Custom Indicators

User Guide

December 4, 2012

Draft Version 1.0

Table of Contents

General

OEC Trader Custom Indicators provides a programming framework and the Integrated Development Environment (IDE) for the development custom indicators. The framework is .NET based. So, it uses Microsoft C# in the integrated development environment. It also supports all .NET languages in the compiled form. Additionally, it is compatible with EasyLanguage ™.

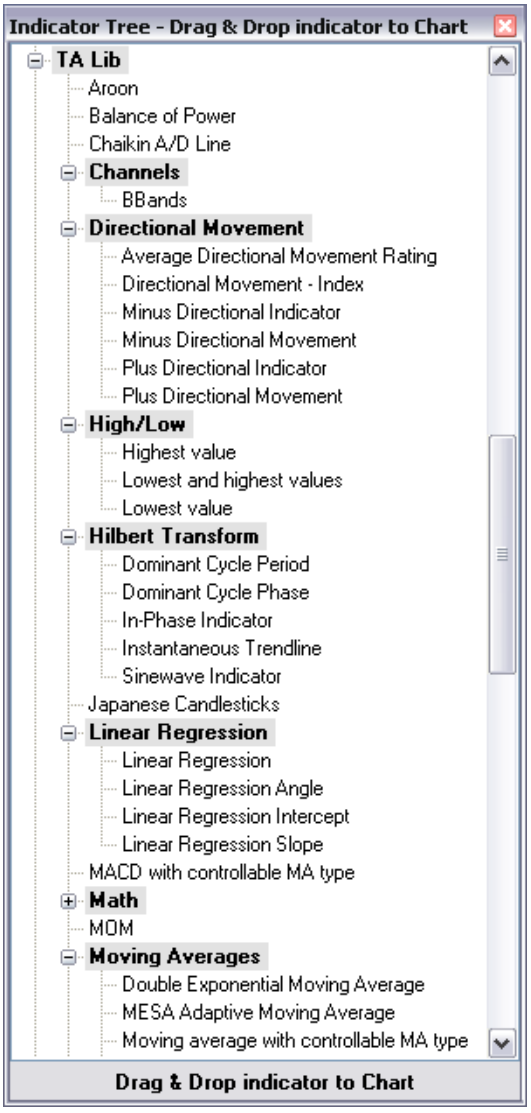
The plug-in has two new types of views that can be accessed through the View menu of OECTrader: Custom Indicator Library Manager and the Code Editors. The first one shows the list of available and editable custom indicators, and the second one is used to edit the source code of indicators.

Additionally, this plug-in provides access to freeware from the TA Lib library.

Notes: First, download the plug-in from the OEC Website. Then, download the Custom Indicators User Guide.

Technical Analysis Library

Figure 1 Indicator Tree



The Technical Analysis Library, TA Lib library, (http://ta-lib.org/) provides common functions for the technical analysis of financial market data. There are more than 150 technical analysis indicators such as ADX, MACD, RSI, Stochastic, Bollinger Bands, etc. and the candlestick pattern recognition.

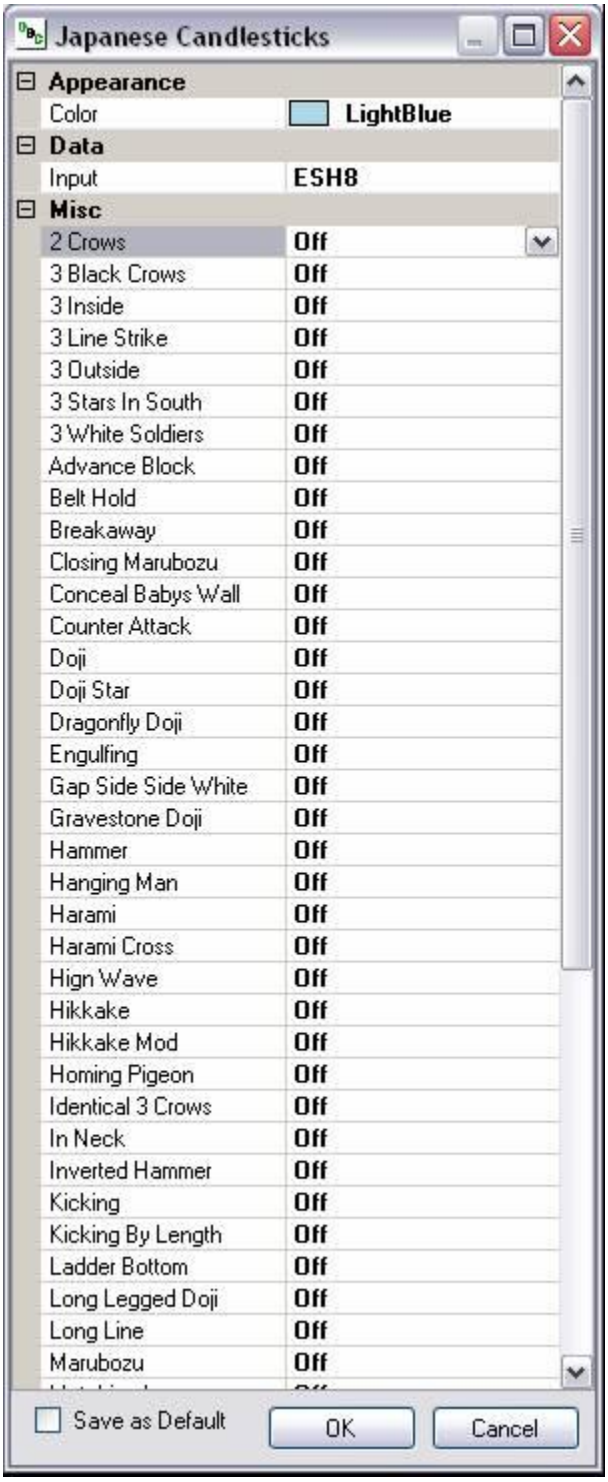
OEC Custom Indicators are a bridge between this library and OEC Charts.

For a more advanced customized version of this library – replace the installed TA-Lib-Core.dll with the newer version and restart OECTrader.

This plug-in analyzes the functions of the DLL to construct the corresponding indicators. Refer to the window *Indicator Tree* to the left.

1. To display the Indicator Tree window, in the OECTrader command bar, select View>Charts>Indicators>Indicator Tree.

Figure 2 Japanese Candlesticks



The TA Lib contains the set of functions for the candlestick pattern analyze.

All of these functions are grouped into one OEC indicator: *Japanese Candlesticks* in the TA Lib category of indicators.

Refer to the window *Japanese Candlesticks* on the left.

1. Set *On* to enable the required patterns to highlight the proper bars.

Custom Analyze Techniques

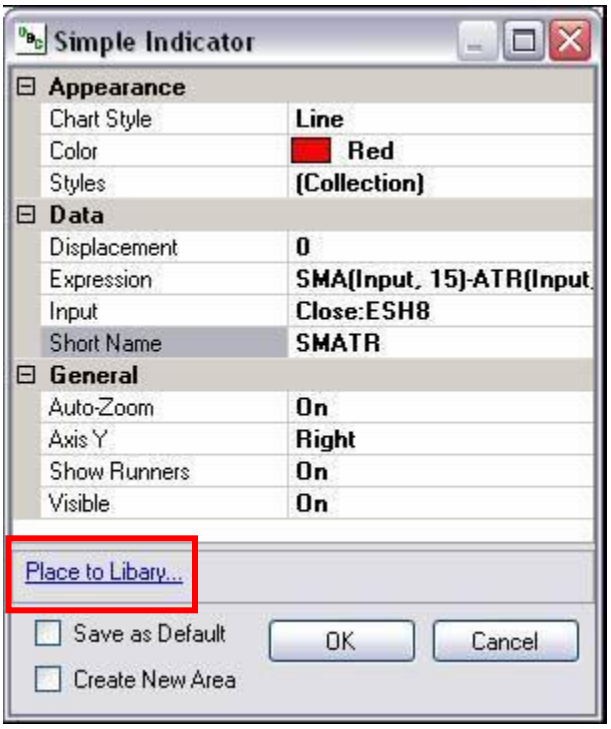
The Custom Indicators provides three types of simple custom analyze techniques in the *Custom Indicators* category that do not require programming skills:

- Simple Indicator
- Coloring, and
- Show Cases

For all of the above, type the formulas of indicator directly in property dialogs.

Simple Indicator

Figure 3 Simple Indicator

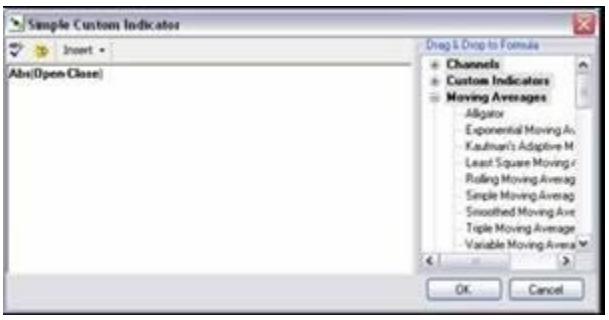


1. Enter the formula in the property Expression and press Close to the results of calculations. Press "…" to open the expanded window with a simple assistance. Refer to the window *Simple Indicator* to the left.

2. Use the Simple Indicator for easy formulas that are used only once.

3. To re-use this indicator, place it in the Custom Indicator Library. Press Place to Library hyperlink in Properties Dialog of Simple Indicator (Refer to the Figure left.). This opens an editor of Rapid Custom Indicators.

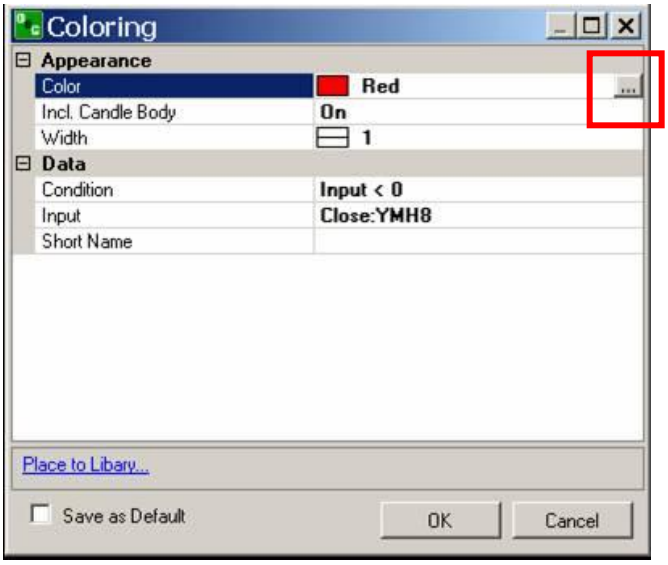
Figure 4 Simple Custom Indicator

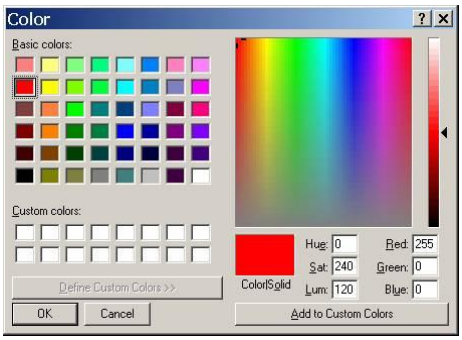


Note: This is the expanded window for expression editing.

