User's Guide

NeoTicker 4

Copyrights and Trademarks

Contact Information:

TickQuest Inc.
411 Richmond Street East, Suite 2
Toronto, Ontario
Canada
M5A 3S5

Printed on 5 August, 2007

Email: ***Support*** (mailto:support@tickquest.com)

Website: ***TickQuest*** (http://www.tickquest.com)

Examples used in this document and featured with the software are generated from data by TeleChart 2005 (TC2005). The data is used by permission from Worden Brothers, Inc., developer of TC2005.

Random Functions Copyright and Disclaimer

Random functions in NTLib and formula language are adopted from the algorithm MT19937, with initialization improved 2002/1/26, copyright © 1997-2002 by Makoto Matsumoto and Takuji Nishimura.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Typographical Conventions

Before you start using this guide, it is important to understand the terms and typographical conventions used in the documentation.

For more information on specialized terms used in the documentation, see the Glossary at the end of this document.

The following kinds of formatting in the text identify special information.

Formatting convention	Type of Information
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.
Emphasis	Use to emphasize the importance of a point or for variable expressions such as parameters.
`TEXT`	Strings that you enter, program listings, file names.
CAPITALS	Names of keys on the keyboard. for example, SHIFT, CTRL, or ALT.
KEY-KEY	Key combinations for which the user must press and hold down one key and then press another, for example, CTRL-P, or ALT-F4.

Disclaimer

Past performance is not a guarantee of future results. Only risk capital should be used to trade futures, stocks, options on futures or stocks, mutual funds or any other type of financial instruments. Whether this product is used in conjunction with futures, stocks, stock indices or mutual funds, all involve a high degree of inherent financial risk and the possibility of loss is great. TickQuest Inc. does not assume any responsibilities, make any guarantees whatsoever, or make any trading recommendations in NeoTicker®. All such investment vehicles carry risks and all trading decisions are ultimately made by you. You are solely and individually responsible for those decisions and the results of those decisions.

Contents

New Features Supplement (Changes from 4.0 Official Release to 4.1 Build 24) 47

New Feature Supplement (Version 4.0 Official Release) 1

Formula Topics and Tutorials 255

Operation Guide 395

CHAPTER 1

New Features Supplement (Changes from 4.1 Build 24 to Latest Build)

This chapter lists the new features introduced in each release and guides you to the help pages.

What's New in 4.1 Latest Build

General

- Improved daily data collection for brokerage and ActiveX feed.

Quote Windows

- Support multiple column delete.
- Support INSERT key to insert row.
- Support DELETE key to delete rows or columns.

Order Entry

- When an order entry form is filled with the same symbol as the incoming hot track symbol, it will not refresh the settings as the user could have made changes to the form.
- Account Manager processed order tab can hide cancelled orders. See ***Account Manager, Processed Orders*** (see "Processed Orders" on page 771).

Tick Filter

- Tick Filter has been rewritten. UI has been revised. Engine has been expanded to support bid/ask series. See ***Tick Filter Manager*** (on page 991).

Interactive Brokers

- Improved order status feed back.
- Due to support of IB historical data, IB can now operate by non-protect symbol mode (new default) or protect symbol mode (classic).

QuoteSpeed

- General improvements.

What's New in 4.1 Build 37 to 38

NeoBreadth®

- History Generator speed improvement.
- History Generator dialog is no longer modal.
- History Generator can now work during market hours to generate near real-time historical data.

RAM Cache

- RAM cache initialization speed improvement.

QuoteSpeed

- Historical data loading speed improvement.

What's New in 4.1 Build 36

General

- When rearranging windows, you are prompt to choose a window resizing option. See ***Function Window Arrangement*** (on page 573).

Time Chart

- Ctrl-Z can unzoom chart. See ***Time Chart, Zooming*** (see "Zooming" on page 1041) ***and Time Chart, Keyboard Short***cuts (see "Keyboard Shortcuts" on page 1223).
- Chart data added histogram style.
- Commands to set/clear all channels in drawing tools. See ***Setting up Lines for Multi-line Drawing Tool*** (on page 1141).

Symbol List Manager

- New command to remove duplicate symbols. See ***Symbol List Manager, Removing Duplicate Symbols*** (see "Removing Duplicate Symbols" on page 933).

System Performance Viewer

- In Postion Query, supports range operator across all fields. See ***System Performance Viewer, Positions / Position Query*** (see "Positions / Position Query" on page 969).
- Position List and Position Query added three new columns - net points, MFE, MAE.

Grid Computing

- Can configure NeoTicker® to maximize CPU usage. See ***Grid Computing*** (on page 507).

What's New in 4.1 Build 33 to 35

Formula

- New random stream functions. See ***Formula, Using Random Stream*** (see "Using Random Stream" on page 288).

Programming

- New random stream functions. See ***NTLib Object, Random Functions*** (see "Random Functions" on page 1617).
- Trading system statistics is now available through the Trade.SystemStats object. ***See Trade Object, SystemStats Ob***ject (see "SystemStats Object" on page 1607).

What's New in 4.1 Build 32

Time Chart

- Andrew Pitchfork supports line detection on all multiples and extended lines.

OLE Automation

- Complete set of trading system statistics is available. See ***TradingSystemStats Object*** (on page 677).

What's New in 4.1 Build 31

General

- When using text files (data source and various text import functions), thousand separator (e.g. comma in 1,000) is supported.

Trade Simulator

- Trade Simulator can import text file for analysis. See ***Trade Simulator, Importing Trades*** (see "Importing Trades" on page 765).

What's New in 4.1 Build 30

User Define Symbol Manager

- Weighted index support symbol list file. See ***User Define Symbol Manager, Settings*** (see "Settings" on page 1252).

OLE Automation

- Report object methods - Ready, OpenNew. See ***Report Object, Properties and Methods*** (see "Properties and Methods" on page 1562).

Indicators

- Optdemo and Optdemosys - they are part of the up coming optimizer's tutorial example. See ***Optimization Demo (optdemo)*** (on page 1991) and ***Optimization Demo System (optdemosys)*** (on page 1993).

What's New in 4.1 Build 26

These are bug fix builds.

What's New in 4.1 Build 24

Note: We've changed the version nomenclature from beta's to build numbers. This allows us to quickly release versions using our new automated release system.

General

- Auto Complete Symbol Entry for automatically complete symbol typing in various symbol edit box. See ***Auto Symbol Completion*** (on page 567).

Time Chart

- Data alignment options in Chart Manager to controls how the data aligns with chart when there is not enough bars to fill the chart. See ***Time Chart Configuration>Data Alignment*** (see "Data Alignment" on page 1073).

Cache Manager

- New advance RAM Cache option to control data integrity checking. See ***Cache MAnager - RAM Cache > Data Integrity Checking*** (see "Data Integrity Checking" on page 466).
- Cache Data Editor. See ***Cache Manager - Disk Cache > Cache Data Editor*** (see "Cache Data Editor" on page 446).
- Supports data file locking/unlocking. See ***Cache Manager - Disk Cache > File Locking*** (see "File Locking" on page 452).

Indicators

- BarsagoBytime to returns number of bars ago using time parameters. See ***Indicator Reference > Barsago By Time (barsagobytime)*** (see "Barsago By Time (barsagobytime)" on page 1750).

Order Interface

- Stealth mode limit and stop order. Unlike traditional broker handled limit and stop orders, stealth order logic is handled by NeoTicker and are not visible to other traders until time of execution. See ***Stealth Stop and Limit Orders*** (on page 750).

Programming

- New Managed Series for creating totally customizable series within an indicator. Support methods are `SetManagedSeries` and `ManagedSeries`. See ***ItSelf Object, Properties and Methods*** (see "Properties and Methods" on page 1525).

