Event Manager Plug-in

Requires VTS-Connect minimum version **4.0.0.88**

The ***Event Manager Plug-in*** allows any VTS function to be executed when an event occurs. Events include when an EA starts, when an EA stops, when a timer expires and on Chart events.

<u>What is a Plug-in?</u>

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.

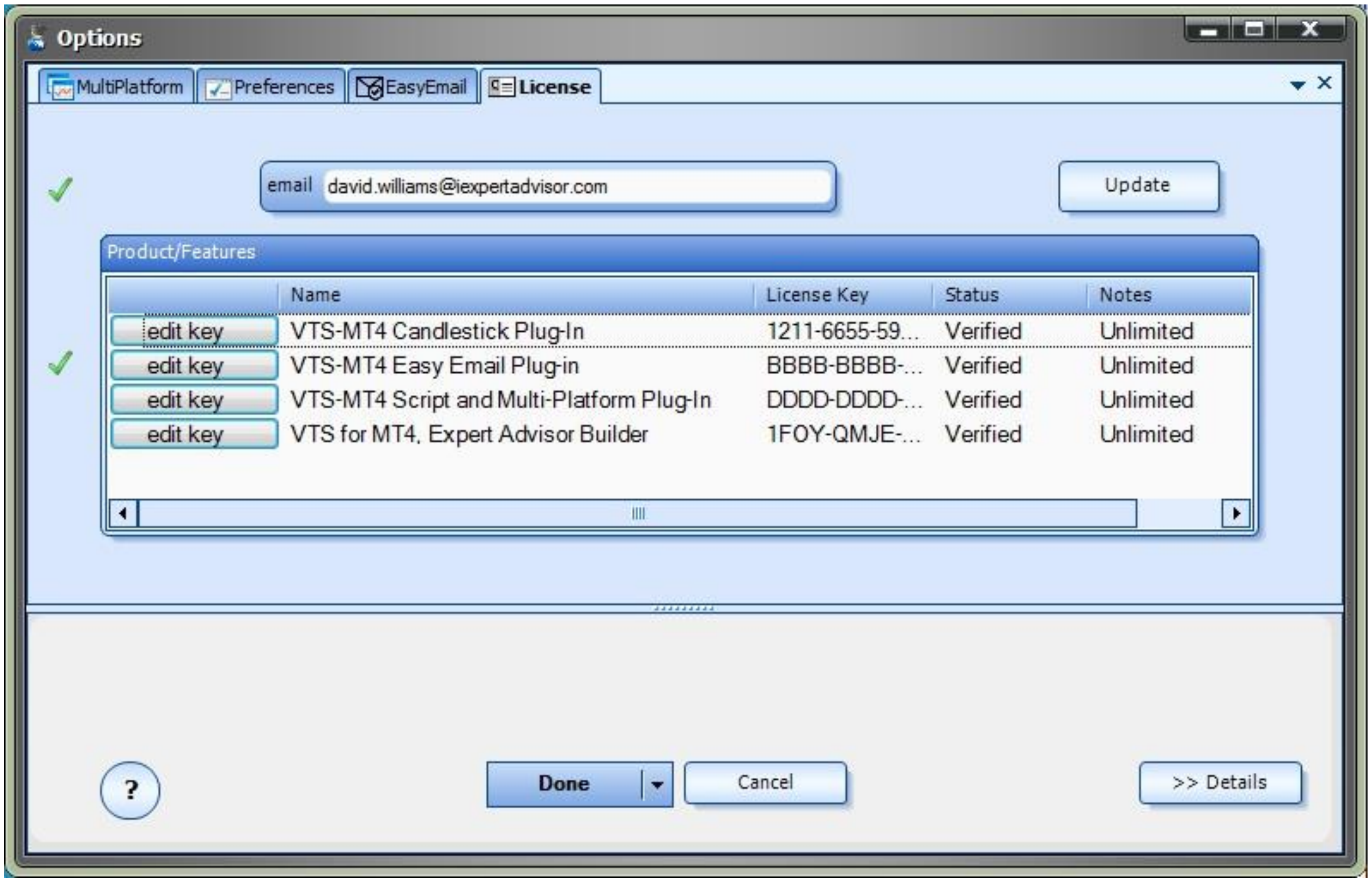
Contents

Enable the Event Manager Plug-in

You must enter your License key to enable the ***Event 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.