Coloring

Figure 5 Coloring





This analyze technique highlights a bar in a color to reference the specified conditions. Refer to the window *Coloring* to the left.

1. Left click on the Red Color to display the *Coloring* Window with the Color Palate tool at the end of the row.
2. Click on the color tool icon to display the Color Palate window. Refer to the lower *Color* window.
3. Select a color and click Ok.

Access Show Cases through Chart

Figure 6 Chart-Indicators

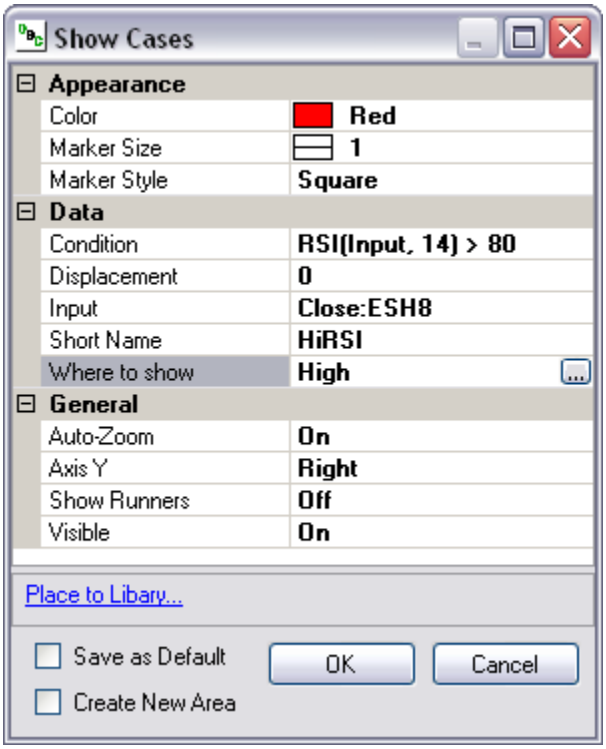
1. Open a Chart, left click on Indicators, select Custom Indicators, and select Show Cases to display the *Show Cases* window. Refer to Figure 8.

Figure 7 Chart-Insert

2. Or, in the Chart, click on Insert, and select Custom Indicators to display another drop-down menu. Refer to the Figure at the left.

Show Cases

Figure 8 Show Cases



This analyze technique displays certain points in history that satisfy the specified conditions via drawing specified markers in specified positions. Refer to Figure *Show Cases* on the left.

Syntax of simple expressions

The expressions of these simple custom indicators have a simplified syntax on the base of C#. Samples of expressions include:

"SMA(Close, 14)", "EMA(Close, 14) – ATR(Input, 15)", "RSI(Input, 30) < 20", etc.

Operators

- Arithmetic operators: +, -, /, *
- Comparison operators: <, >, == (equal), != (not equal), >= (greater or equal), <= (lesser or equal)
- Logical operations: & (and), | (or)
- A special *IF* construction: <condition_expression> ? <true_expression> : <false_expression>. For example, *Abs(Input) < 10.0 ? 0 : Input* shows zero instead of input value, if its absolute value is less than 10.0.

Keywords for input data

- **Use the Input** word to refer to time series that is selected in the property Input of indicator.
- **Use Open, High, Low, Close, Volume** – for historical data.

Figure 9 Field Definitions

Field	Description
TickSize	Tick size of corresponding symbol. For example, 0.25 for ES
DPP	Dollars per point value of contract. For example, 50 for ES
PriceScale	Price scale
PointValue	DPP / PriceScale
MinMove	PriceScale * TickSize
AskPrice	Current ask price (real-time!)
BidPrice	Current bid price (real-time!)
LastPrice	Current last price (real-time!)
PrevClose	Settlement price (real-time!)

Mathematical Functions

Figure 10 Mathematical Definitions

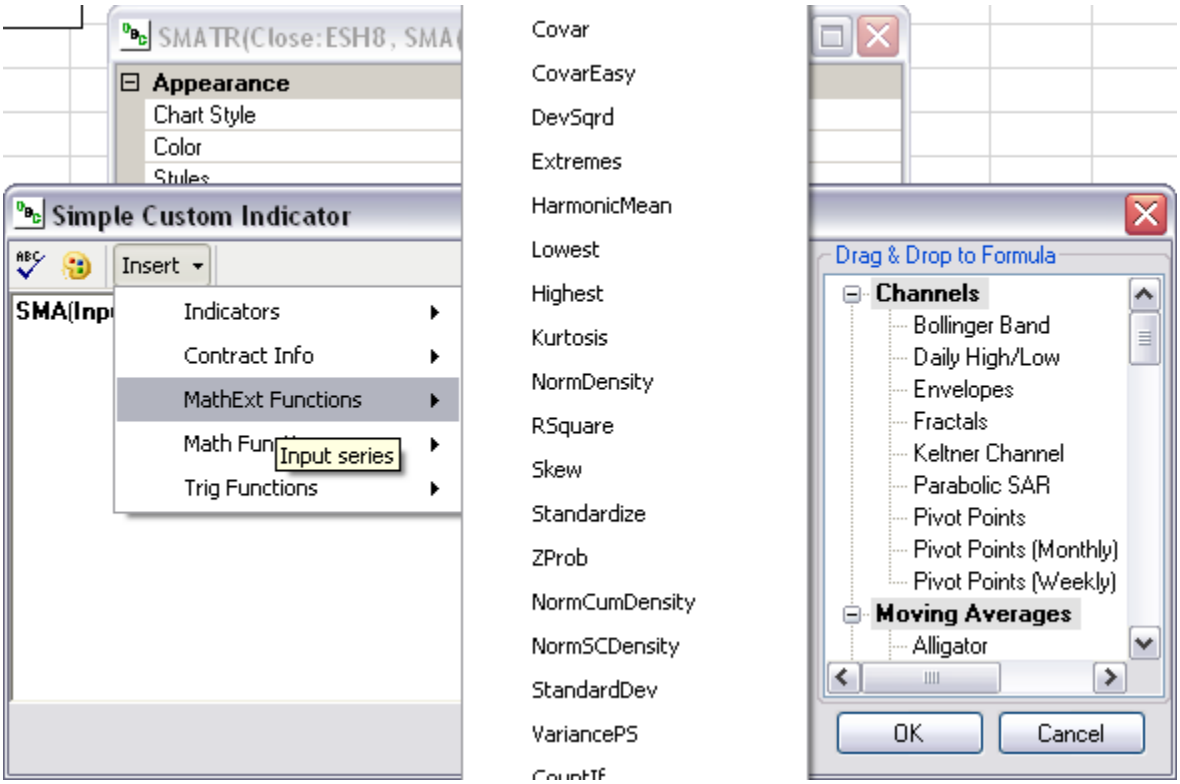
Field	Description
Abs(val)	Absolute value of val. Example: Abs(-5) returns 5.
CrossesOver(sa1, sa2)	Determines if the series sa1 crosses over the series sa2 on the bar under consideration. Example: CrossesOver(Close, SMA(Close, 15)) returns “true”, if Close prices cross over its simple moving average.
CrossesUnder(sa1, sa2)	Determines if the series sa1 crosses under the series sa2 on the bar under consideration. Example: CrossesUnder(Close, SMA(Close, 15)) returns “true”, if Close prices cross under its simple moving average.
Frac(number)	Returns the fractional part of a number.
Max(val_1, val_2, etc)	Returns the larger of parameters
Max2(val_1, val_2, val_3, etc)	Returns the second larger of parameters
Min(val_1, val_2, etc)	Returns the smaller of parameters
Min2(val_1, val_2, val_3, etc)	Returns the second smaller of parameters
Mod(val_1, val_2)	Returns the remainder of division
Neg(val)	Returns the negative value
Pos(val)	Returns the absolute value
Random(max_val)	Returns a random number between 0.0 and the specified maximum max_val
Sign(val)	Returns a value indicating the sign of a number.
Square(val)	Returns the square of a specified number.

Sum(val_1, val_2, val_3, etc)	Returns the sum of parameters
Acos(val),Asin(val), Atan(val), Cos(val), Cosh(val), Sin(val), Sinh(val), Tan(val), Tanh(val)	Trigonometry functions
Ceiling(val)	Returns the smallest integer greater than or equal to the specified number.
Exp(val)	Returns **e** raised to the specified power.
Floor(val)	Returns the largest integer less than or equal to the specified number.
Log(val)	Returns the logarithm of a specified number.
Log10(val)	Returns the base 10 logarithm of a specified number.
Pow(val, power)	Returns a specified number raised to the specified power.
Round(val, prec)	Rounds a value to the nearest integer or specified number of decimal places.
Sqrt(val)	Returns the square root of a specified number.
Truncate(val)	Calculates the integral part of a number.
AvgDeviation(series, length)	Calculates average deviation for 'length' items
Correlation(seriesA, seriesB, length)	Calculates the correlation of two series
CoefficientR(seriesA, seriesB, length)	Calculates the Pearson product moment correlation coefficient *R*
Covar(seriesA, seriesB, length)	Calculates the covariance of two series
DevSqrd(series, length)	Calculates the sum of squares of deviations from average value
HarmonicMean(series, length)	Calculates the harmonic mean
Kurtosis(series, length)	Calculates the Kurtosis of a series

NormDensity(series, length)	Calculates the normal density
RSquare(seriesA, seriesB, length)	Calculates the square of Pearson product moment correlation coefficient *R*
Skew(series, length)	Calculates the skewness of a series
Standardize(series, length, numDevs)	Returns a normalized value on the base of distribution of a series
NormCumDensity(series, length)	Calculates the normal density
StandardDev(series, length,dataType)	Calculates the standard deviation
CountIf(series, length)	Returns non-zero values in series
Factorial(num)	Returns the factorial of num
LinearRegAngle(series, length)	Calculates an angle of linear regression line in the current point
LinearRegSlope(series, length)	Calculates a slope of linear regression line in the current point
LinearRegValue(series, length, bar)	Returns a value of linear regression line 'bar' bars ago

The full list of supported mathematical and EasyLanguage functions are in the *Insert* menu of the expression editor. Refer to the Figure below.

Figure 11 MathExt Functions Menu



Most of these functions are reflected from the corresponding EasyLanguage functions.