What's New from 4.1 Beta 9 to 4.1 Beta 10

We recommend Beta 10 for users who:

- Use Sim Server (redesigned user interface)
- Use RealTick, IQFeed, DTNIQ real-time feeds (significantly better data handling)
- Require Quotes Plus QP3 support (fixed serious bugs)

- Require Excel integration (new Excel RTD support)
- Prefer candlestick charts (candlestick drawing has been rewritten)
- Print a lot of charts to a printer.
- Want to write indicators in a .NET language.

Beta 10 contains significant code changes.

General

- New User Preference option to control sound replay style. See ***Configuration Guide, Playing Sound*** (see "Playing Sound" on page 508).
- Use can use Ctrl-Space to pop up main menu in any monitor. See ***Main Window Operation Guide, Shortcuts*** (see "Shortcuts" on page 628). Note: this feature is incomplete. If you open a window from the menu, it may not opened in the menu's monitor.

Time Chart

- Candlestick is now default to draw using sharp edges. An user preference option lets you revert to old style candlestick. See ***Visual Style for Candlestick, Bar and Box*** (on page 1086).
- When creating a Text drawing tool, the editor is open right away for entering text. ***See Time Chart, Text*** Tool (see "Text Tool" on page 1163).
- Smart mode printing has been rewritten. See ***Printing*** (on page 585).
- Button to move Data View window right in front of chart under Chart Manager>Misc tab. This is for fixing incorrect Data View window positioning problems. See ***Data View Window*** (on page 1030).

Pattern Scanner

- In setup window, under Indicator tab, a new copy button has been introduced. See ***Adding, Editing, Copying Indicator*** (on page 831).

Dynamic Table / Dynamic Grid

- Can drag and drop tab sheets to reorder the sheets. See ***Dynamic Table Set up*** (on page 565).
- Can drag and drop tab sheets to move sheets across Dynamic Tables. See ***Dynamic Table Set up*** (on page 565).
- Dynamic Grid support grid level number formatting. See ***Dynamic Grid Operation Guide, Grid Level Setup*** (see "Grid Level Setup" on page 547).

Simulation Server

- New re-designed user interface. See ***Setting up NeoTicker® as Simulator*** (on page 9) and ***Simulation Server Operation Guide*** (on page 899).

OLE Automation

- New FWTimeChart function GetChartDataRange to return the combined start and end time of all data contained in chart. See ***OLE Automation, FWTimeChart Object*** (see "FWTimeChart Object" on page 672).
- New FWTimeChart method TickReplay to drive tick replay. See ***OLE Automation, FWTimeChart Object*** (see "FWTimeChart Object" on page 672).
- New App method TakeScreenshot. See ***OLE Automation, App Object - Misc*** (see "App Object - Misc" on page 662).
- New FunctionWindow method Activate. See ***OLE Automation, FunctionWindow Object*** (see "FunctionWindow Object" on page 664).

Excel/DDE

- DDE link can now use all the fields available in formula. This may break compatibility with old DDE field names. DDE example has been revised. See ***RTD / DDE Link Manager, How It Works and Topic Reference*** (see "How It Works and Topic Reference" on page 891).
- New Excel RTD support. DDE Link Manager is renamed to RTD / DDE Link Manager as it now supports both standards. See ***RTD / DDE Link Manager*** (on page 889).

IQFeed

- Support IQFeed 4.1.1.1. This version of IQFeed is not compatible with previous version. If you use Beta 10, you will need to upgrade IQFeed client to 4.1.1.1. See ***Configuring IQFeed*** (on page 27).

Formula

- New functions haslasttrade, lasttradeprice, lasttradesize. See ***Accessing Data Links and Fields*** (on page 278).
- Real-time bid/ask/trade fields are available to indicator calculated on bar completion. See ***Accessing Data Links and Fields*** (on page 278).

Trading Systems

- Block order intelligently when single entry per direction is enabled and a live open market order in the same direction is already in place. See ***Order Defaults and Completion Related Topics, Single Entry Per Direction*** (see "Single Entry Per Direction" on page 749).

Programming

- Indicator Setup window has been re-organized. A new IDL tab has been introduced for new options: early binding for VB IDL indicators and .NET indicators. See ***Indicator Specification*** (on page 1627).
- Early binding provides significant performance improvements for IDL written in VB. ***See IDL Over***view (on page 1635).
- New Params object function Exists to determine if a parameter exists (enabled by script writer). See ***Properties and Methods for Params Object*** (on page 1524).
- New Excel object method OpenWorkbook, CloseWorkbook. See ***Excel Object*** (on page 1563).
- Indicators that are calculated on bar completion now supports the following real-time fields in data object: lastprice, lastsize, bidprice, bidsize, askprice, asksize. See ***Properties and Methods for Real-Time Bid/Ask/Last Trade Data*** (on page 1542).
- Support .NET languages. See ***IDL Interface (DLL, ActiveX and .NET indicators)*** (on page 1635).
- New function NTLib.ApplicationDirectory. See ***NTLib Object, File Functions*** (see "File Functions" on page 1622).
- New function NTLib.IndicatorDirectory. See ***NTLib Object, File Functions*** (see "File Functions" on page 1622).
- New version of nested indicator calls: Itself.MakeIndicatorEx, ItSelf.UpdateIndicatorEx, Itself.IndicatorEx, Itself.CompressSeriesEx, Itself.UpdateCompressSeriesEx. The new calls offer significant performance improvements for IDL indicators. See ***Using Other Indicators*** (on page 1479), ***Creating Higher Time Frame Series*** (on page 1452) and ***ItSelf Object, Properties and Methods*** (see "Properties and Methods" on page 1525).
- The entry function `idlcallex` is the new standard for entering into an IDL indicator. See ***IDL Interface (DLL, ActiveX and .NET indicators)*** (on page 1635)

CHAPTER 2

New Features Supplement (Changes from 4.0 Official Release to 4.1 Build 24)

What's New from 4.1 Beta 9 to 4.1 Beta 10

We recommend Beta 10 for users who:

- Use Sim Server (redesigned user interface)
- Use RealTick, IQFeed, DTNIQ real-time feeds (significantly better data handling)
- Require Quotes Plus QP3 support (fixed serious bugs)
- Require Excel integration (new Excel RTD support)
- Prefer candlestick charts (candlestick drawing has been rewritten)
- Print a lot of charts to a printer.
- Want to write indicators in a .NET language.

Beta 10 contains significant code changes.

General

- New User Preference option to control sound replay style. See ***Configuration Guide, Playing Sound*** (see "Playing Sound" on page 508).
- Use can use Ctrl-Space to pop up main menu in any monitor. See ***Main Window Operation Guide, Shortcuts*** (see "Shortcuts" on page 628). Note: this feature is incomplete. If you open a window from the menu, it may not opened in the menu's monitor.

Time Chart

- Candlestick is now default to draw using sharp edges. An user preference option lets you revert to old style candlestick. See ***Visual Style for Candlestick, Bar and Box*** (on page 1086).
- When creating a Text drawing tool, the editor is open right away for entering text. ***See Time Chart, Text*** Tool (see "Text Tool" on page 1163).
- Smart mode printing has been rewritten. See ***Printing*** (on page 585).
- Button to move Data View window right in front of chart under Chart Manager>Misc tab. This is for fixing incorrect Data View window positioning problems. See ***Data View Window*** (on page 1030).

Pattern Scanner

- In setup window, under Indicator tab, a new copy button has been introduced. See ***Adding, Editing, Copying Indicator*** (on page 831).