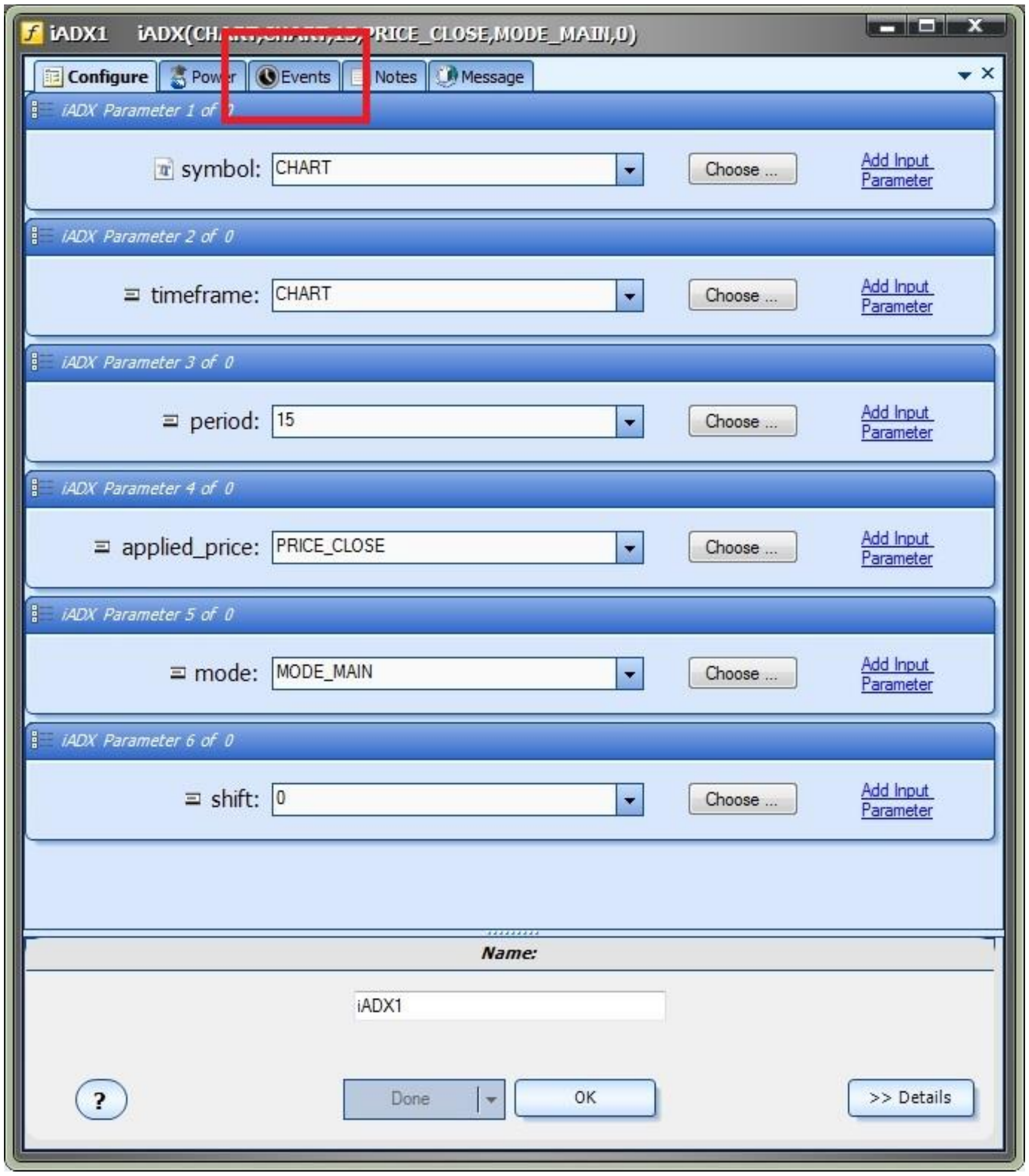
email: david.williams@iexpertadvisor.com