Figure 12 Functions-Indicators

AC(Series, FastPeriod, SlowPeriod)	Acceleration/Deceleration Oscillator
ASlashD(Series, Window)	Accumulation/Distribution
Alligator(Series, FastOffset, MedianOffset, SlowOffset, FastPeriod, MedianPeriod, SlowPeriod)	Alligator. Example: Alligator(InputData, 3, 5, 8, 5, 8, 13)[0] – returns the fast series
ADX(Series, Period)	Average Directional Index
ATR(Series, Window)	Average True Range

AO(Series, FastPeriod, SlowPeriod)	Awesome Oscillator
BBand(Series, Window, NumStdDev)	Bollinger Band
CCI(Series, Window)	Commodity Channel Index
HL(Series)	Daily High/Low
DPO(Series, Window)	Detrended Price Oscillator
EMV(Series)	Ease of Movement
Envelopes(Series, Window, Percent)	Envelopes
EMA(Series, Window)	Exponential Moving Average
FO(Series, Window)	Forecast Oscillator
Fractals(Series)	Fractals
KAMA(Series, Window)	Kaufman's Adaptive Moving Average
KAMAFilter(Series, Window, FilterPercentage)	Kaufman's Adaptive Moving Average Filter
Keltner(Series, Window, NumDev)	Keltner Channel
LSMA(Series, Window)	Least Square Moving Average
MACD(Series, SignalWindow, SlowWindow, FastWindow)	MACD
MASS(Series, Window, AveragePeriod)	Mass Index
Median(Series)	Median Price
Momentum(Series, Window)	Momentum
MFI(Series, Window)	Money Flow Index
NVI(Series, StartValue)	Negative Volume Index

OBV(Series)	On Balance Volume
SAR(Series, Maximum, Step)	Parabolic SAR
Perf(Series)	Performance
PPs(Series)	Pivot Points
PVI(Series, StartValue)	Positive Volume Index
PO(Series, Percentage, MovingAverageType, SlowWindow, FastWindow)	Price Oscillator
PVT(Series)	Price Volume Trend
RoC(Series, Window)	Rate Of Change
RSI(Series, Window)	Relative Strength Index
RMA(Series, K)	Rolling Moving Average
SMA(Series, Window, Offset)	Simple Moving Average
SMMA(Series, Window, Offset)	Smoothed Moving Average
StdDev(Series, Window)	Standard Deviation
Stoch(Series, PeriodD, PeriodK)	Stochastic
LinearR(Series, Window, Multiplier)	Tangent of Linear Regression
TRIMA(Series, Window)	Triangular Moving Average
Triple(Series, FastPeriod, MedianPeriod, SlowPeriod)	Triple Moving Average
TRIX(Series, Window)	TRIX
Typical(Series)	Typical Price
VMA(Series, K)	Variable Moving Average

Volatility(Series, Window)	Volatility
CHV(Series, Window, AveragePeriod)	Volatility Chaikins
VR(Series, Window)	Volatility Ratio
VOSC(Series, Percentage, LongPeriod, ShortPeriod)	Volume Oscillator
WghtClose(Series)	Weighted Close
WMA(Series, Window)	Weighted Moving Average
PercentR(Series, Window)	William's %R
WCCI(Series, FastPeriod, SlowPeriod)	Woodies CCI

Use this sample for the Indicators to return multiple series:

BBand(Close, 15, 2)[0]

where [0] refers to the first line (Top). To refer to the second one, the Bottom, [1] is used.

The full list of functions-indicators are in the Indicator Tree of expression editor.

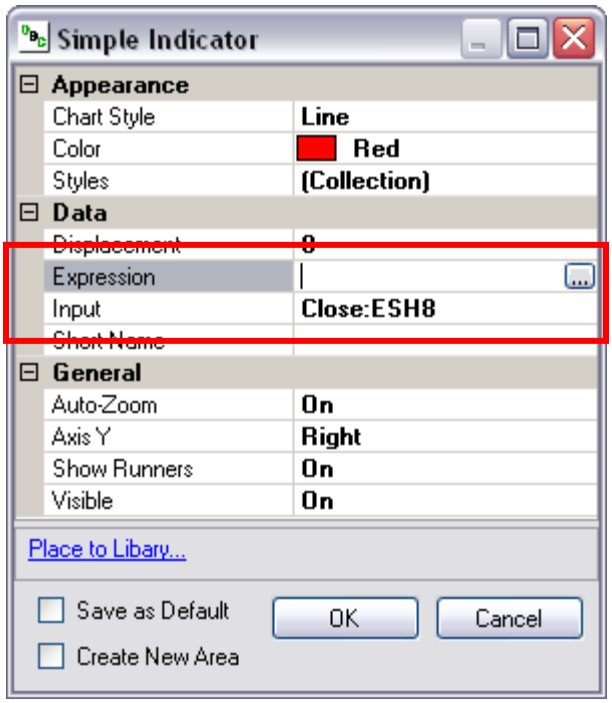
Figure 13 Drag and Drop to Formula

Refer to the Figure to the left.

1. Use the scroll bar to move down the list.
2. To open a folder, click on the plus (+) symbol next to the item.

The best way to use these functions-indicators is to access the item from the drag & drop indicator tree inside the expression editor:

Figure 14 Simple Indicator-Expression



1. Open the Property Dialog for the Simple Custom Indicator and press [...] in the line Expression field to display the dialog window. Refer to the red box to the left.

2. In Figure 15, select the required indicator, drag it to the editing area and drop the item.

Figure 15 Drag & Drop to Formula window

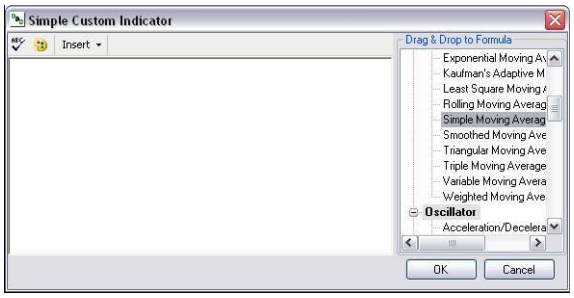
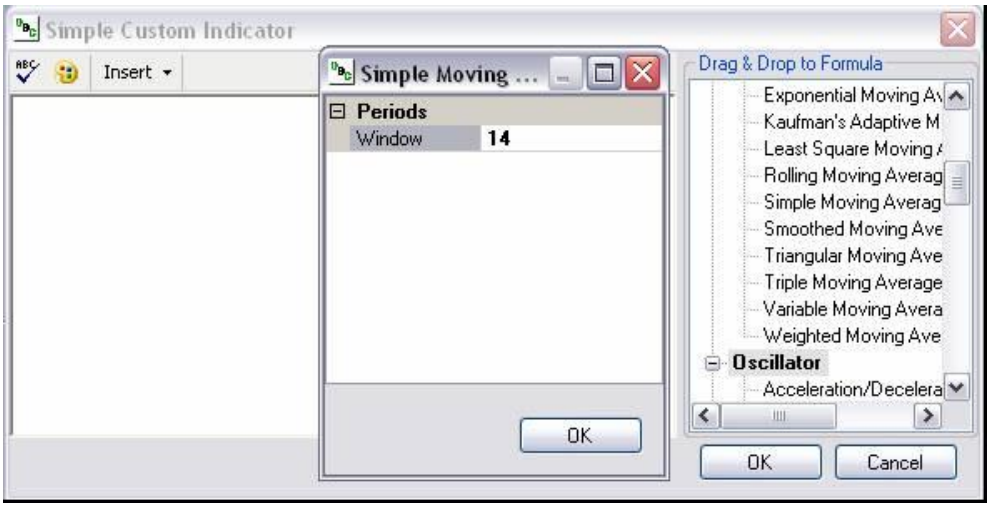
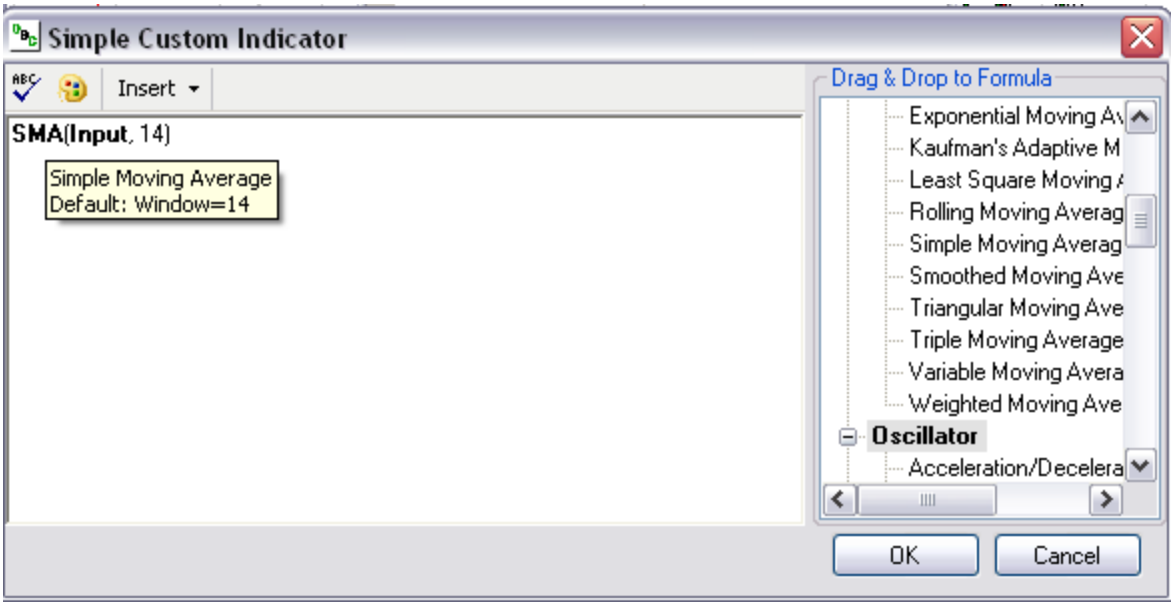


Figure 16 Simple Moving



3. Edit the parameters of the indicator in the opened Property Dialog and press OK. Refer to the Figure on the left.

Figure 17 Simple Custom Indicator



4. The result displays in the left panel of the Simple Custom Indicator window above.

5. Enter a *+* and repeat the drag and drop, then edit the manipulations for the Average True Range. Refer to the Simple Custom Indicator window below.

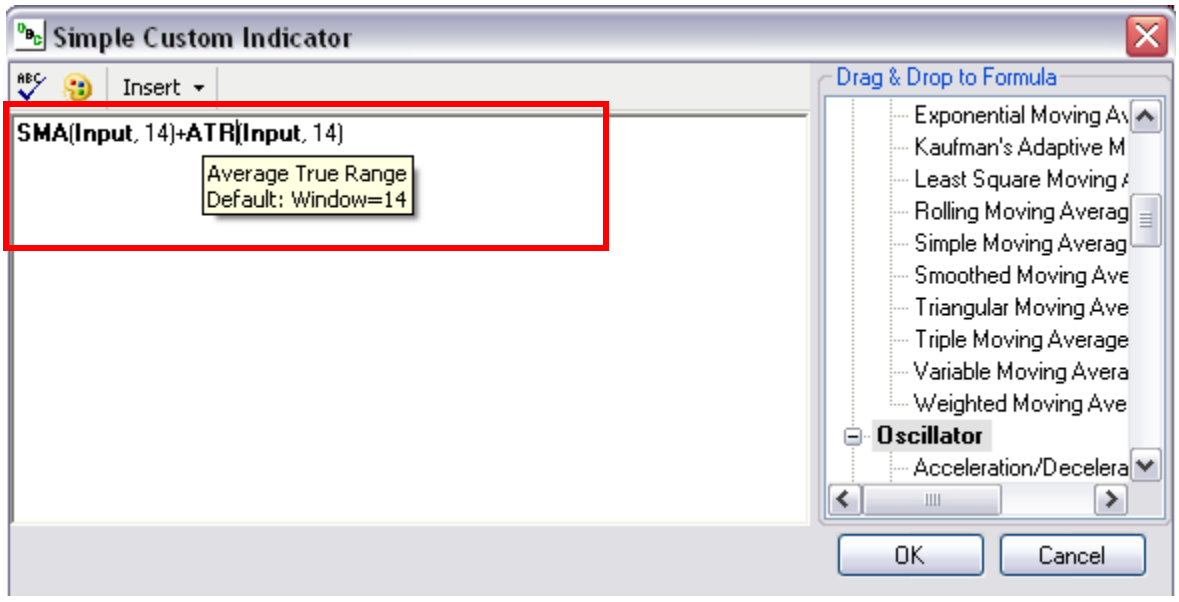
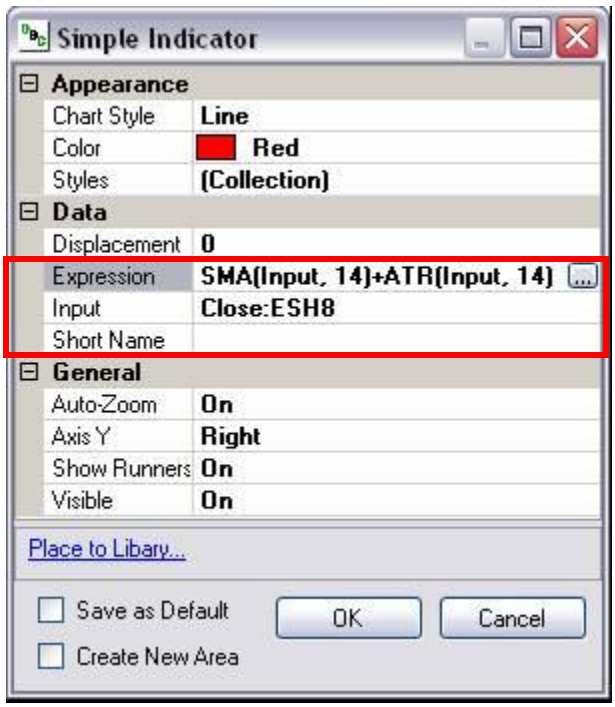


