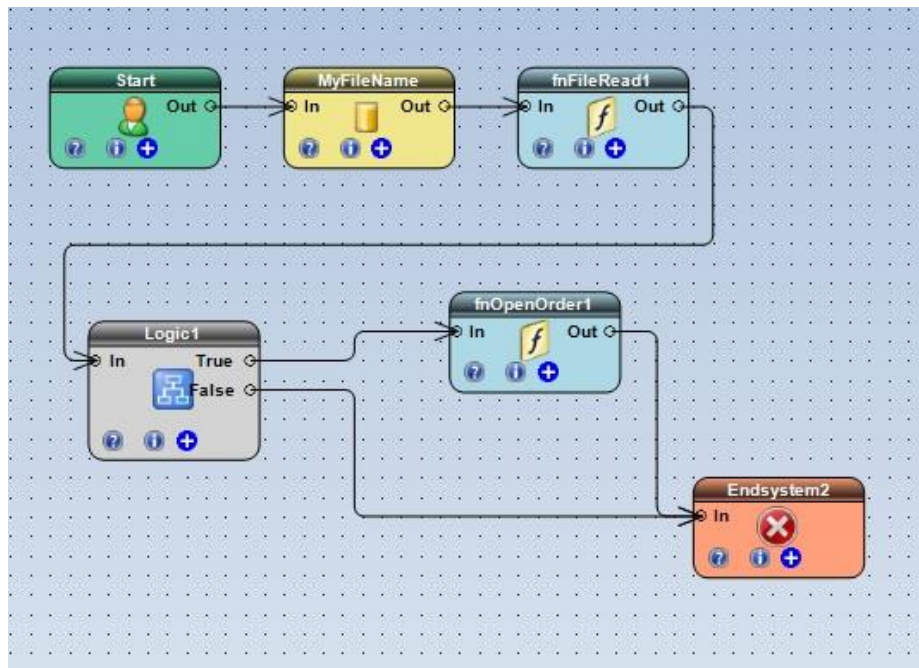


Reading Data

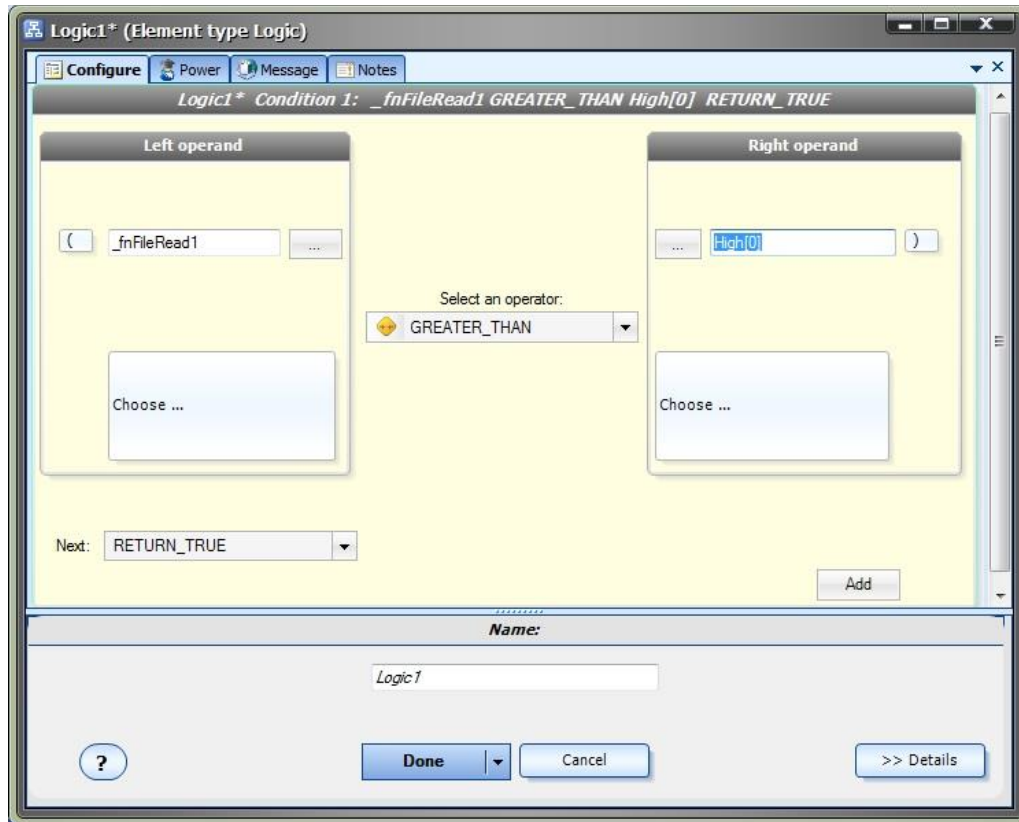
This drawing will read data from a file. The function **fnFileRead** is configured as shown in the [fnReadFile section](#).