	Name	License Key	Status	Notes
edit key	VTS-MT4 Candlestick Plug-In	1211-6655-59...	Verified	Unlimited
edit key	VTS-MT4 Easy Email Plug-in	BBBB-BBBB-...	Verified	Unlimited
edit key	VTS-MT4 Script and Multi-Platform Plug-In	DDDD-DDDD-...	Verified	Unlimited
edit key	VTS for MT4, Expert Advisor Builder	1FOY-QMJE-...	Verified	Unlimited

Using the Event Manager Plug-in

The ***Event Manager Plug-in*** allows any VTS function to be executed when an event occurs. Events include when an EA starts, when an EA stops, when a timer expires and many Chart events, such as mouse clicks and object creation.

- **Event Manager** configuration is found on the "Event" tab of all Functions.
- All VTS Elements are configured by clicking the (+) on the bottom of the Element.
- Clicking the (+) opens the Function configuration window. Note the "Event" tab.



iADX Parameter 1 of 0
symbol: CHART

iADX Parameter 2 of 0
timeframe: CHART

iADX Parameter 3 of 0
period: 15

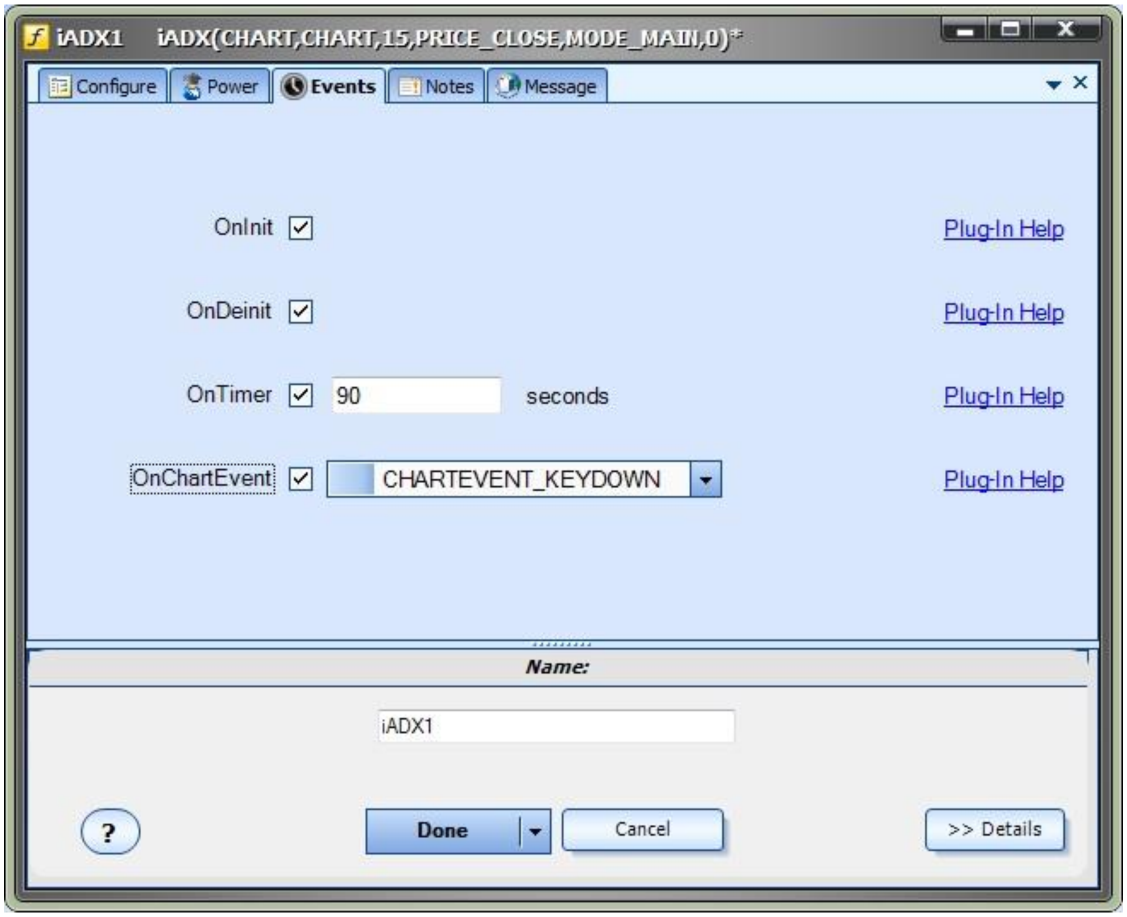
iADX Parameter 4 of 0
applied_price: PRICE_CLOSE