Dynamic Table / Dynamic Grid

- Can drag and drop tab sheets to reorder the sheets. See ***Dynamic Table Set up*** (on page 565).
- Can drag and drop tab sheets to move sheets across Dynamic Tables. See ***Dynamic Table Set up*** (on page 565).
- Dynamic Grid support grid level number formatting. See ***Dynamic Grid Operation Guide, Grid Level Setup*** (see "Grid Level Setup" on page 547).

Simulation Server

- New re-designed user interface. See ***Setting up NeoTicker® as Simulator*** (on page 9) and ***Simulation Server Operation Guide*** (on page 899).

OLE Automation

- New FWTimeChart function GetChartDataRange to return the combined start and end time of all data contained in chart. See ***OLE Automation, FWTimeChart Object*** (see "FWTimeChart Object" on page 672).
- New FWTimeChart method TickReplay to drive tick replay. See ***OLE Automation, FWTimeChart Object*** (see "FWTimeChart Object" on page 672).
- New App method TakeScreenshot. See ***OLE Automation, App Object - Misc*** (see "App Object - Misc" on page 662).
- New FunctionWindow method Activate. See ***OLE Automation, FunctionWindow Object*** (see "FunctionWindow Object" on page 664).

Excel/DDE

- DDE link can now use all the fields available in formula. This may break compatibility with old DDE field names. DDE example has been revised. See ***RTD / DDE Link Manager, How It Works and Topic Reference*** (see "How It Works and Topic Reference" on page 891).
- New Excel RTD support. DDE Link Manager is renamed to RTD / DDE Link Manager as it now supports both standards. See ***RTD / DDE Link Manager*** (on page 889).

IQFeed

- Support IQFeed 4.1.1.1. This version of IQFeed is not compatible with previous version. If you use Beta 10, you will need to upgrade IQFeed client to 4.1.1.1. See ***Configuring IQFeed*** (on page 27).

Formula

- New functions haslasttrade, lasttradeprice, lasttradesize. See ***Accessing Data Links and Fields*** (on page 278).
- Real-time bid/ask/trade fields are available to indicator calculated on bar completion. See ***Accessing Data Links and Fields*** (on page 278).

Trading Systems

- Block order intelligently when single entry per direction is enabled and a live open market order in the same direction is already in place. See ***Order Defaults and Completion Related Topics, Single Entry Per Direction*** (see "Single Entry Per Direction" on page 749).

Programming

- Indicator Setup window has been re-organized. A new IDL tab has been introduced for new options: early binding for VB IDL indicators and .NET indicators. See ***Indicator Specification*** (on page 1627).
- Early binding provides significant performance improvements for IDL written in VB. ***See IDL Over***view (on page 1635).
- New Params object function Exists to determine if a parameter exists (enabled by script writer). See ***Properties and Methods for Params Object*** (on page 1524).
- New Excel object method OpenWorkbook, CloseWorkbook. See ***Excel Object*** (on page 1563).
- Indicators that are calculated on bar completion now supports the following real-time fields in data object: lastprice, lastsize, bidprice, bidsize, askprice, asksize. See ***Properties and Methods for Real-Time Bid/Ask/Last Trade Data*** (on page 1542).
- Support .NET languages. See ***IDL Interface (DLL, ActiveX and .NET indicators)*** (on page 1635).
- New function NTLib.ApplicationDirectory. See ***NTLib Object, File Functions*** (see "File Functions" on page 1622).
- New function NTLib.IndicatorDirectory. See ***NTLib Object, File Functions*** (see "File Functions" on page 1622).
- New version of nested indicator calls: Itself.MakeIndicatorEx, ItSelf.UpdateIndicatorEx, Itself.IndicatorEx, Itself.CompressSeriesEx, Itself.UpdateCompressSeriesEx. The new calls offer significant performance improvements for IDL indicators. See ***Using Other Indicators*** (on page 1479), ***Creating Higher Time Frame Series*** (on page 1452) and ***ItSelf Object, Properties and Methods*** (see "Properties and Methods" on page 1525).

- The entry function `idlcallex` is the new standard for entering into an IDL indicator. See ***IDL Interface (DLL, ActiveX and .NET indicators)*** (on page 1635)

What's New from 4.1 Beta 8 .o 4.1 Beta 9

Although we recommend users to give Beta 9 a try, this version has more risky changes than previous betas.

General

- Controls for main window tool bar have been revised. See ***Customizable Main Window Tool Bar*** (on page 498).

Indicators

- New indicator Parabolic SAR EX, which supports both Factor and Max settings. See ***Parabolic SAR EX (parabolicex)*** (on page 2001).

Formula

- In formula indicators, a new data field "symbol" is added. The field is intended to be used in reporting. See ***Formula, Accessing Data Links*** (see "Accessing Data Links and Fields" on page 278).

OLE Automation

- New method calls for FWTimeChart: ScrollByBar, ScrollByPage, ChartAtRightEdge, GetChartVisibleRange. See ***OLE Automation, FWTimeChart Object*** (see "FWTimeChart Object" on page 672).

What's New from 4.1 Beta 7 to 4.1 Beta 8

Beta 8 is an emergency bug fix release that address the problem of broken quote window formula.

What's New from 4.1 Beta 6 to 4.1 Beta 7

4.1 is a matured beta release. We recommend all users who plan to upgrade to 4.1 to give 4.1 Beta 7 a try.

General

- In NeoTicker® icon menu in Windows task bar, an item is added to restore main window to primary monitor. See ***Finding a Lost Window*** (on page 572).

Trading Systems

- New fill type – Bid Limited Worse and Ask Limited Worst for Forex testing. See ***How Orders are Filled*** (on page 1197).

Simulation Server

- You can specify Forex symbols and Simulation Server can generate Forex ticks based on bid/ask/midpoint. See ***Forex Tick Simulation*** (on page 908).

User Defined Symbols and NeoBreadth®

- A delay field is added to User Defined Symbol and NeoBreadth® which allows construction of symbols using delay data. See ***NeoBreadth, Reference*** (see "Reference" on page 642) and ***User Define Symbol Manager, Settings*** (see "Settings" on page 1252).

Formula

- Functions to read Gheap – gHeapValue, gHeapReal, gHeapInt, gHeapRealList, gHeapIntList, gHeapRealTable, gHeapIntTable. Gheap is used by varies parts of NeoTicker® to share data. See ***Generic Functions - GHeap Related*** (on page 301).

Programming

- New fill type ftBidLimitedWorst and ftAskLimitedWorst for Forex testing. See ***Trade Object, System Setting Properties*** (see "System Setting Properties" on page 1604).
- New NTLib function FileExists, DirectoryExists, DirectoryFilelist. See ***File Functions*** (on page 1622).

What's New from 4.1 Beta 5 to 4.1 Beta 6

4.1 is a matured beta release. We recommend all users who plan to upgrade to 4.1 to give 4.1 Beta 6 a try.

Alerts

- Alert Editor allowing specifying the location of the alert pop up window.

What's New from 4.1 Beta 4 to 4.1 Beta 5

4.1 is a matured beta release. We recommend all users who plan to upgrade to 4.1 to give 4.1 Beta 5 a try.

General

- Application wide decimal places preference in User Preference. See ***Decimal Places*** (on page 499).
- eSignal, DTN IQFeed, MB Trading supports better symbol limit warning and protection against symbol lock out by data service. System Log added error alert option in error tab. See ***Symbol Limit*** (on page 524).

TickQuest Backfill

- Our own backfill service with tick data support for users who use data services that do not provide historical data. See ***Setting Up Backfill*** (on page 541).

Continuous Contract Manager

- Continuous contract support, data vendor independent. See ***Continuous Contract Manager*** (on page 511).