Figure 18 Final Result Display



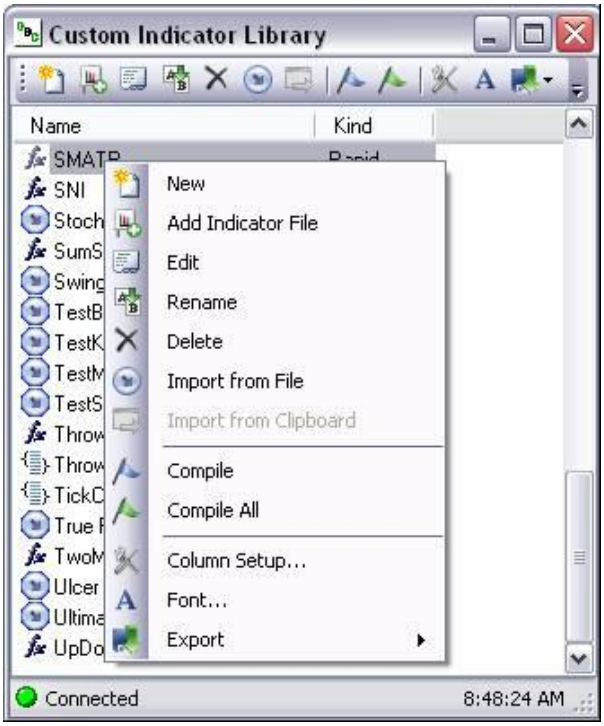
Advanced Custom Indicator Library

Besides the simple analyze techniques, the plug-in manages a library of advanced custom indicators.

Figure 19 View-Customer Indicators –Custom Indicator Library

1. To access the Custom Indicator Library from the OECTrader command bar, click on View, select Custom Indicators, and select Custom Indicator Library to display the Custom Indicator Library window. Refer to the top Figure to the left.

Figure 20 Custom Indicator Library

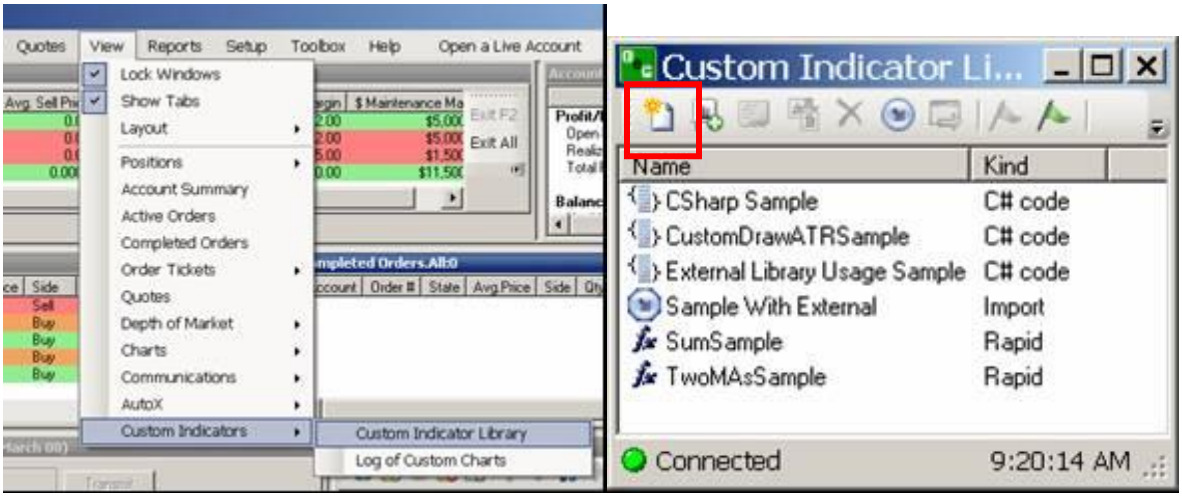


1. To display the drop down menu, right click on the Custom Indicator Library window. Refer to the lower Figure to the left.

Create a New Indicator from the View

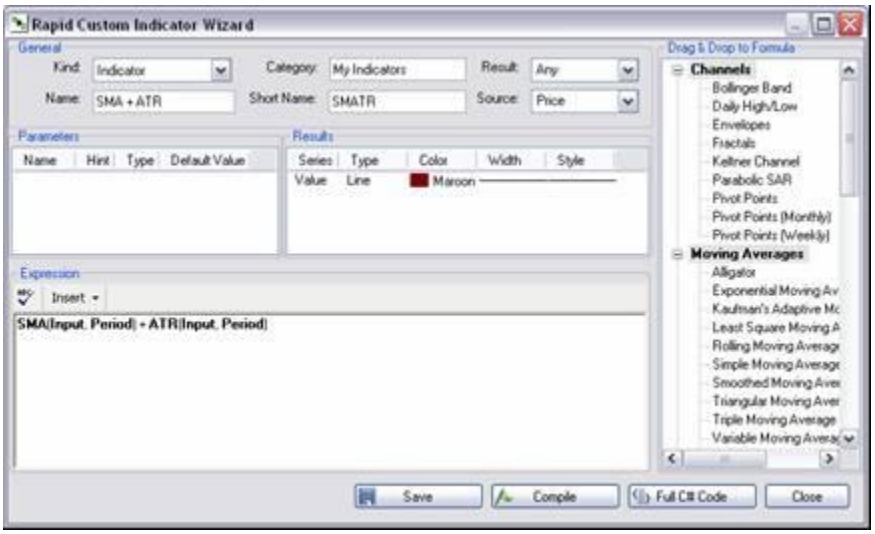
1. To access the Rapid Custom Indicator Wizard window, click View in the command bar, select Custom Indicators and select Custom Indicator Library to display the new window. Refer to the Figures below.

Figure 21 View>Custom Indicators>Custom Indicator Library



From the toolbar on the Custom Indicator Library window (right Figure above) select and click the first icon (Create a new indicator) to display the wizard. Refer to the Figure below.

Figure 22 Rapid Custom Indicator Wizard

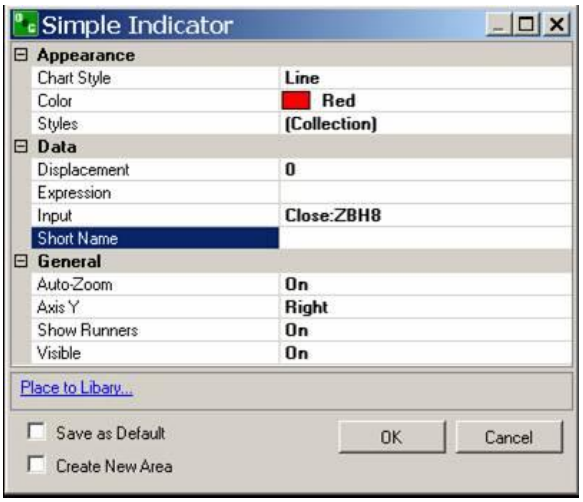


Create a Simple Indicator from a Chart

Figure 23 Chart Access

1. From a Chart, select Indicators, and select Simple Indictor to display the Simple Indicator window in the Figure below.