iADX Parameter 5 of 0
mode: MODE_MAIN

iADX Parameter 6 of 0
shift: 0

Name: iADX1

Clicking the "Event" tab opens the **Event** Configuration:


iADX1 iADX(CHART,CHART,15,PRICE_CLOSE,MODE_MAIN,0)*

OnInit ✓ Plug-In Help

OnDeinit ✓ Plug-In Help

OnTimer ✓ 90 seconds Plug-In Help

OnChartEvent ✓ CHARTEVENT_KEYDOWN Plug-In Help

Name: iADX1

The Event configuration window options are:

Event	Description
OnInit	When **OnInit** is checked, the function will execute when the EA is attached to a chart and whenever the EA's input parameters are reconfigured.
OnDeInit	When **OnDeInit** is checked, the function will execute when the EA is removed from a chart.
OnTimer	When **OnTimer** is checked, the function will execute every **N seconds**. Note, the **seconds** value is global and is the same for functions of the EA.
OnChartEvent	When **OnChartEvent** is checked, the function will execute when certain Chart Events occur. The Chart Events are listed below.

Chart Events supported by the VTS EA-Builder

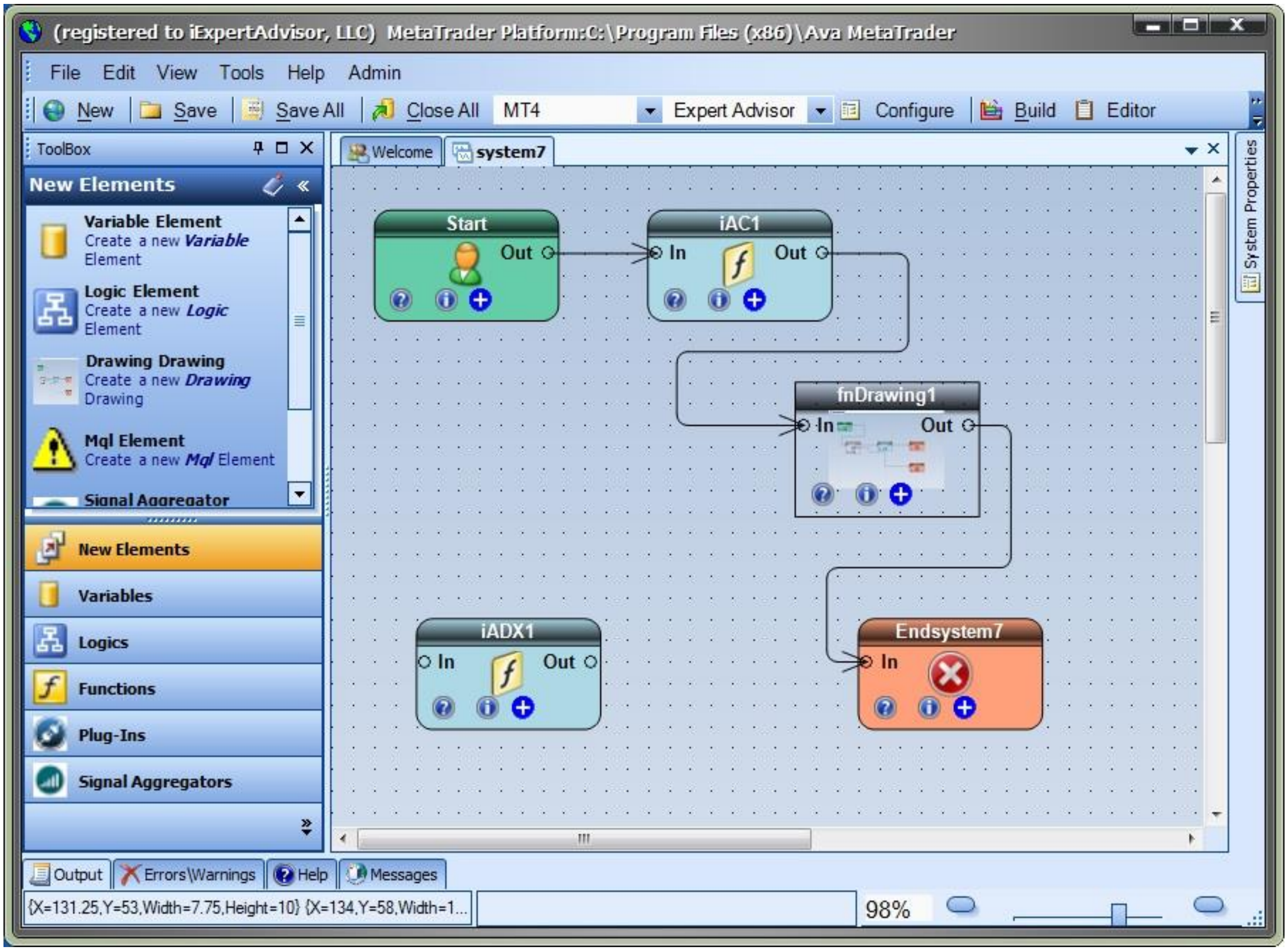
Event	**Description**
CHARTEVENT_KEYDOWN	event of a keystroke, when the chart window is focused;
CHARTEVENT_MOUSE_MOVE	mouse move events and mouse click events (if CHART_EVENT_MOUSE_MOVE=true is set for the chart);
CHARTEVENT_OBJECT_CREATE	event of graphical object creation (if CHART_EVENT_OBJECT_CREATE=true is set for the chart);
CHARTEVENT_OBJECT_CHANGE	event of change of an object property via the properties dialog;
CHARTEVENT_OBJECT_DELETE	event of graphical object deletion (if CHART_EVENT_OBJECT_DELETE=true is set for the chart);
CHARTEVENT_CLICK	event of a mouse click on the chart;
CHARTEVENT_OBJECT_CLICK	event of a mouse click in a graphical object belonging to the chart;
CHARTEVENT_OBJECT_DRAG	event of a graphical object move using the mouse;
CHARTEVENT_OBJECT_ENDEDIT	event of the finished text editing in the entry box of the Label Edit graphical object;
CHARTEVENT_CHART_CHANGE	event of chart changes;

Event Manager on a VTS Drawing

Normally a VTS Element must be on a Drawing ***and*** connected by a Link to be executed.

- Connection is **not** mandatory for a Function to execute due to an Event.
- The Function may or may not be connected by a Link in order to execute by an Event.

In the Drawing below, the Functions **iAC1** and **fnDrawing1** can be configured to execute on Events, as well as the **unconnected** Function **iADX1**.