Dynamic Grid

- Dynamic Grid pop up menu revised to compact the menu on top level. - cell setup is now a sub menu; all grid visual settings grouped under grid visual; new commands – copy cell to temp file, copy temp file to various destinations, cell setup load most recently used. See ***Dynamic Grid Operation Guide*** (on page 545)
- Dynamic Grid – double click to edit symbol. See ***Dynamic Grid Operation Guide, Double Click Action*** (see "Double Click Action" on page 563).
- Forex and Forex + Value display styles. See ***Dynamic Grid Operation Guide, Cell Level Setup*** (see "Cell Level Setup" on page 553).

Time Chart

- Chart data series can be drag and drop into other function windows. See ***Time Chart Operation Guide, Data Series and Indicators, Drag and Drop*** (see "Drag and Drop" on page 1125).

Account Manager

- Open Orders tab – Mark All Orders Cancelled. See ***Account Manager, Open Orders*** (see "Open Orders" on page 769).
- Positions tab – Remote Flat button. See ***Account Manager, Positions*** (see "Positions" on page 772).
- Live Systems tab – added ability to resolve position mismatch using right click pop up menu. See ***Account Manager, Live Systems*** (see "Live Systems" on page 776).
- Open Orders – pop up menu. See ***Account Manager, Open Orders*** (see "Open Orders" on page 769).
- Processed Order page. See ***Account Manager, Processed Orders*** (see "Processed Orders" on page 771).

Indicator Manager

- Script Manager has been renamed Indicator Manager to properly reflect its intended usage. See ***Indicator Manager*** (on page 1462).

MB Trading

- MB Order Server now recognizes Forex symbols, automatically provide price multiples and minimum tick size. See ***Configuring MB Trading as Real-Time Data Feed*** (on page 33).

Interactive Brokers

- New IB server with data streaming and level 2 ready. See ***Configuring Interactive Brokers as Real-Time Data Feed*** (on page 23).

Formula

- New time in force order placement functions, supplementary functions and constants. See ***Formula Order Placement Methods (Time In Force)*** (on page 323).

Programming

- Trade Object – new property OrderPlacementUsingTradeSimOnly. See ***Trade Object, System Setting Properties*** (see "System Setting Properties" on page 1604).

What's New from 4.1 Beta 3 to 4.1 Beta 4

Symbol Info Manager

- Added Forex data type. See ***Symbol Info Manager, Basic Tab and Related Fields*** (see "Basic Tab and Related Fields" on page 923).

Indicator Manager (aka Script Manager)

- Added Filename column. See ***Indicator Manager*** (on page 1462).

Account Manager

- New quick order entry area. See ***Account Manager Order Placement*** (on page 773).
- New Live Systems report. See ***Account Manager, Live Systems*** (see "Live Systems" on page 776).
- Status bar acts as quick jump button to switch among the tab sheets. See ***Introduction to Account Manager*** (on page 767).
- Position tab added quick buttons to flat an open position or go in reverse direction ***Account Manager, Positions*** (see "Positions" on page 772).

Programming / Indicator Objects

- New Trade object order placement methods that address very high frequency trading. All these new methods have time in force equals otfDay. For more details, refer to the original versions which have time in force equals to otfFillorKill. The methods are BuyAtMarketDay, SellAtMarketDay, LongAtMarketDay, LongExitAtMarketDay, ShortAtMarketDay, ShortExitAtMarketDay, ExitCurrentPositionDay, BuyAtMarketDayEx, SellAtMarketDayEx, LongAtMarketDayEx, LongExitAtMarketDayEx, ShortAtMarketDayEx, ShortExitAtMarketDayEx, ExitCurrentPositionDayEx, ExitAllCurrentPositionsDay. See ***Order Placement - Market Day Order Methods*** (on page 1581).

What's New from 4.1 Beta 2 to 4.1 Beta 3

General

- New Order Placement popup menu system introduced, allowing order placement directly from all function windows. To activate the service, change the Order Interface Setup, Shortcut options for F2/F3 to Function Window Popup Menu. See ***Placing Order with Shortcut Keys*** (on page 252) for a tutorial and ***Order Entry Shortcut Keys Setup*** (on page 739) for setup options.

Formula

- Quote formulas added the following context functions, MinTickSize, MinTickSize (Symbol), DefaultOrderSize, DefaultOrderSize (Symbol), ScaleOrderSize, ScaleOrderSize (Symbol). See ***Context-Driven Functions for Quote Window Formula Columns*** (on page 306).

- Indicator formulas added the following context function, ScaleOrderSize. See ***Context-Driven Functions for Time Chart, Pattern Scanners and Inside Formula Indicators*** (on page 308).

Programming

- Trade object added the following properties, ScaleOrderSize, ScaleOrderSizes [Index]. See ***Trade Object, System Setting Properties*** (see "System Setting Properties" on page 1604) and ***Trade Object, Portfolio System Setting Properties*** (see "Portfolio System Setting Properties" on page 1601).

Account Manager

- Confirmation page added the following quick buttons, Send All, Cancel All, and Confirmation Setup. See ***Account Manager, Confirming Orders*** (see "Confirming Orders" on page 768).
- Open Order page added quick button Cancel All. See ***Account Manager, Open Orders*** (see "Open Orders" on page 769).
- Position page added quick button Flat All. See ***Account Manager, Positions*** (see "Positions" on page 772).

Order Interface

- F2/F3 shortcuts supporting new actions - function window popup menu, and route order information to designated order form. See ***Placing Order with Shortcut Keys*** (on page 252), ***Order Entry Shortcut Keys*** (on page 732) and ***Order Entry Shortcut Keys Setup*** (on page 739).
- Security type defaults added support of Scale Size. See ***Order Interface, Trade Size and Scale Size Setup*** (see "Trade Size and Scale Size Setup" on page 752).

Symbol Info Manager

- Added new field Scale Size. See ***Symbol Info Manager, Basic Tab and Related Fields*** (see "Basic Tab and Related Fields" on page 923).

Quote Window

- Column Properties added new style Order Placement that enables right-click on a column to bring up the order placement popup menu immediately. See ***Placing Order with Shortcut Keys*** (on page 252).

What's New from 4.1 Beta 1 to 4.1 Beta 2

4.1 is currently a beta release, we recommend upgrading only if you use the following features:

- RealTick
- QuoteSpeed
- IQFeed
- Instant Data Server
- Long term trading system testing
- Drawing tools (heavy usage only)
- Currently using 4.1 Beta 1

General

- When setting up a real-time server, additional checks and visible password mode to reduce possibility of mistyped password. See ***Server Setup Reference, Login Tab*** (see "Login Tab" on page 520).

RealTick

- RealTick is supported as a data feed. See ***Configuring RealTick*** (on page 31).

Trading Systems

- Under indicator editor, System tab, new option to adjust resolution of equity. Equity now has a default resolution of day to reduce memory usage. See ***Trading System Settings*** (on page 1193).
- System Performance Viewer, equity graph hourly compression. See ***System Performance Viewer, Summary / Equity Graph*** (see "Summary / Equity Graph" on page 961).
- System Performance Viewer, additional lines in Equity Graph – cash level, margin used, regression. See ***System Performance Viewer, Summary / Equity Graph*** (see "Summary / Equity Graph" on page 961).
- System Performance Viewer, options to turn off Equity Graph lines. See ***System Performance Viewer, Summary / Equity Graph*** (see "Summary / Equity Graph" on page 961).

Programming

- New Heap object methods: PriceProfileAddPriceRangeFixedVolume, PriceProfileRemovePriceRAngeFixedVolume. Volume is not divided by the slots like the non-fixed volume version of the calls. See ***Heap, PHeap and GHeap, Properties and Methods for Price Profiling*** (see "Properties and Methods for Price Profiling" on page 1556).