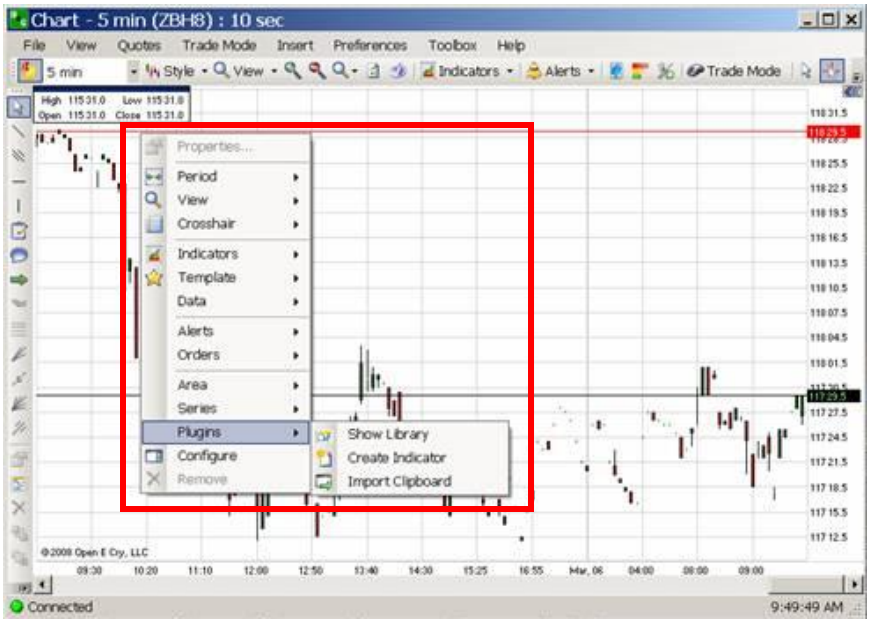
Figure 24 Simple Indicator



Create a Custom Indicator from the Chart with the Wizard

1. Open a Chart, right click anywhere in the Chart to display the Properties drop-down menu in the Figure below.

Figure 25 Chart-Right Click Access



2. Select Plug-ins and then select Create Indicator to display the Wizard window in the Figure below.

Figure 26 Rapid Custom Indicator Wizard

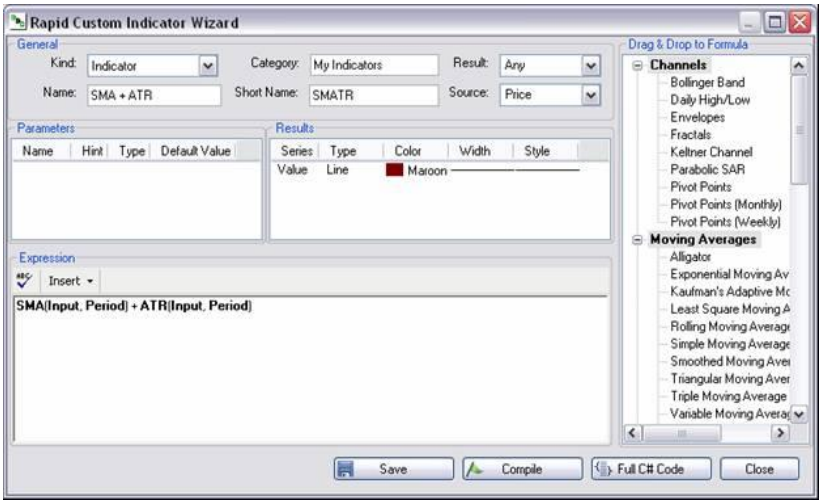
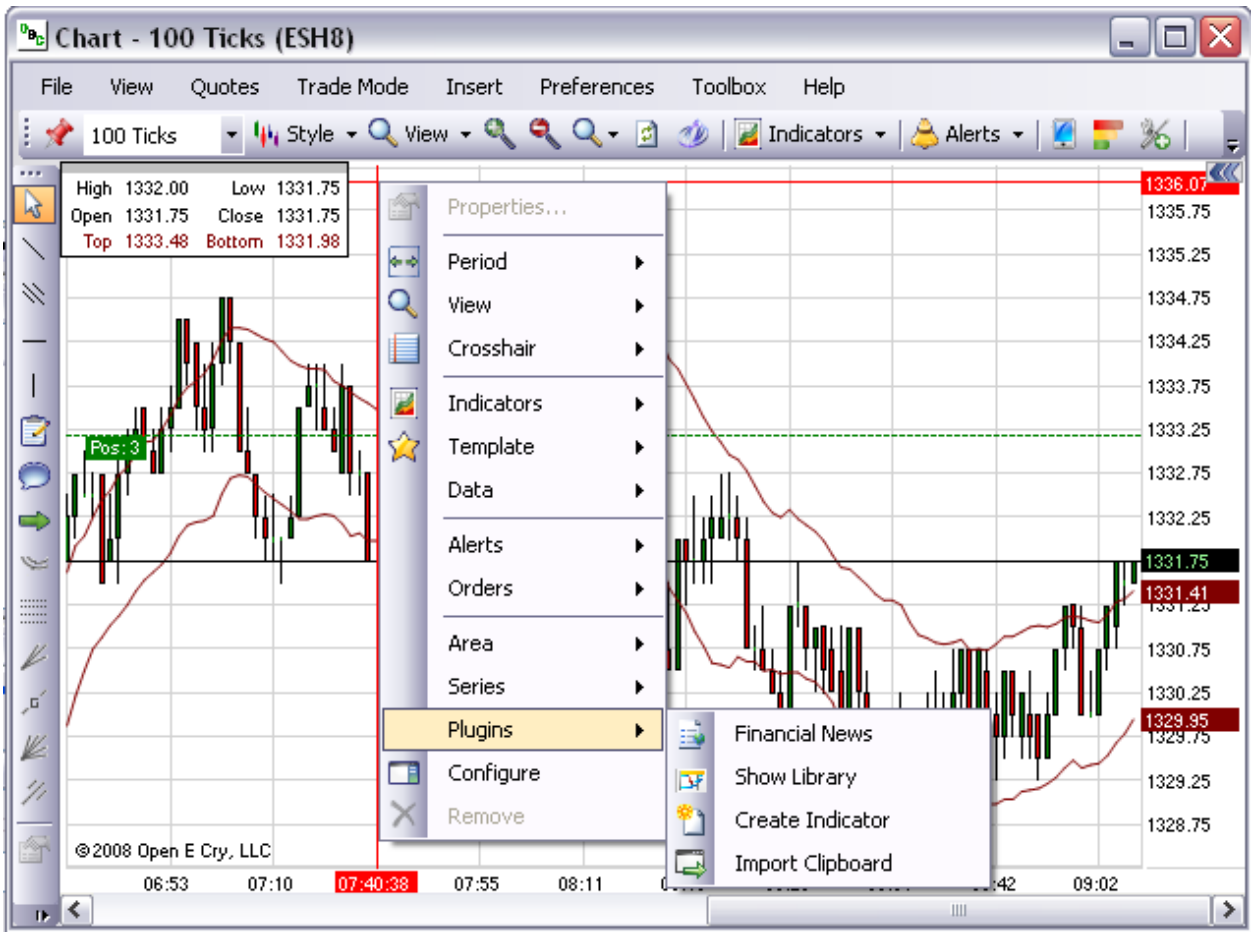


Figure 27 Chart-Right Click Access to Indicator



The library represents files of certain types from the folder CustomIndicators.

All files are stored in one of the open formats: text files for C# code and imported indicators, XML – for Rapid Indicators and binary .NET assembly for precompiled indicators.

The structure of the folder next:

- **<Program Files>**
  - **OEC** – home folder of OEC software
    - **s** – root folder that contains the plug-in and other external DLLs
      - **CustomIndicators** – It is the home folder of Custom Indicator Library. C# code is stored right here.

- **Bin** – It contains verified and compiled indicators, including 3rd party developed indicators.
- **Rapid** – This folder contains Rapid Custom Indicators in files with *.ci.xml extension.
- **Import** – This folder contains the imported EasyLanguage files with extension *.EL in open text format.
- **Help** – This folder contains help documentation.
- **Samples** – These are the source codes of samples.

The icon of listed items represents the type of source codes (imported, C# or Rapid Indicator) and the status of the verification and compilation.

The window provides the next commands. Use:

- **New** to open a wizard of indicators.
- **Edit** to open editor for selected file. Type of editor will be selected according to type of file: Rapid Custom Indicator wizard, IDE for C# or EasyLanguage ™.
- **Rename** to rename selected file.
- **Delete** to delete selected file.
- **Import File** to import EasyLanguage ™ source code from a file in text format.
- **Import Clipboard** to import the current content of Windows clipboard as EasyLanguage ™ code.
- **Compile** to verify and compile selected file.
- **Compile All** to verify and compile all files.

Rapid Custom Indicators

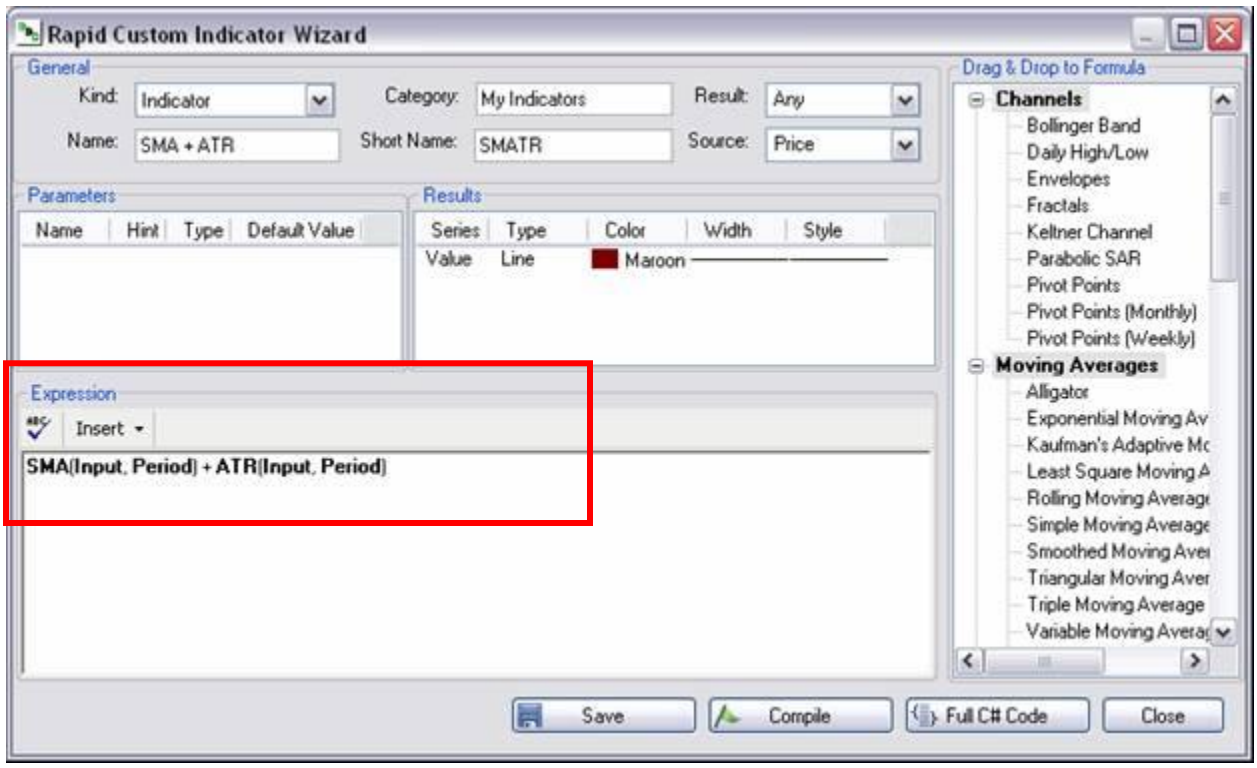
To make a developed Simple Indicator *SMA(Input, 14) + ATR(Input, 14* from the previous example as an ordinary indicator to use anywhere in charts without editing, use the Rapid Custom Indicators function.

To add a new indicator to the library, select *New* in the Custom Indicator Library to display the Rapid Custom Indicator Wizard and complete the data fields:

- **Short Name** is used in expressions and references.
- **Name** is used in list of indicators on the Chart.
- **Category** is used for placing the indicator to a corresponding category in the list of indicators.
- **Result** scale has several options:
    - **Any** places the indicator by default to the same area as input series like Moving Averages.
    - **Free** is used for indicators with free or uncategorized scale like TRIX.
    - **Percent** is used for indicators that calculate a percentage, for example, RSI.
    - **Price** displays that results are prices regardless of input data.
    - **Volume** is used for indicators that return volume values.
- **Source** scale has the same options as the Result scale, but is intended to filter the input series in Property Dialog. Additionally, it has *Stock* option for indicators that require OHLCV as the input data. For example, Average True Range uses High, Low and Close prices, so ATR uses the *Stock* option and the Property Dialog for ATR shows just historical series in the *Input* property.
- To edit the **Result series** the same way as styles of lines in Property Dialog, use a default style of the indicator.
- Copy, type, or construct the expression to **Expression** field.

Refer to the Figure below.

Figure 28 Rapid Custom Indicator Wizard

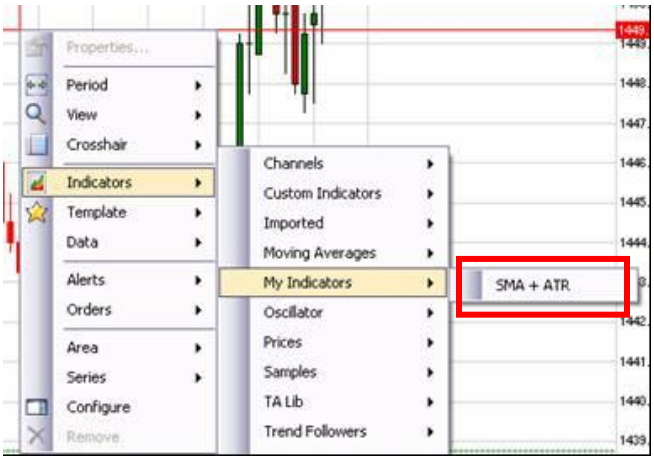


Parameterize the Rapid Custom Indicator

1. Add a parameter *Period* to the list of parameters via the pop-up menu inside this list. Then, replace hardcoded periods of indicators 14 and 15 in the expression to the word *Period*.
2. Parameterize the Rapid Custom Indicator.
3. Now click Save.
4. Press Compile to verify the indicator and add it to Indicators list of the Chart. Refer to the Figure above.