The last line of the file is read, and the column that holds the **BidPrice** is returned as a double.

The value is stored in a variable named **_fnFileRead1**.



The variable **_fnFileRead1** is used in the Logic condition and is compared to the High price:



If the **BidPrice** (as stored in the **_fnFileRead1** variable) is greater than the High price of the current candle, the **true** path of the Logic Element is followed and a trade is opened by the function [fnOpenOrder](#).

Glossary

A

Anchor

Each Element has 1 or more Link Anchors, a small circle where a link will attach.

Ask

The Asking price for a currency, submitted by a seller.

B

Balanced parenthesis

The number of open parenthesis must match the number of close parenthesis in a logical statement. If they do not match, the expression is considered unbalanced and will produce syntax errors when compiled.

Bid

The Bid price for a currency, submitted by a buyer.

buy limit

A pending order that opens a **long** position when the price breaks **below** a level.

buy stop

A pending order that opens a **long** position when the price breaks **above** a level.

C

Caption

The top part of an Element that shows the name. The caption area is used to select an Element, and used to open a Function Drawing Element by double-clicking.

Cartesian

A system in which the location of a point is given by coordinates that represent its distances from perpendicular lines that intersect at a point called the origin. A Cartesian coordinate system in a plane has two perpendicular lines (the x-axis and y-axis).

CHART

A VTS definition used for selecting a value for a symbol or timeframe parameter. When the EA runs, it will use the value of the chart that is running on.

clone

To make an exact duplicate.

Comment

1. A text attribute that is stored with an open trade; the comment field can be viewed in the platform trade tab.
2. An MQL function that writes text to the price chart.

CSV

Comma Separated File

Custom Indicators

Technical indicators created using MQL. Stored in the "experts\indicators" folder beneath the Metatrader installation folder.

D

Data Scope

The scope of a variable element defines **where** the variable can be accessed and used.

Data Type

MQL, like all programming languages, uses data types to store information. A data type is simply a section of memory used to store a value.

The MQL data types are:

Integer (int): An integer is a whole number. A whole number means the number does not have a decimal point. For example, the number of total open trades is an integer: 0, 1, 2 etc. There is no concept of 1.5 open trades. However, the balance of a MetaTrader account is a number with a decimal point - such as 10,000.99. A number with a decimal point is called a real number in the field of mathematics. A real number data type is called a double in MQL.

Double (double): A double is a real number. A real number means the number has a decimal point.

The balance of a MetaTrader account is a real number - such as 10,000.99.

Boolean (bool): A boolean is a data type that can one of two values: true or false.

String (string): A string is one or more characters. A string value is set using double quotes. For example: message="Hello World".

Void (void): A void type is a "nothing" type. It is used in to indicate that no data type is inferred.

Date and Time (datetime): A datetime data type is used to store calendar dates and times. The data type itself is actually an integer. The current time is an integer whose value is the number of seconds elapsed since midnight January 1, 1970. (This is a common approach, used in many computer languages, to define the current time.)