(registered to iExpertAdvisor, LLC) MetaTrader Platform:C:\Program Files (x86)\Ava MetaTrader
File Edit View Tools Help Admin
New Save Save All Close All MT4 Expert Advisor Configure Build Editor
ToolBox
New Elements
Variable Element Create a new Variable Element
Logic Element Create a new Logic Element
Drawing Drawing Create a new Drawing Drawing
Mql Element Create a new Mql Element
Signal Aggregator
New Elements
Variables
Logics
Functions
Plug-Ins
Signal Aggregators
Welcome
system7
Start
Out
iAC1
In
Out
fnDrawing1
In
Out
iADX1
In
Out
Endsystem7
In
System Properties
Output
Errors\Warnings
Help
Messages
{X=131.25,Y=53,Width=7.75,Height=10} {X=134,Y=58,Width=1...
98%

MQL Event Documentation

Much of the VTS EA-Builder is a pass through of the underlying MetaTrader MQL functionality.

The MQL documentation should be referred to for the latest information about features that are offered via MetaTrader such as Event handling.

The MQL Event Documentation is below. Please refer to the documentation provided by your MetaTrader platform for the most current information. MQL Help is found via the MetaEditor.

Event Handling Functions

The MQL4 language provides processing of some predefined events. Functions for handling these events must be defined in a MQL4 program; function name, return type, composition of parameters (if there are any) and their types must strictly conform to the description of the event handler function.

The event handler of the client terminal identifies functions, handling this or that event, by the type of return value and type of parameters. If other parameters, not corresponding to below descriptions, are specified for a corresponding function, or another return type is indicated for it, such a function will not be used as an event handler.

OnStart

The OnStart() function is the Start event handler, which is automatically generated only for running scripts. It must be of void type, with no parameters:

void OnStart();

For the OnStart() function, the int return type can be specified.

OnInit

The OnInit() function is the Init event handler. It must be of void or int type, with no parameters:

void OnInit();

The Init event is generated immediately after an Expert Advisor or an indicator is downloaded; The OnInit() function is used for initialization. If OnInit() has the int type of the return value, the non-zero return code means unsuccessful initialization, and it generates the Deinit event with the code of deinitialization reason REASON_INITFAILED.

OnInit() function execution result is analyzed by the terminal's runtime subsystem only if the program has been compiled using #property strict.

To optimize input parameters of an Expert Advisor, it is recommended to use values of the ENUM_INIT_RETCODE enumeration as the return code.. During initialization of an Expert Advisor before the start of testing you can request information about the configuration and resources using the TerminalInfoInteger() function.

ENUM_INIT_RETCODE

Identifier	Description

INIT_SUCCEEDED	Successful initialization, testing of the Expert Advisor can be continued. This code means the same as a null value – the Expert Advisor has been successfully initialized in the tester.
INIT_FAILED	Initialization failed; there is no point in continuing testing because of fatal errors. For example, failed to create an indicator that is required for the work of the Expert Advisor. This return value means the same as a value other than zero - initialization of the Expert Advisor in the tester failed.
INIT_PARAMETERS_INCORRECT	This value means the incorrect set of input parameters. The result string containing this return code is highlighted in red in the general optimization table. Testing for the given set of parameters of the Expert Advisor will not be executed

The OnInit() function of the void type always denotes successful initialization.

OnDeinit

The OnDeinit() function is called during deinitialization and is the Deinit event handler. It must be declared as the void type and should have one parameter of the const int type, which contains the code of deinitialization reason. If a different type is declared, the compiler will generate a warning, but the function will not be called.

```
void OnDeinit(const int reason);
```

The Deinit event is generated for Expert Advisors and indicators in the following cases:

- before reinitialization due to the change of a symbol or chart period, to which the mql4 program is attached;
- before reinitialization due to the change of input parameters;
- before unloading the mql4 program.

OnTick

The NewTick event is generated for Expert Advisors only when a new tick for a symbol is received, to the chart of which the Expert Advisor is attached. It's useless to define the OnTick() function in a custom indicator or script, because the NewTick event is not generated for them.

The Tick event is generated only for Expert Advisors, but this does not mean that Expert Advisors required the OnTick() function, since not only NewTick events are generated for Expert Advisors, but also events of Timer, BookEvent and ChartEvent are generated. It must be declared as the void type, with no parameters:

```
void OnTick();
```

OnTimer

The OnTimer() function is called when the Timer event occurs, which is generated by the system timer only for Expert Advisors and indicators - it can't be used in scripts. The frequency of the event occurrence is set when subscribing to notifications about this event to be received by the EventSetTimer() function.