- New Trade object methods: CancelAllOrdersByType, CancelAllOrdersByTypeEx. ***See Order Placement - Special Order Handling Meth**ods* (on page 1588) and ***Portfolio Special Order Handling Methods*** (on page 1600).
- Script Editor supports line number. See ***Edit Area (Pane 1, Pane 2)*** (on page 1625).

What's New from 4.01 to 4.1 Beta 1

4.1 is currently a beta release, we recommend upgrading only if you use the following features:

- IQFeed.
- Instant Data Server.
- Automatic order placement to a brokerage such as MB Trading or IB using a trading system.
- The new QuoteSpeed data feed support.
- Need to import tick data with bid/ask into Disk Cache.
- Use a tick replay workflow in time charts.

General

- In Server Setup window, Misc tab. New defaults for reconnection option is introduced. See ***Server Setup Reference, Misc Tab*** (see "Misc Tab" on page 521).

Time Chart

- New item added to data series pop up menu – Set Object Ordering. See ***Re-arranging the Display Order of Data Series and Indicators*** (on page 1120).
- In tool bar, right clicking on short cut buttons will open pop up menu to drive Shortcut Manager. See ***Shortcut Manager Operation Guide*** (on page 911).
- In Chart Manager, Data tab, the Show All and Hide All buttons have been consolidated to the All button with pop up menu. See ***Chart Manager*** (on page 1013).
- In Chart Manager, Data tab, support delete all operation. See ***Chart Manager*** (on page 1013).

Formula

- New functions BrokerPosSize and BrokerPosAvgEntry to retrieve position size and average entry price from your brokerage. The symbol is always that of the primary data series. See ***Broker Functions*** (on page 329).
- New functions to return trading system related information about the primary data series. See ***Misc Formula Trading System Functions*** (on page 327).

- New functions to return trading time related information. See ***Context-Driven Functions (Time Frame Related)*** (on page 310).
- New fields to query tick level statistics on a data link. See ***Accessing Data Links*** (see "Accessing Data Links and Fields" on page 278).

Indicators

- Indicator directory is now user configurable. See ***Indicator Directory*** (on page 507).
- New indicator `Countdown Time`. Apply this indicator to a chart, the indicator will display the time remaining to complete the latest bar. See ***Countdown Time (countdown_time)*** (on page 1805).

Order Placement

- In Account Manager, background color is introduced to improve readability of orders. A Visual tab has been introduced to set options. See ***Account Manager, Visual Options*** (see "Visual Options" on page 780).

Cache Manager

- When using Sim Server, Cache Manager is in read only mode, i.e. cannot modify cache data. This is now clearly indicated in Cache Manager. See ***Cache Manager - Disk Cache, Read-Only Mode (Simulation Server)*** (see "Read-Only Mode (Simulation Server)" on page 454).
- When importing tick data, you can now optionally construct minute data. See ***Importing Text File to Disk Cache*** (on page 453).

Shortcut Manager

- Shortcut Manager can now specify the type of shortcut to fire. Two new types of shortcut are introduced – Multi Time Frame and Action. See ***Shortcut Manager Operation Guide*** (on page 911).

QuoteSpeed

- Now directly support QuoteSpeed as a data feed. See ***Configuring QuoteSpeed*** (on page 29).

IQ Feed

- Support IQFeed 2.3.0.7, which provides significant performance improvements. See ***Configuring IQFeed*** (on page 27).

As of NeoTicker® 4.1, older IQFeed client are no longer supported. You must install IQFeed Client 2.3.0.7 or higher.

Internet EOD Data

- If you type the symbol in the following format EXCH:SYMBOL, EOD data will be retrieved from Lycos, providing an alternative data source in addition to the default Yahoo data. See ***Internet EOD Data*** (on page 537).

Turn Key Setups

- We've added a new argument -OPG. -OPG will open the previous session groups. ***See Turn Key Opera***tion (on page 594).

NeoBreadth®

- History Generator supports a batch mode. Note: History Generator interface is currently being revised. Some features are not implemented yet as of Beta 1. If you cannot click on something you see, it is not yet working. See ***NeoBreadth, Overview*** (see "Overview" on page 629) and ***NeoBreadth, Reference*** (see "Reference" on page 642).

Programming

- New Data object properties – LastBarsnum and LastDateTime. These properties are for accessing the bar position and time of the last bar in historical calculation. These properties are mainly intended for indicators that draw. Note that information about last bar price is deliberately left out to prevent future peeking. For example, Data1.LastBarsnum returns an integer of the barsnum of the last bar in the data series; Data1.LastDateTime returns a double of the date time of the last bar in the data series. See ***Data and Link Series Object, Properties and Methods*** (see "Properties and Methods" on page 1538).
- New function NTLib.GetSymbolSpecs to query SymbolInfoManager. See ***NTLib, Miscellaneous Routines*** (see "Miscellaneous Routines" on page 1624).
- New Data object properties to query tick level statistics. See ***Data and Link Series Object, Properties and Methods for Tick Statistics (Data Only)*** (see "Properties and Methods for Tick Statistics (Data Only)" on page 1544).
- New Data object properties to query information of the last trade. See ***Data and Link Series Object, Properties and Methods*** (see "Properties and Methods" on page 1538).
- New Heap object methods to allocate/resize lists and tables. See ***Heap, PHeap and GHeap, Properties and Methods for Label Referencing (List)*** (see "Properties and Methods for Label Referencing (List)" on page 1554) and ***Heap, PHeap and GHeap, Properties and Methods for Label Referencing (Table)*** (see "Properties and Methods for Label Referencing (Table)" on page 1556).
- Heap object can now be used to do price profiling, i.e. the calculation side of Volume Profile indicator. See ***Heap, PHeap and GHeap, Properties and Methods for Price Profiling*** (see "Properties and Methods for Price Profiling" on page 1556).

- New method Trade.GetBrokerPos(symbol, reportedsize, reportedprice, estsize, estprice) to retrieve position for a symbol at your brokerage. See ***Trade Object, Broker Properties and Methods*** (see "Broker Properties and Methods" on page 1578).

What's New from 4.0 to 4.01

4.01 is a maintenance release with limited features introduced. We recommend users to use 4.01 only if they encounter problems in 4.0.

Indicators

- Reftimeex indicator. See ***Reference Time Ex (reftimeex)*** (on page 2117).
- Latch indicator. See ***Latch (latch)*** (on page 1918).
- Countdown_tick indicator. See ***Countdown Tick (countdown_tick)*** (on page 1803).

Time Chart

- A new indicator drawing style is introduced – background. This style will fill the complete pane with color and useful to turn an indicator into a background highlight tool. For example, with a SlowK indicator, you can set the drawing style to background, and in indicator editor, under color tab, set the 30-70 rule. This will make SlowK to highlight the pane background by using 30-70 rule. See ***Background Drawing Style for Indicator*** (on page 1124).

Quote Window

- New fields BATick10 and BATick16. These are variations of Tick10 and Tick16 that are based on bid/ask classifications, i.e. trade at bid, trade at ask (whereas Tick10 and Tick16 are based on up/down ticks). See ***Quote Window, Available Fields*** (see "Available Fields" on page 867).

CHAPTER 3

New Feature Supplement (Version 4.0 Official Release)

This chapter describes the changes from Version 3 to Version 4 of NeoTicker®.

If you want to transfer files from previous version of NeoTicker® to Version 4, go to ***Backup Tool*** (on page 429). Read the section Previous Version Converter.

This chapter lists the new features introduced guides you to the help pages. It only lists new features that have documentation. Minor refinements are not listed here. For a complete list, you should read the What's New file.