Color (color): A color data type is actually an integer type. This data type is defined to make color assignments easy.

Datetime

Date and Time (datetime): A datetime data type is used to store calendar dates and times. The data type itself is actually an integer. The current time is an integer whose value is the number of seconds elapsed since midnight January 1, 1970. (This is a common approach, used in many computer languages, to define the current time.)

Default Value

The value a parameter or function will resolve to if no other value is explicitly set.

deinit

A special MQL function that is called when an EA is stopped.

Delete

To delete an Element from a drawing, select the Element and press the delete key of your keyboard.

To delete an item from the Toolbox, select the item, right-click and select delete. If the item is not set as read-only, it may be deleted.

To delete a System and all of its Elements, on the Welcome screen, right-click on the system and Select delete.

Done

Clicking the Done button will save all changes and close the window.

Down Levels

In a trading grid, a term used to describe the area of a price chart below the initial starting price position.

For example, if the grid begins at 1.3450:

- The price range from 1.3649 to 1.3620 may be considered level 1. (a 20 point range)
- The price range from 1.3619 to 1.3600 may be considered level 2.

These are considered "down" levels because they are located **below** the starting grid price of 1.3450.

Drawing Mode

A System created using the EA Wizard can be edited in Drawing mode by selecting Drawing mode in the properties window.

Drawing Pad

The center window in the VTS application, used for dragging, dropping, connecting and configuring Elements.

E

EA

Expert Advisor - an automated trading system that runs on the Metatrader platform.

ECN

Electronic Communication Network

EMA

Exponential Moving Average

Error

An **error** prevents an EA from being built. A **warning** allows an EA to be built but provides information that may be critical.

ex4

The extension of an executable Expert Advisor.

Expert Advisor

Expert Advisor - an automated trading system that runs on the Metatrader platform.

F

FOREX

FOReign Currency **EX**change

I

init

A special MQL function that is called when an EA is started..

L

LWMA

Linear weighted moving average or LWMA is a moving average which assigns more value to current prices and is thus more responsive to latest price trends.

M

MA

Moving average

MagicNumber

A unique number that allows an EA to identify the positions it has opened.

Main System Drawing

The Main System Drawing is the drawing that has the same name as the system. It is the entry point for the system.

market order

An order that is placed at the current market price.

mathematical expression

A expression that calculates a value; it may use any mathematic operation as well as any of the MQL math functions.

Members Area

The Members Area is located at www.iExpertAdvisor.com. It is a members-only area of the website where members can access the latest version of their purchased products.

MetaEditor

An editor application; part of the Metatrader platform. The MetaEditor is used to create and edit mq4 files.

Metalang

The executable application used to convert an mq4 file into an ex4 file.

MetaTrader

The MetaTrader trading platform is an intuitive, easy to use Forex trading platform.

mode

VTS supports to modes of operation: **Wizard** mode and **Drawing** mode.

Moving average

A measure of the average price or exchange rate of a currency pair over a specific time frame.

mq4

MQL

Meta Query Language

N

Next value

In a logical expression, the value that connects two adjacent logical conditions or terminates a logical expression. The available values are: AND OR RETURN_TRUE

O

Operand

In a logical condition, the left and right values that are compared.

Operator

In a logical condition, the center value that operates (performs a comparison) on the left and right operands.

Order Ticket

A unique number assigned to a trade when it is opened.

Order Type

Type of order.

Market orders can be:

OP_BUY
OP_SELL.
Pending orders can be:
OP_BUYLIMIT
OP_SELLLIMIT
OP_BUYSTOP
OP_SELLSTOP

P

Parameter

An input value to a function that is used by the function to perform an operation or calculation.

PIP

Pip or "percentage in point," refers to the last digit of a currency price.

Platform

The trading system that communicates with a broker's system to open, close and modify trades.

Platform function

The built-in MQL functions.