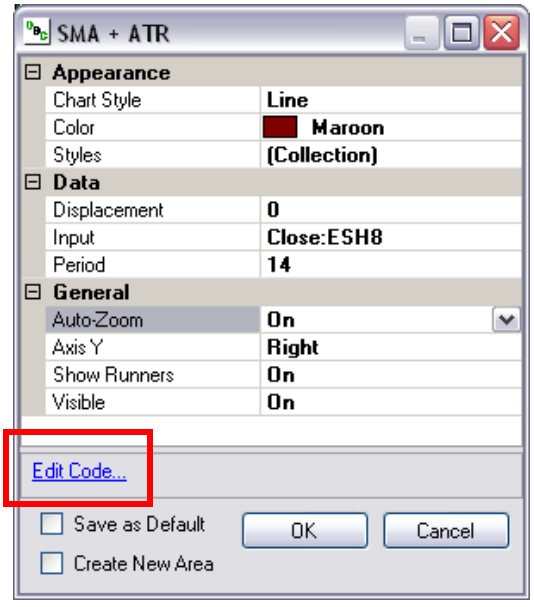
Parameterize from the Chart

Figure 29 Chart



The new indicator displays in Indicators list after successful verification and compilation. Refer to the window above.

Figure 30 SMA+ATR Indicator-Property Dialog Box

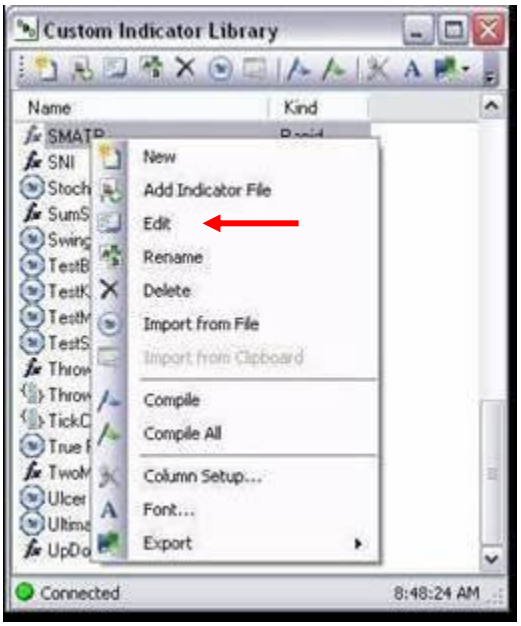


1. Modify the list of parameters, names and default style of lines, formulas and other properties of the Rapid Custom Indicator Wizard any time, via the Custom Indicator Library window.

2. Or, click the Edit Code… hyperlink inside Properties Dialog of the selected indicator. Refer to the Figure to the left.

Multi-Series Rapid Custom Indicators

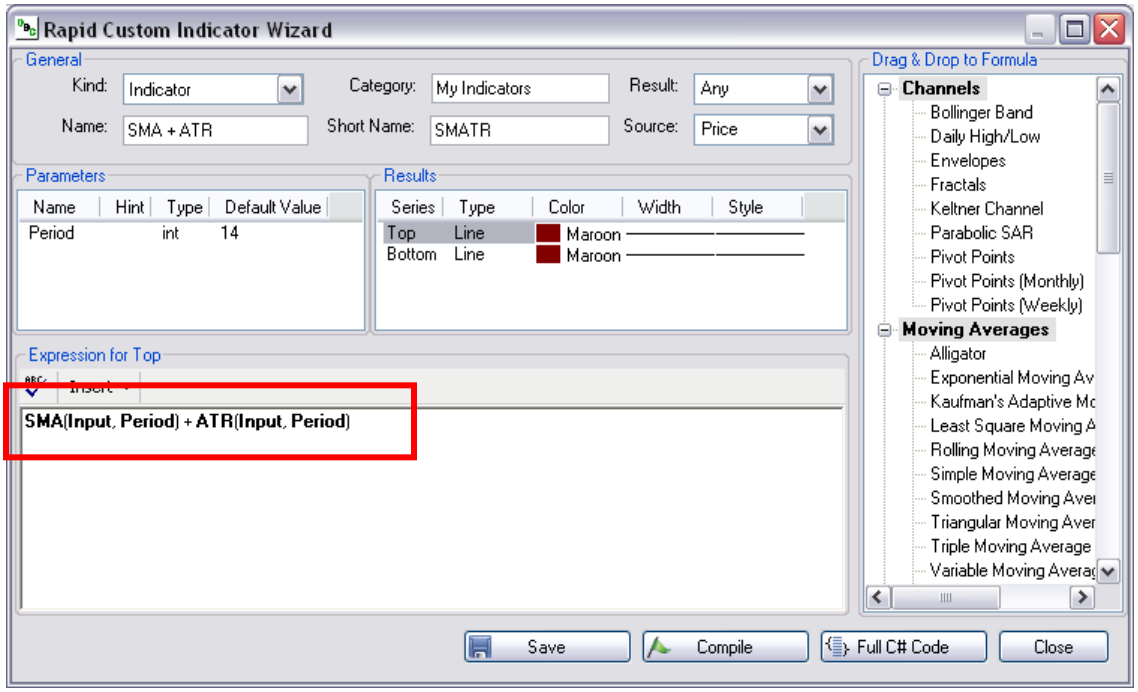
Figure 31 Custom Indicator Library



1. To extend the SMA+ATR indicator from the previous example and show the two series: SMA+ATR and SMA-ATR, select Edit in Custom Indicator Library window for this indicator to open Rapid Custom Indicator Wizard.

2. In the Wizard window, rename series *Value* to *Top* via the pop-up menu of the *Result series* list and add one more series *Bottom*. Refer to the Rapid Custom Indicator Wizard window below.

Figure 32 Rapid Custom Indicator Wizard

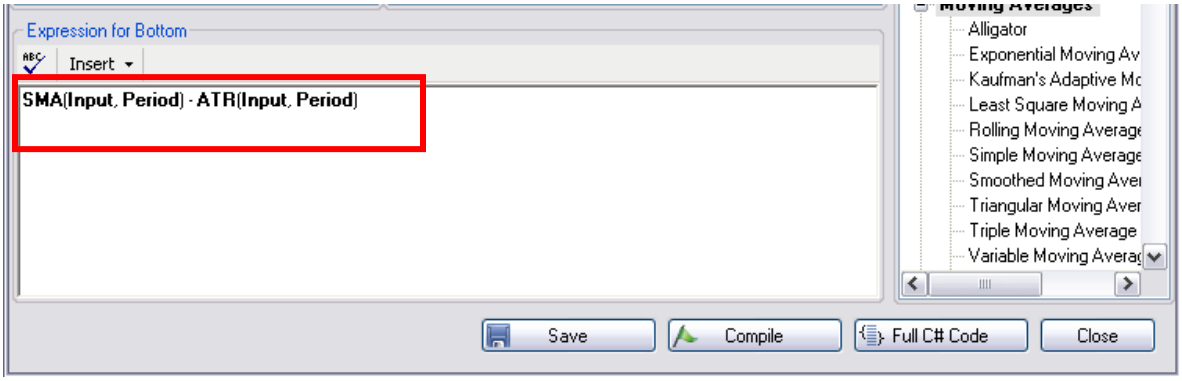


Create a Two Series Expression

1. Now the old expression is a new expression for the series.

2. Select the second series and enter an expression *SMA(Input, Period)-ATR(Input, Period)* in the editor. Refer to the Figure below.

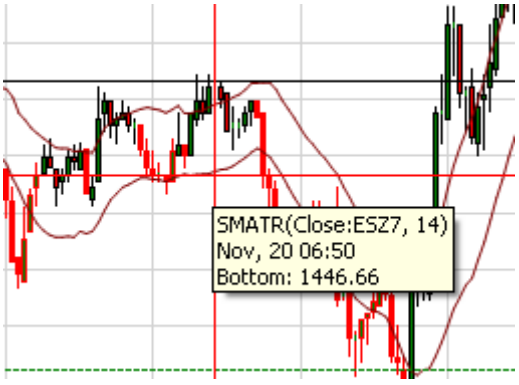
Figure 33 Rapid Custom Indicator Wizard-Expression for Bottom



3. Compile the results to display on the Chart in the *Tooltip* box. A *Tooltip* is a dialog box that displays information at the point of the mouse cursor.

4. Refer to the Figure below.

Figure 34 Chart Results for SMA+ATR Two Series



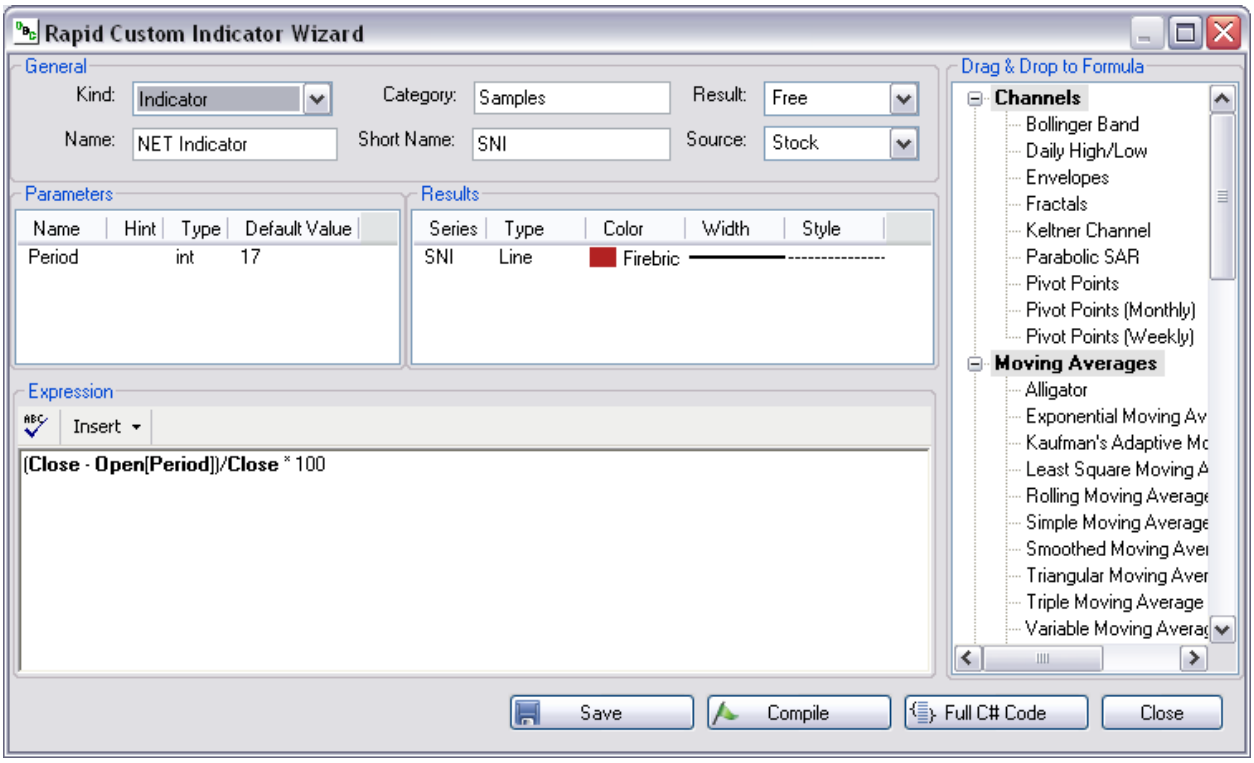
5. Samples of custom indicators contain one or more useful samples of multi-series indicator: TwoMA (Refer to the samples of Rapid indicators.).