Introduction

There are over 200 improvements in NeoTicker® from version 3 to version 4. This guide provides a summary of the changes.

What you can expect in Version 4 compared to Version 3:

- More user friendly
- Better performance
- More stable
- Better user support
- Key new features: Order Interface to live broker, Dynamic Table, CME/CBT Depth Support, Symbol Log Tool, Screen Shot Tool, Backfill, third party developer support, MB Trading as data feed.
- Areas receive significant enhancements: Time Chat, Quote Window, Simulation Server, Trade Simulator, System Performance Viewer, Disk Cache, eSignal connection, QCharts connection, TC2005, Jurik indicator support, formulas, script/IDL programming, OLE Automation.

In the sections below, we will first give a high level description of the improvements. The last section lists the details of the changes.

General Usability

Many controls have been revised to behave more logically. We also make control behavior more consistent among different controls. Areas we have made significant improvements in usability includes Time Chart, Quote Window, System Performance Viewer, Holiday List Manager and indicator editing.

For power users, we've introduced more hot keys to speed up operations.

We've completed our locale support. For users whose computer is set to non-US locale, you can expect to use date time and numbers in your locale's format and the files are fully compatible across different locales.

General Performance

Overall performance has been improved. This is achieved by optimizing file performance, tick processing and memory. Version 4 in general feels slicker than version 3.

Stability

We've added preventive measures in many areas. These are areas that work under normal circumstances, but can cause stability problem when NeoTicker® is under stress, e.g. heavy CPU load, near full memory usage, large number of symbols.

Help System

More places are now linked to help file through context sensitive help (F1 key).

The help file has been revised. New materials have been added (documentation for new features, new prebuilt groups, etc). Existing materials have been revised.

In version 4, we are also starting to make better integration between the program and our online resources (forums, etc).

Time Chart

Time chart now knows how to change holiday and trading time according to the primary symbol in the chart. It will look up the information from Symbol Info Manager.

A Quick Jump menu button is added to window caption for accessing commonly used chart features.

A broken line style (Visual Break) has been introduced. You can set up conditions that stop and resume drawing of data series and indicator lines.

You can color data series and indicator using one of the many built-in coloring scheme, e.g. Color according to up/down, hour of day, etc. Coloring is formula-based. So you can easily introduce your own customized coloring scheme.

Drawing tools have been extensively revised. We've made the drawing tool editors more user friendly. New drawing tools are introduced – 3 Point Fibonacci, 3 Point Projection and Arrow Tools. Current drawing tools are revised to introduce new functionalities - Fibonacci Tools support extension and retracement; Marker tools can draw arrow.

Performance has been improved by rewriting some of the more CPU intensive parts of time charts like data loading, indicator calculations, float markers and cursor value.

In version 3, Meta Style can turn indicator into data only if the number of plots in the indicator exactly matches a specific Meta Style's number of plots. In version 4, this limitation is removed. As long as there sufficient number of plots, the Meta Style can be applied.

There is better integration with other parts of NeoTicker® - User Defined Symbols, NeoBreadth® Symbols and trading systems.

Other new features and improvements in charts – Momentum Bars Superposition, optional indicator dependency settings, Data View improvements, new legend display options, improved usability, historical data backfill.

Quote Window

For Quote Window, focus of improvement is usability. You will find new features to help you use Quote Window better. For example, you can now assign color code to symbols, undo sorting, easily assign predefined time frames, verifying row filtering formula etc. Overall, you can expect Quote Window to be easier to use and works better with other parts of NeoTicker®.

Dynamic Table

Dynamic Table is a new type function window. A Dynamic Table contains one or more sheets. Each sheet can hold a Dynamic Grid. Dynamic Table is for users who wants a more flexible type of quote service than Quote Window can provide.

Level 2/CME Depth/CBT Depth

Level 2 window is now a general depth data tool. In addition to Level 2 data, you can use it to display CME/CBT depth data.

Pattern Scanner

We've made some minor adjustment to Pattern Scanner.

Option Chain

When connecting to eSignal, option symbols can be resolved directly from stock symbol.

Simulation Server

Biggest change in Simulation Server is in the quality of simulation. It now supports:

- Multiple days replay
- A higher maximum replay speed
- Improved response
- Randomized tick arrival for ticks that have identical time stamp.

Trade Simulator

Trade Simulator can now handle many more symbols. It also supports partial fill simulation and fraction entries.

Order Interface

Order Interface is completely rewritten for Version 4. It is now a full fledge middleware between NeoTicker®'s analytical tools (charts, quote window, trading systems), order service (real-life brokerage and Trade Simulator) and performance analysis tools (System Performance Viewer). It supports features such as position tracking, many types of order entry forms, account management, and an open architecture to plug-in different brokers.

Currently, NeoTicker® can connect to MB Trading, Interactive Brokers, and any brokerage that is supported by Ninja Trader.

System Performance Viewer

It is now easier to analyze extremely large system testing. System Performance Viewer can now load large performance files faster. You can query by regular expression and equity graph plotting is much more efficient to handle large number of data points.

User Defined Symbols/NeoBreadth

You can now set symbols that are defined by spread and ratio by tick.

On NeoTicker® startup, you can now interrupt User Defined Symbol/NeoBreadth® initialization, and later resume the initialization manually.

Cache Manager – Disk Cache

Main areas of change are importing and downloading historical data into disk cache. You will find more options to make the import and download more flexible.

Holiday List

Holiday List has been rewritten. The rewrite takes care of two issues:

- The new manager is more user friendly and in general feels slicker.
- The old holiday list was identified as one of the problematic area in terms of overall stability. The rewrite solves this problem.

Symbol Log Tool

This is a new tool that allows you to write down logs for symbols. Symbol logs are persistent between sessions and are accessible from Quote Window.

Screen Shot Tool

A new screen shot tool is introduced. You can save screen shots to file, upload to a website, etc.

Data Feeds/Services

Backfill

The biggest news on the data side is Backfill. Backfill is new feature that benefits: users who use broker data feed (with no historical data) and users who use NeoTicker® as an afterhour research tool. Backfill will automatically fill in missing historical data (EOD and minute) in charts, quote formulas, etc. Because of the low cost nature (free or very reasonable subscription fee), it is a perfect tool for users who are looking for an entry level data service or use NeoTicker® as an afterhour research tool.

For example, we are now recommending users who look for an entry level data service to use MB Trading with LiveChart subscription. This combination provides lots of value at a price that is unbeatable.

MB Trading as Data Feed

MB Trading can be used as a data feed with historical data provided through LiveChart. It supports stocks, futures and Level 2 data.

eSignal

There are many improvements made when you use eSignal with NeoTicker®. We've made the connection more stable and performs better. New features include fundamental data support, better server status reporting, the ability to resolve full names of futures and indices and support of CME/CBT depth data.

QCharts/Quote.com

If you use QCharts with NeoTicker®, you will notice the connection speed is much improved. We've also added fundamental data support.

TC2005

We have upgraded our Telechart support from TC2000 to TC2005.

Jurik Indicators

The Jurik indicators wrappers have been rewritten to supporting dynamic and variable implementation of Jurik indicators.

Formula

We've greatly improve the performance of formula. Formula now runs almost as fast as IDL indicators.

Quote Window formula now provides a construct to override the default number of days to load when evaluating indicators.

New functions have been added:

- A Choose function for multiple conditions and results. You no longer need to use very deep if functions for this purpose.
- Format output to Report window.
- Hexadecimal functions
- Color space functions
- Visual break functions.
- CurrentBar function to report current bar.
- Order Interface functions.
- MakeIndicator to support indicators with multiple plots and different time frame from hosting formula.
- ParamIs function.
- Level 2 functions in quote formula.