You can unsubscribe from receiving timer events for a particular Expert Advisor using the EventKillTimer()

function. The function must be defined with the void type, with no parameters:

void OnTimer();

It is recommended to call the EventSetTimer() function once in the OnInit() function, and the EventKillTimer() function should be called once in OnDeinit().

Every Expert Advisor, as well as every indicator works with its own timer and receives events only from it. As soon as the mql4 program stops operating, the timer is destroyed forcibly, if it was created but hasn't been disabled by the EventKillTimer() function.

OnTester

The OnTester() function is the handler of the Tester event that is automatically generated after a history testing of an Expert Advisor on the chosen interval is over. The function must be defined with the double type, with no parameters:

double OnTester();

The function is called right before the call of OnDeinit() and has the same type of the return value - double. OnTester() can be used only in the testing of Expert Advisors. Its main purpose is to calculate a certain value that is used as the Custom max criterion in the genetic optimization of input parameters.

In the genetic optimization descending sorting is applied to results within one generation. I.e. from the point of view of the optimization criterion, the best results are those with largest values (for the Custom max optimization criterion values returned by the OnTester function are taken into account). In such a sorting, the worst values are positioned at the end and further thrown off and do not participate in the forming of the next generation.

OnChartEvent

OnChartEvent() is the handler of a group of ChartEvent events:

- CHARTEVENT_KEYDOWN — event of a keystroke, when the chart window is focused;
- CHARTEVENT_MOUSE_MOVE — mouse move events and mouse click events (if CHART_EVENT_MOUSE_MOVE=true is set for the chart);
- CHARTEVENT_OBJECT_CREATE — event of graphical object creation (if CHART_EVENT_OBJECT_CREATE=true is set for the chart);
- CHARTEVENT_OBJECT_CHANGE — event of change of an object property via the properties dialog;
- CHARTEVENT_OBJECT_DELETE — event of graphical object deletion (if CHART_EVENT_OBJECT_DELETE=true is set for the chart);
- CHARTEVENT_CLICK — event of a mouse click on the chart;
- CHARTEVENT_OBJECT_CLICK — event of a mouse click in a graphical object belonging to the chart;
- CHARTEVENT_OBJECT_DRAG — event of a graphical object move using the mouse;
- CHARTEVENT_OBJECT_ENDEDIT — event of the finished text editing in the entry box of the LabelEdit graphical object;
- CHARTEVENT_CHART_CHANGE — event of chart changes;
- CHARTEVENT_CUSTOM+n — ID of the user event, where n is in the range from 0 to 65535.
- CHARTEVENT_CUSTOM_LAST — the last acceptable ID of a custom event (CHARTEVENT_CUSTOM +65535).

The function can be called only in Expert Advisors and indicators. The function should be of void type with 4 parameters:

```
void OnChartEvent(const int id, // Event ID
const long& lparam, // Parameter of type long event
const double& dparam, // Parameter of type double event
const string& sparam // Parameter of type string events
);
```

For each type of event, the input parameters of the OnChartEvent() function have definite values that are required for the processing of this event. The events and values passed through these parameters are listed in the table below.

Event	Value of the id parameter	Value of the lparam parameter	Value of the dparam parameter	Value of the sparam paramete
Event of a keystroke	CHARTEVENT_KEYDOWN	code of a pressed key	Repeat count (the number of times the keystroke is repeated as a result of the user holding down the key)	The string value of a bit mask describing the status of keybo buttons
Mouse events (if property CHART_EVENT_MOUSE_MOVE=true is set for the chart)	CHARTEVENT_MOUSE_MOVE	the X coordinate	the Y coordinate	The string value of a bit mask describing the status of mouse buttons
Event of graphical object creation (if CHART_EVENT_OBJECT_CREATE=true is set for the chart)	CHARTEVENT_OBJECT_CREATE	—	—	Name of the created graphica object
Event of change of an object property via the properties dialog	CHARTEVENT_OBJECT_CHANGE	—	—	Name of the modified graphic object
Event of graphical object deletion (if CHART_EVENT_OBJECT_DELETE=true is set for the chart)	CHARTEVENT_OBJECT_DELETE	—	—	Name of the deleted graphica object
Event of a mouse click on the chart	CHARTEVENT_CLICK	the X coordinate	the Y coordinate	—
Event of a mouse click in a graphical object belonging to the chart	CHARTEVENT_OBJECT_CLICK	the X coordinate	the Y coordinate	Name of the graphical object, which the event occurred
Event of a graphical	CHARTEVEN	—	—	Name of the moved graphical