Platform Tools

The Metatrader tools used by VTS to build Expert Advisors.

Platform Tools Folder

The path to the location of the Metatrader platform.

point

Same as PIP: "percentage in point," refers to the last digit of a currency price.

Price constants

Used to get the price value of a candle (or bar):

PRICE_CLOSE
PRICE_OPEN
PRICE_HIGH
PRICE_LOW
PRICE_MEDIAN
PRICE_TYPICAL
PRICE_WEIGHTED

prototype

A function prototype is a declaration of a function that omits the function body but does specify the function's return type, name, number of argument and their types. While a function definition specifies what a function does, a function prototype can be thought of as specifying its interface.

R

Read Only

An Element, folder or a file that can be read but can not be changed.

Read Write

An Element, folder or a file that can be read and written to (changed).

Return Type

The type of data that is return from a function. See Data Type.

S

Save

To commit changes to the file system

Save As

Save an element or system to a new name. The original still exists; the original item is *cloned* and then the name is changed.

Scope

The scope of a variable element defines *where* the variable can accessed and used.

Script

A MetaTrader script is created using MQL code and/or VTS Elements. Scripts are located in their own folder underneath the "experts" folder and appear under their own menu in the MetaTrader platform.

Note: a Script executes just once, while an Expert Advisor executes each time a new price value is received.

sell limit

A pending order that opens a short position when the price breaks above a level.

sell stop

A pending order that opens a short position when the price breaks below a level.

shift

The bar or candle that is currently forming (furthest to the right on a candlestick chart) is numbered zero. Each bar to the left increases in number. A series, or array, of Indicator values also follow this concept of number 0 starting at the far right with each number increasing to the left. The **Shift** of an indicator value is the index in the array of the value, where shift=0 is the latest value and shift=1 is the value from the last period.

SMA

Simple Moving Average

SMMA

Smoothed Moving Average

Sound Files

The files found in the "sounds" folder of the MT platform that can be used by the MQL function PlaySound.

Strategy Template

A Strategy Template is type of "New System" that appears on the Welcome screen. It creates a new copy of the Strategy Template system with a new name.

syntax

The spelling and grammar of a programming language. Computers are inflexible machines that understand what you type only if you type it in the exact form that the computer expects. The expected form is called the syntax.

syntax error

Errors created when MQL code does not follow the correct form (or syntax)

System

The high-level object that contains all elements and produces an Expert Advisor.

System Functions

All of the functions used within a System; organized in a special folder in the function toolbox.

System Drawing

Same as Main System Drawing. The Main System Drawing is the drawing that has the same name as the system. It is the entry point for the system.

System Logics

All of the logics used within a System; organized in a special folder in the logics toolbox.

System Variables

All of the variables used within a System; organized in a special folder in the variable toolbox.

T**Terminal**

The name of the executable file that is the Metatrader platform: terminal.exe

Tick

The minimum upward or downward movement in the price of a currency pair. The term "tick" also refers to the change in the price of a currency pair from trade to trade.

Ticket

A unique number assigned to a trade when it is opened.

Trade Signal

A signal to open or close a buy or sell trade.

Trading Grid

A trading system that opens trades based on price movement through different levels of a grid. The total profit or loss of the entire set of trades is calculated on each tick and the grid is closed if either a profit target or a maximum loss is reached.

Type

Same as Data Type. A data type is simply a section of memory used to store a value.

U**Up Levels**

In a trading grid, a term used to describe the area of a price chart above the initial starting price position.

For example, if the grid begins at 1.3450:

- The price range from 1.3651 to 1.3670 may be considered level 1. (a 20 point range)
- The price range from 1.3671 to 1.3690 may be considered level 2.

These are considered "up" levels because they are located **above** the starting grid price of 1.3450.

V**VTS**

Visual Trader Studio. A visual application for creating automated trading systems, including Expert Advisors.

W**Warning**

An **error** prevents an EA from being built. A **warning** allows an EA to be built but provides information that may be critical.