Script/IDL Programming

Many new object methods and properties have been added to support the new features in Version 4.

A new class of update method is introduced (Timer Indicator). Timer Indicator will update on timer in addition to update by tick/bar.

In the installation, we've added more IDL examples.

OLE Automation

New OLE Automation methods have been added – to drive quote window, manipulate charts and user defined symbols.

Protection

We've introduced new features specifically for third party developers. Features include write protecting function windows, indicator encryption and support for product activation.

Details

General Usability

- Indicator Quick Reference has been improved. You can now sort and search indicators. See ***Indicator Quick Reference*** (on page 606).
- User preference to turn off many confirmation dialogs for expert users. See ***Confirmation Dialogs*** (on page 497).
- F5 will now put main window to the bottom if main window is already at top. See ***Shortcut Summary*** (on page 399).
- CTRL-F9, CTRL-F10 as minimize/maximize/restore hot keys. They work like MS Office equivalent hot keys. See ***Shortcut Summary*** (on page 399).
- Default time frame setting for the whole package in User Preference. See ***Default Time Frame*** (on page 506).
- Clearly defined how to specify numbers and date time for computers under different locales. See ***Locales*** (on page 625).
- Under Order menu, new options to save and open all order forms. See ***Saving Multiple Order Forms*** (on page 733).
- Open prebuilt group window has been revised. See ***Tour of Prebuilt Groups*** (on page 57).

Stability

- Server log include new Error Events for tracking various problems. It is currently used to log memory problems. See ***Observing Real-Time Performance with System Log*** (on page 525).

Backfill

- EOD and minute data backfill from TickQuest website. See ***Setting Up Backfill*** (on page 541).

Time Chart

- You can set up data series and indicators to display in broken line style. See ***Broken Lines*** (on page 1123).
- Coloring bars for data series using pre-defined conditions and formula. See ***Color Data Series*** (on page 1111). Also see known issues below.

- Coloring indicator plots using pre-defined conditions and formula. See ***Color Indicators*** (on page 1116).
- Cursor value has been rewritten. See ***Cursor Values*** (on page 1023). Also see known issues below.
- Option to set indicator update dependency to primary link or to all links. See ***Update by Tick Indicators and Dependency*** (on page 1102).
- Indicator now has 3 visual break style: none, invalid and visual. See ***Broken Lines*** (on page 1123).
- New Data View configuration options. See ***Data View Window*** (on page 1030).
- Drawing tool setup window - Set Default button pops up a list of all templates to set to. See ***Drawing Tool Templates*** (on page 1143).
- You can use drawing tool pop up menu to open label properties editor directly. See ***Configuring Drawing Object Labels*** (see "Configuring Drawing Tool Labels" on page 1133).
- New Superposition chart type - Momentum Bars. See ***Superposition Chart*** (on page 1178).
- If your data feed supports company name, you can display a combination of company name and symbol in a data series' legend. See ***Company Name Support*** (on page 1071).
- Introduced a Fast mode for Crosshair Float Markers. Fast mode utilizes graphics card hardware to speed up Crosshair Float Markers drawings. See ***Crosshair Float Marker*** (on page 1072).
- Trading system features available in Chart Manager's indicator tab. See ***Chart Manager Window*** (see "Chart Manager" on page 1013).
- Buttons in Chart Manager's indicator tab have be re-organized to make space for trading system features. See ***Chart Manager Window*** (see "Chart Manager" on page 1013).
- Drawing tool setup window **Set Template** button now supports user defined template names. See ***Drawing Tool Templates*** (on page 1143).
- Drawing tool pop up menu added new command to load settings from template. See ***Drawing Tool Templates*** (on page 1143)
- Right clicking on a drawing tool will open pop up menu for the drawing tool without first selecting the drawing tool. See ***Pop up Menu Reference*** (on page 1217).
- Chart Manager supports named Time Frame defined in the Time Frame Manager. See Data Settings Tab in ***Chart Manager Window*** (see "Chart Manager" on page 1013).
- New Quick Jump caption bar menu providing short cuts such as Chart Manager, time frame changes, etc. See ***Quick Jump Button*** (on page 1036).
- New up arrow and down arrow tools. See ***Up Arrow and Down Arrow Tools*** (on page 1148).
- All drawing tools with channels (trend channels, fans, support/resistance, Fibonacci) are rewritten to support individual line width per channel, an improved interface for editing channel values. See ***Setting up Lines for Multi-line Drawing Tool*** (on page 1141).

- Fibonacci Level drawing tool now supports two additional drawing modes – extension and retracement, along with classical measurement mode. See ***Fibonacci Level Tool*** (on page 1168).
- Fibonacci Time drawing tool now supports extension drawing mode, along with classical measurement mode. See ***Fibonacci Time Tool*** (on page 1171).
- New drawing tools – 3 point projection, 3 point Fibonacci Level, 3 point Fibonacci time. See ***3-Point Projection*** (on page 1158), ***Fibonacci Level Tool*** (on page 1168) and ***Fibonacci Time Tool*** (on page 1171).
- When changing primary symbol, time frame (e.g. Trading time) will change based on the symbol. See ***Trading Time and Holiday List Settings*** (on page 1086).
- Options to control how trade simulator trades are displayed when there are trading systems in the chart. See ***Trading System Markings*** (on page 1185).
- New trading system display options – Position Line, Pos Winner color, Pos Loser color. See ***Trading System Markings*** (on page 1185).

Quote Window

- Sorting now takes sections into account. See ***Vertical Sections*** (on page 862).
- You can undo sorting in quote window. See ***Sorting Quote Columns*** (on page 864).
- Row filtering now has three states: completely disabled, calculated but visually not applied, calculated and visually applied. See ***Row Filtering*** (on page 864).
- CTRL-C, CTRL-V to copy and paste data across quote windows and external programs like Excel. See ***Quote Window Copy and Paste*** (on page 881).
- Setup supports pre-defined time frames. See ***Specifying the Data Settings for Indicator Calculation*** (on page 859).
- New fields – BPosAvgEntry and BPosSize for querying the position average entry price and position size for the symbol in Order Interface, i.e. If Order Interface is connected to your broker, these fields will tell the average entry price and position for the symbol in your brokerage account. See ***Position Information as Quote Fields and Formula Functions*** (on page 782).
- New quote fields FirstLog, LastLog to query/edit logs entered by Symbol Log Tool. See ***Symbol Log Tool Operation Guide*** (on page 934).
- New quote field Marked to mark a symbol. You can operate on marked symbols through the **Marked** sub menu. See ***Marking Symbols*** (on page 881).
- New quote field ColorCode to display/assign color code to a symbol. See ***Color Coding Symbols*** (on page 882).
- You can adjust quote window update speed in quote setup. See ***Adjusting Update Speed*** (on page 883).
- You can modify how quote window behaves when it receives a symbol list from another window. See ***Receive Symbol List Option*** (on page 884).
- Undo sort can restore quote window to the order before start of timer sort. See ***Sorting Quote Columns*** (on page 864).

Indicators

- New indicator Typical Price, which equals (High + Low + Close) / 3. See ***Indicator Reference, R-Z*** (see "R-Z" on page 1700).
- New xcross, xcrossconst indicators – returns values if either xabove or xbelow happened. See ***Indicator Reference, A-G*** (see "A-G" on page 1681).

Dynamic Grid

- You can re-configure the double click action of a Dynamic Grid. See ***Double Click Action*** (on page 563).
- Added copy current cell to all cell capability (with and without symbol settings). See ***Copying a Cell*** (on page 559).