object dragging using the mouse	T_OBJECT_DRAG			object
Event of the finished text editing in the entry box of the LabelEdit graphical object	CHARTEVENT_OBJECT_ENDEDIT	—	—	Name of the LabelEdit graphi object, in which text editing h completed
Event of chart Changes	CHARTEVENT_CHART_CHANGE	—	—	—
ID of the user event under the N number	CHARTEVENT_CUSTOM+N	Value set by the EventChartCustom() function	Value set by the EventChartCustom() function	Value set by the EventChartCustom() function

OnCalculate

The OnCalculate() function is called only in custom indicators when it's necessary to calculate the indicator values by the Calculate event. This usually happens when a new tick is received for the symbol, for which the indicator is calculated. This indicator is not required to be attached to any price chart of this symbol.

The OnCalculate() function must have a return type int.

```
int OnCalculate (const int rates_total, // size of input time series
const int prev_calculated, // bars handled in previous call
const datetime& time[], // Time
const double& open[], // Open
const double& high[], // High
const double& low[], // Low
const double& close[], // Close
const long& tick_volume[], // Tick Volume
const long& volume[], // Real Volume
const int& spread[] // Spread
);
```

Parameters of open[], high[], low[] and close[] contain arrays with open prices, high and low prices and close prices of the current time frame. The time[] parameter contains an array with open time values, the spread[] parameter has an array containing the history of spreads (if any spread is provided for the traded security). The parameters of volume[] and tick_volume[] contain the history of trade and tick volume, respectively.

To determine the indexing direction of time[], open[], high[], low[], close[], tick_volume[], volume[] and spread[], call ArrayGetAsSeries(). In order not to depend on default values, you should unconditionally call the ArraySetAsSeries() function for those arrays, which are expected to work with.

The first rates_total parameter contains the number of bars, available to the indicator for calculation, and corresponds to the number of bars available in the chart.

We should note the connection between the return value of OnCalculate() and the second input parameter prev_calculated. During the function call, the prev_calculated parameter contains a value returned by OnCalculate() during previous call. This allows for economical algorithms for calculating the custom indicator in order to avoid repeated calculations for those bars that haven't changed since the previous run of this function.

For this, it is usually enough to return the value of the rates_total parameter, which contains the number of bars in the current function call. If since the last call of OnCalculate() price data has changed (a deeper history downloaded or history blanks filled), the value of the input parameter prev_calculated will be set to zero by the terminal.

To understand it better, it would be useful to start the indicator, which code is attached below.

Indicator Example:

```
#property indicator_chart_window
#property indicator_buffers 1
//---- plot Line
#property indicator_label1 "Line"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrDarkBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- indicator buffers
double LineBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function |
//+------------------------------------------------------------------+
int OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
return(INIT_SUCCEEDED);
}
//+------------------------------------------------------------------+
//| Custom indicator iteration function |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
const int prev_calculated,
const datetime& time[],
const double& open[],
const double& high[],
const double& low[],
const double& close[],
const long& tick_volume[],
const long& volume[],
const int& spread[])
{
//--- Get the number of bars available for the current symbol and chart period
int bars=Bars(Symbol(),0);
Print("Bars = ",bars,", rates_total = ",rates_total,", prev_calculated = ",prev_calculated);
Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- return value of prev_calculated for next call
return(rates_total);
}
```