OEC .NET Custom Indicator Framework

The OEC Custom Indicator Framework is a .NET library that provides full access to OEC Charts indicator infrastructure: the user writes a single-line simple indicator as well as advanced analyze techniques with complex custom drawing functionality. To support developers, the plug-in contains an integrated development environment with the developer documentation and samples.

A simple way to start development of a new indicator is to use Rapid Custom Indicator Wizard. *Full C# code* button generates the C# file and opens the IDE with it.

Figure 35 Rapid Custom Indicator Wizard



Full C# Code generates the C# class and opens a source code editor.

Study the sample below. All non-bold code is generated automatically on the base of the wizard dialog content.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using OEC.UI;
```

```
using OEC.Chart;
using OEC.Chart.Indicators;
using OEC.Chart.Details;
using OEC.Chart.BaseSeries;
using System.Drawing.Drawing2D;

namespace OEC.Chart.Custom.Temporary
{

      [TimeSeries("NET Indicator", "SNI", "Samples", AreaScale.Free,
SourceScale=AreaScale.Stock)]
      public class SNI : OEC.Chart.Custom.CustomIndicator
      {
            [XmlSerializable, Category("Data")]
            public int Period = 15;


            protected override void Calculate()
            {
                  Result.Value = (Close-Open[Period])/Close*100;
            }


            protected override DataSeries[] DefaultSubseries
            {
                  get
                  {
                        return new DataSeries[]{
                              new DataSeries(this, "SNI", new
ChartStyle(SeriesChartType.Line,System.Drawing.Color.Firebrick,2,DashS
tyle.Dash))

                        };
                  }
            }

            protected Series SNISeries
            {
                  get
                  {
                        return RArray[0];
                  }
            }

      }
}
```

Notes:

- All information is about classes, properties, methods, and fields that are used in custom indicators is found in Context Help.
- The **CustomIndicator** class is a base class of custom indicators.
- The **TimeSeries** attribute describes meta-information and should be defined. Most of the fields of this attribute are reflected in Rapid Wizard dialog.
- Virtual **DefaultSubseries** property should be overriden to declare a list of subseries (lines) and default styles of these subseries.
- The **XmlSerializable** attribute should be used to mark variables and properties as persistent.
- The Virtual **Calculate**() method should be overridden to implement the algorithm of the indicator.
- The **Result** and the **RArray** properties should be used to store the results of calculation.
- The **InputData** refers to the data of the input series that is selected by the user in Input property of Indicator Property dialog.
- The Stock, Open, Close, High, Low, Volume can be used the same way as in the simple expressions (Keywords for input data). All other information from this category can be found in ContractInfo property of indicator.
- Mathematical functions can be found in **MathExt**, **Trig** and the standard **System.Math** classes.
- Functions-indicators can be used the same in the manner as in the simple expressions.

External Assemblies

There is also the ability to add a reference to external assemblies.

To add one or more *//#link NameOfAssembly.dll* directives to first lines of code, use this example:

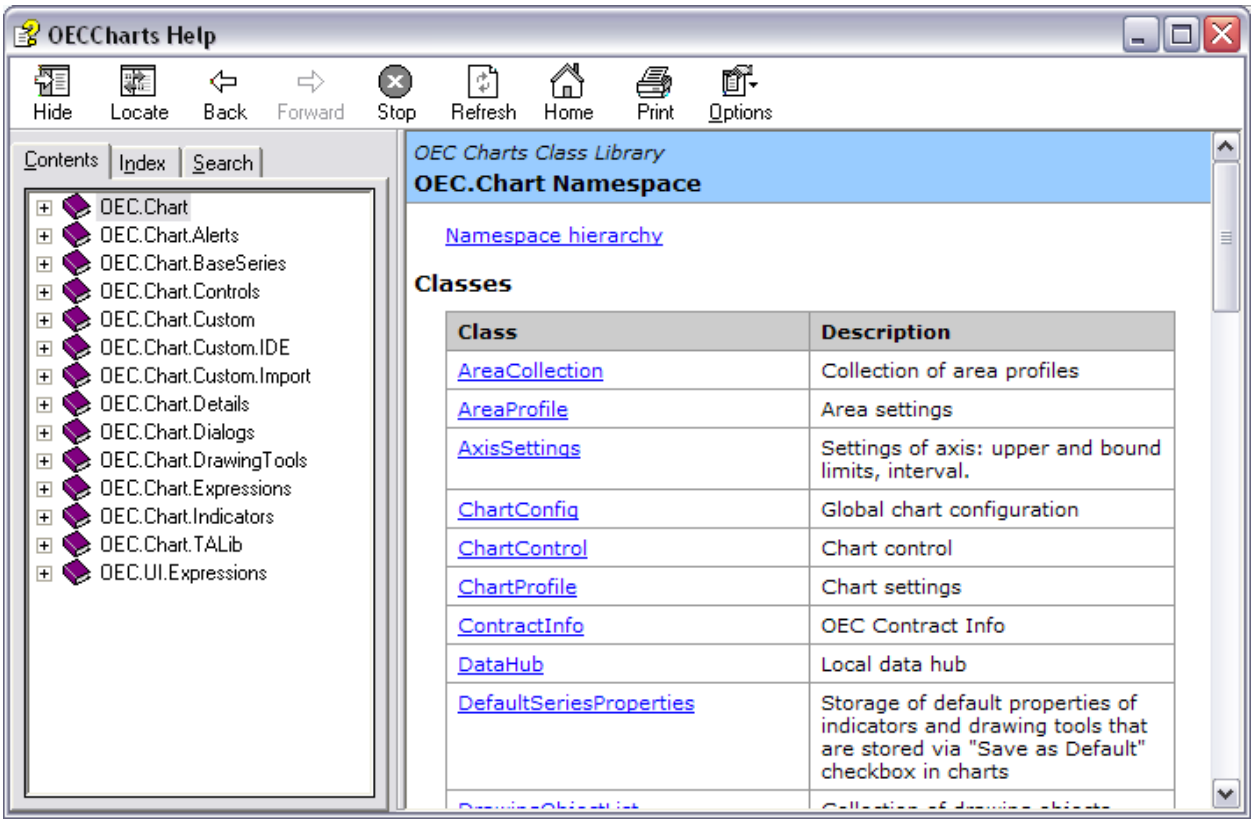
//#link SampleMath.dll

All external libraries should be placed to the OEC\Plug-ins folder.

Context Help

Context help (chm file) for the OEC Charts library displays from Help menu of the code editor.

Figure 36 OECCharts Help

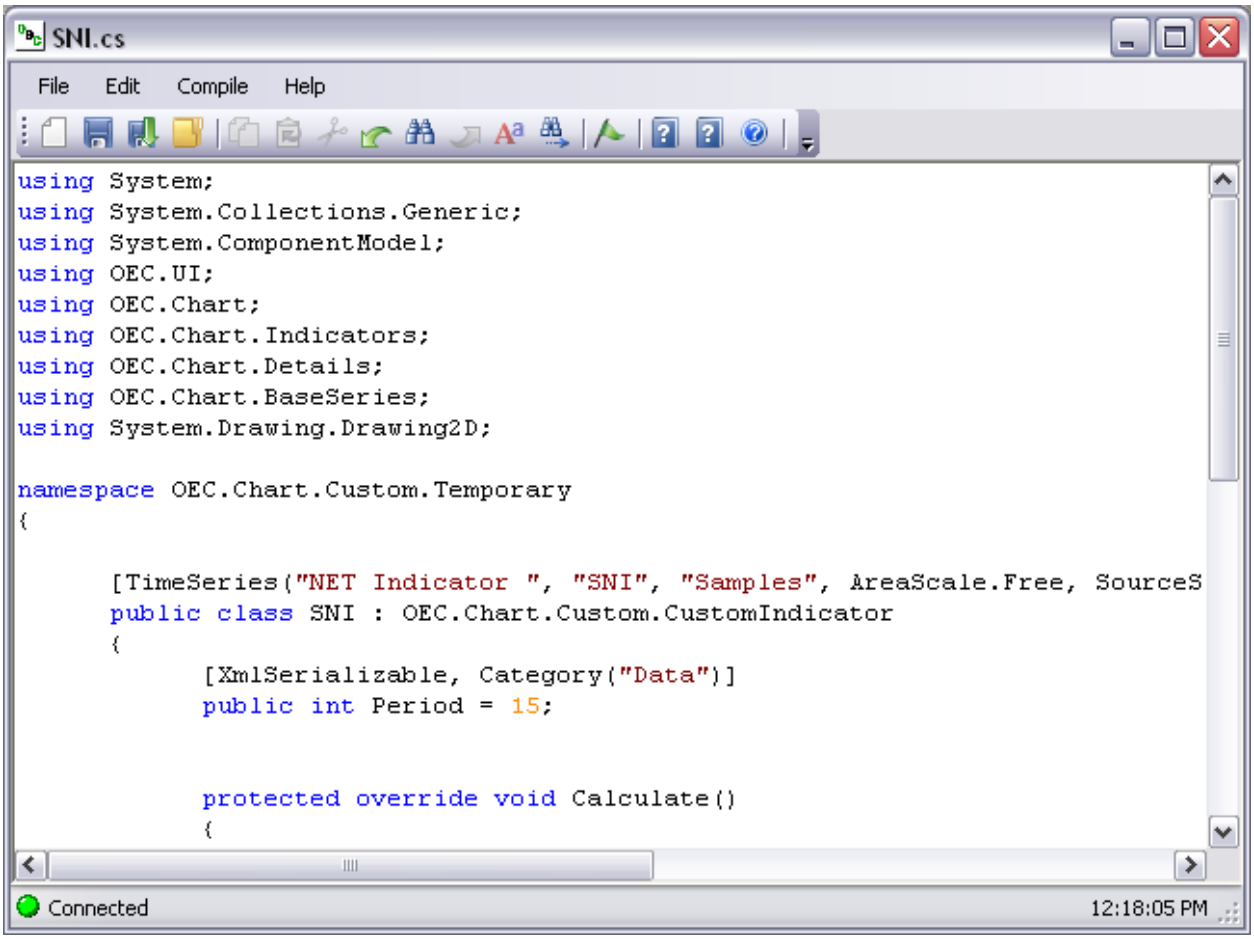


Code Editor

The code editor provides functionality to edit, verify, and compile source code of indicators. The editor displays the next groups of menu commands:

- File menu to save, create, and open files
- Edit menu
- Compile menu
- Help menu

Figure 37 SNI.cs



EasyLanguage ™ Compatibility

The plug-in also uses and develops EasyLanguage analyze techniques.

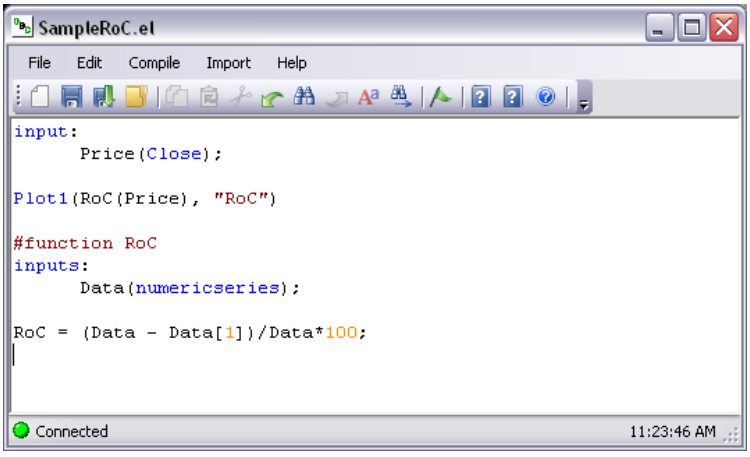
Add EasyLanguage Indicators

There are two ways to create a new EasyLanguage document:

1. The Import File command menu in the Custom Indicator Library Manager offers to select a text file with the EasyLanguage code. Note: The plug-in does not recognize the native file formats of TradeStation because it is a closed format. Use the following steps to complete the procedure.
    a. Open an EasyLanguage file in TradeStation editor.
    b. Open Notepad or Wordpad program and copy to it a content of TradeStation editor.
    c. Save the content of Notepad/Wordpad as a file with EL extension (SomeIndicator.EL, for example)
2. Perform a "copy-paste" from the TradeStation editor to the plug-in editor directly from the Windows clipboard.
3. Or, use Import Clipboard on the command menu to open the EasyLanguage code editor with the current content of the Windows clipboard.
    a. Copy the content to the clipboard somewhere before this operation.

Both of these commands open the EasyLanguage code editor. Refer to the Figure SampleRoC.el.

Figure 38 SampleRoC.el



EasyLanguage Code

The code editor is identical to the C# Code Editor, except for the *Import* menu.

Direct Verify and Compile

Use the **Compile** command in top menu of the code editor to verify the EasyLanguage code directly without translating to C#. A successfully verified and compiled indicator is used immediately in charts. All EasyLanguage indicators are placed to the *Imported* category of the *Indicators* list.

Translating to C#

Because C# provides a wider functionality than EasyLanguage, sometimes it translates the EasyLanguage code to C# to continue development. **Import** commands in the top menu of the code editor are intended for these purposes.

Functions

The plug-in provides the ability to add its own EasyLanguage functions. In contrast to TradeStation, the body of functions is a part of the same file as the body of indicator. To separate the code of the indicator and the code of functions, use the directive #function *name of function*. One indicator file contains the unlimited number of functions.

Example:

```
input:
      Price(Close);

Plot1(RoC(Price), "RoC")

#function RoC
inputs:
      Data(numericseries);

RoC = (Data - Data[1])/Data*100;
```

EasyLanguage code editor also moves functions from another editors.

Copy the code of function from elsewhere and execute the command menu **Add Function**. This command determines the name of the function and inserts it with the **#function** directive to the end of file.

Library of EasyLanguage Functions

The user creates an individual library of EasyLanguage functions.

Save an EasyLanguage file with an extension *.lib.el* (for example, *MyFunctions.lib.el*). All files with this extension are treated as parts of the common library.

If the indicator contains a call of some function, the EasyLanguage compiler searches it in this next sequence:

1. Internally supported functions of the compiler (Refer to List of supported EasyLanguage functions)
2. External functions that declared with “external” directive
3. Functions of the same file as the code of indicator (Refer to Functions)
4. Functions of *.lib.el files

External Libraries

All DLL of external libraries that are referenced via *external* directive of EasyLanguage are located in OEC\s folder.

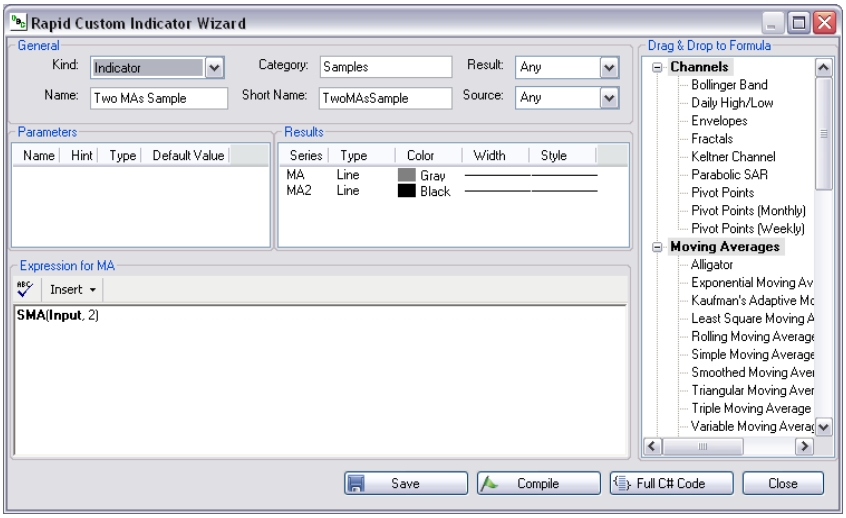
Samples

Rapid Indicators

- Sum: sum all input data
- TwoMA:

This indicator calculates two simple moving averages. There is one is for input data and a second one that is a moving average of the first calculated series. Refer to the Figure *Rapid Custom Indicator Wizard* on the Figure below.

Figure 39 Rapid Custom Indicator Wizard



The MA2 series uses the previously calculated results of the MA series.

C# samples

- For SNI, refer to the OEC .NET Custom Indicator Framework.
- *Sample External Indicator* is a demo of the usage external .NET assemblies. CustomIndicators\Samples\SampleMath folder contains C# sources of the external library.

EasyLanguage Samples

- SampleRoC:is a sample of function usage.
- ExportTest is a demo of *external* EasyLanguage constructions. CustomIndicators\Samples\ExportedDLL contains the C++ sources of external library.

Glossary

Refer to the table of information below for definitions of terminology and related operational concepts Acronyms display within parentheses after the entry.

Term/Acronym	Definition
.NET	A Microsoft operating system platform that incorporates applications, a suite of tools and services and a change in the infrastructure of the company's Web strategy. *
C#	Pronounced "see-sharp." A hybrid of C and C++, it is a Microsoft programming language developed to compete with Sun's Java language. C# is an object-oriented programming language used with XML-based Web services on the .NET platform and designed for improving productivity in the development of Web applications.*
EasyLanguage	**Easy Language** is a specialist programming language that is built in to TradeStation. It is used to create custom indicators for financial charts and also to create algorithmic trading strategies for the markets. External DLL's can be called from within Easy Language which allows the functionality of Easy Language to be extended greatly. **
expression	In programming, an expression is any legal combination of symbols that represents a value. Every expression consists of at least one *operand* and can have one or more *operators*. Operands are values, whereas operators are symbols that represent particular actions. In the expression x + 5 x and 5 are operands, and + is an operator. *
OEC .NET Custom Indicator Framework	OEC Custom Indicator Framework is a .NET library that provides full access to OEC Charts indicator infrastructure: the user writes as a single-line simple indicator as well as advanced analyze techniques with complex custom drawing functionality.
OECTrader	OEC proprietary software for trading futures and options.
Plug-in*	A hardware or software module that adds a specific feature or service

	to a larger system.
	A hardware or software module that adds a specific feature or service to a larger system. The idea is that the new component simply connects to the existing system.*
TA Lib Library	TA Lib library (http://ta-lib.org/) provides common functions for the technical analysis of financial market data: more than 150 technical analysis indicators such as ADX, MACD, RSI, Stochastic, Bollinger Bands, etc. and candlestick pattern recognition.
Tooltip	The OECTrader help dialog box that displays information at the point of the mouse cursor.

*Webopedia.com

**Wikipedia.org

Document History

Software Version	Contributor	Date
OECTrader 3.2.1.8	Information Technology-VV	3/5/08
OECTrader 3.2.1.8	Information Technology-VV	3/24/08

Appendix

Attachment A List of supported EasyLanguage functions

```
AverageFC
SummationFC
ExtremesFC
HighestFC
CSI
DirMovement
MassIndex
RSI
Stochastic
Detrend
MACD
OHLCPeriodsAgo
CloseD
CloseW
XAverage
XAverageOrig
EaseOfMovement
MFI
Range
NormGradientColor
FastKCustomEasy
FastKCustom
TrueRangeCustom
UlcerIndex
MoneyFlow
AccumSwingIndex
SwingIndex
CCI
UltimateOscillator
TrueRange
TrueHigh
TrueLow
LWAccDis
AbsoluteBreadth
ArmsIndex
Next3rdFriday
SortArray
KurtosisArray
LinRegForecastArray
LinRegArray
LinRegInterceptArray
```

```
LinRegSlopeArray
NormDensityArray
NumericRankArray
PercentileArray
PercentRankArray
QuartileArray
SkewArray
StdErrorArray
TrimMeanArray
GradientColor
Alert
AbsoluteBreadth
AbsValue
AccumSwingIndex
Alert
Arctangent
ArmsIndex
Average
AverageArray
AvgDeviation
AvgDeviationArray
CalcDate
CalcTime
CCI
Ceiling
CoefficientR
CoefficientRArray
CoefficientREasy
Correlation
Cosine
CoTangent
CountIf
Covar
CovarArray
CovarEasy
Cum
DateTimeToString
DateToJulian
DateToString
DayFromDateTime
DayOfMonth
DayOfWeek
DayOfWeekFromDateTime
DevSqrd
DevSqrdArray
EaseOfMovement
```

```
ExpValue
Extremes
Factorial
FastKCustom
FastKCustomEasy
Floor
FracPortion
GradientColor
HarmonicMean
HarmonicMeanArray
Highest
HighestArray
HighestBar
HoursFromDateTime
IntPortion
JulianToDate
Kurtosis
KurtosisArray
LinearReg
LinearRegAngle
LinearRegAngleFC
LinearRegFC
LinearRegSlope
LinearRegSlopeFC
LinearRegValue
LinearRegValueFC
LinRegArray
LinRegForecastArray
LinRegInterceptArray
LinRegSlopeArray
Log
Lowest
LowestArray
LowestBar
LWAccDis
MaxList
MaxList2
MedianArray
MFI
MinList
MinList2
MinutesFromDateTime
MinutesToTime
Mod
ModeArray
MoneyFlow
```

```
Month
Neg
Next3rdFriday
NormCumDensity
NormDensity
NormDensityArray
NormGradientColor
NormSCDensity
NumericRankArray
PercentileArray
PercentRankArray
Permutation
Pos
Power
QuartileArray
Random
Range
RGB
Round
RSquare
RSquareArray
SecondsFromDateTime
Sign
Sine
Skew
SkewArray
SortArray
Square
SquareRoot
StandardDev
StandardDevArray
Standardize
StandardizeArray
StdDev
StdDevS
StdError
StdErrorArray
SumList
Summation
SummationArray
SummationIf
SummationRecArray
SummationSqrArray
SwingIndex
Tangent
TimeToMinutes
```

```
TimeToString
TrimMeanArray
TrueHigh
TrueLow
TrueRange
TrueRangeCustom
UlcerIndex
UltimateOscillator
VariancePS
XAverage
XAverageOrig
Year
ZProb
```