Dynamic Table

- Dynamic Table is a new type function window. A Dynamic Table contains one or more sheets. Each sheet can hold a Dynamic Grid. See ***Dynamic Table Operation Guide*** (on page 565).

Simulation Server

- Supports multiple day replay. See ***Simulation Server Operation Guide, Running Simulation Server*** (see "Running Simulation Server" on page 903).
- You can control simulation speed from 0X to 300X. See ***Running Simulation at Different Speeds*** (on page 905).
- New button to quickly set to use demo data. See ***Simulation Server Operation Guide, Running Simulation Server*** (see "Running Simulation Server" on page 903).
- Added options tab to randomize symbol tick arrival and an option to deliver tick data like a stream. See ***Simulation Server Operation Guide, Controlling Tick Arrival*** (see "Controlling Tick Arrival" on page 907).

Trade Simulator

- Partial fill simulation. See ***Partial Fill Simulation*** (on page 764).

Order Interface

- Many changes has been made to support order placement to brokers. Order related documentation has been rewritten and reorganized. See ***Order Related Topics*** (on page 693) and ***Tutorial: Placing Orders to Your Broker*** (on page 245).
- Interactive Brokers order placement support. See ***Configuring Order Placement to Interactive Brokers*** (on page 51).
- MB Trading order placement support. See ***Configuring Order Placement to MB Trading*** (on page 53).
- Order Interface has been rewritten. See ***Order Interface*** (on page 735).

- Position tracking feature - when position of a symbol is tracked, position information is retrieved from broker automatically when program starts. To track a position, either add the symbol manually to track in Account Manager, or set the auto tracking option in Order Interface Setup (default is on). In the latter case, if a symbol has been used in an order, the position for the symbol will be tracked. See ***Position Tracking*** (on page 773).
- Ninja Trader support. See ***Configuring Order Placement to NinjaTrader*** (on page 53).
- In Account Manager, **Open Orders** tab, command is provided to mark staled order as cancelled. See ***Stale Order Handling*** (on page 780).
- Can open Account Manager by clicking the account connection status. See ***Manual Order Placement, General Usage Section*** (see "General Usage" on page 705).
- Order Interface Setup now provides sound options on order on action and status changes. See ***Order Defaults and Completion Related Topics, Sound Setup Section*** (see "Sound Setup" on page 749).
- Account Manager, under Position tab, Remove Selected button can remove position tracking for current row or a selection of rows. See ***Position Tracking*** (on page 773).
- New order form - Strategic Order Entry form. See ***Strategic Order Entry Form*** (on page 714).
- Auto enabled order interface option. See ***Connecting to Your Broker, Advanced Options*** (see "Connection Related Options" on page 752).
- Option to not auto connect to a broker at startup. See ***Order Interface, Connection Related Options*** (see "Connection Related Options" on page 752).
- IB Order Server 2.0 is released simultaneously. See ***Configuring Order Placement to Interactive Brokers*** (on page 51).
- UI reorganized to support currency field and more security types. See ***Order Interface, Security Type, Exchange and Currency Information*** (see "Security Type, Exchange and Currency Information" on page 747).

MB Trading as Data Feed

- MB Trading can be used as a data feed, with historical back fill capability with LiveChart. See ***Configuring MB Trading as Real-Time Data Feed*** (on page 33).
- Support Level 2 data. See ***Level 2 Window Operation Guide*** (on page 615).

Internet EOD Data

- In the manager, added option to set time out. See ***Internet EOD Data Connection Setting*** (on page 609).

System Performance Viewer

- System Performance Manager is expanded to handle all opened System Performance Viewers. See ***System Performance Manager*** (on page 947).

- System Performance Viewer can auto refresh with parent window. See ***System Performance Viewer, Auto Refresh*** (see "Auto Refresh" on page 986).
- System Performance Viewer short cut to save itself without going through the parent window. See ***System Performance Viewr, Saving*** (see "Saving" on page 987).
- Equity graph can plot using daily, week, monthly and yearly equity changes. See ***System Performance Viewer, Summary / Equity Graph*** (see "Summary / Equity Graph" on page 961).
- Regular expression searching in position query, providing searching features such as wild card, etc. See ***System Performance Viewer, Positions / Position Query*** (see "Positions / Position Query" on page 969).

User Defined Symbol/NeoBreadth®

- You can interrupt UDS on startup. See ***Interrupting UDS on Startup*** (on page 1250).
- You can interrupt NeoBreadth® on startup. See ***Interrupting NeoBreath® on Startup*** (on page 641).
- Spread and ratio style now supports update by tick. See ***User Define Symbol Manager, Settings*** (see "Settings" on page 1252).
- Allow specifying weight in symbol list under Weighted Index tab. See **Has Weight** in ***User Define Symbol Manager, Settings*** (see "Settings" on page 1252).

Cache Manager - Disk Cache

- You can choose symbols to download when requesting historical data. See ***Retrieving a Range of Data to Fill Disk Cache*** (on page 455).
- Option to force update file when requesting data from data vendor to fill disk cache (**Forced Update On All Files**). See ***Retrieving Data from Data Vendor Every Night*** (on page 456).
- When importing tick text file into Disk Cache, an extra field TradeType is supported. TradeType determines the trade type of the tick. 0 for trade. 1 for bid. 2 for ask. See ***Compatible Data File Formats*** (on page 530).
- When importing text file, new option to adjust time stamp by number of seconds. See ***Importing Text File to Disk Cache*** (on page 453).
- Settings for importing multiple files can be saved for later use. See ***Importing Text File to Disk Cache*** (on page 453).
- Option to adjust volume by a multiple when importing text files. See ***Importing Text File to Disk Cache*** (on page 453).
- When downloading historical data, option to start from first missing date. See ***Cache Manager, Retrieving Data from Data Vendor Every Night*** (see "Retrieving Data from Data Vendor Every Night" on page 456).
- When downloading historical data, option to generate zero length cache file. See ***Cache Manager, Retrieving Data from Data Vendor Every Night*** (see "Retrieving Data from Data Vendor Every Night" on page 456).

Cache Manager - RAM Cache

- In RAM Cache option, under Bars Per Series tab, a Use Defaults button has been added. See ***Adjusting RAM Cache Size*** (see "Bars Per Series" on page 464).

Script/IDL Programming

- `MakeIndicator` can refer to specific plot in a source link. See ***Using Other Indicators*** (on page 1479).
- New method ItSelf.UpdateIndicator. This method is used for accessing indicator instance that have been already created by MakeIndicator. UpdateIndicator is much faster than MakeIndicator. Indicators using UpdateIndicator will see significant performance improvements over implementation that relies on MakeIndicator alone. See ***Using Other Indicators*** (on page 1479) and ***Example: UpdateIndicator*** (on page 1488).
- New properties ItSelf.VisualBreak, ItSelf.VisualBreakEx[AIndex]. Setting visualbreak to true will make a plot breaks when the plot's visual break style is set to “visual”. See the properties and methods section of ***ItSelf Object*** (on page 1525).
- PrevVisualBreak[BarsAgo], PrevVisualBreakEx[Plot, BarsAgo] for accessing previous visual break settings in script/IDL. See properties and methods section of ***ItSelf Object*** (on page 1525).
- Programming topic for setting visual breaks. See ***Visual Break Programming*** (on page 1489).
- You can create user manipulable drawing tools with `ChartDrawingObjects`. See ***ChartDrawingObjects and DrawingObjects Objects*** (on page 1565).
- New drawing object methods `GetPoint` for returning the coordinate of a drawing tool's control point. See the method and properties section of ***ChartDrawingObjects and DrawingObjects Objects*** (on page 1565).
- Pane related properties for drawing tools: PaneCount and Pane. See the methods and properties section of ***ChartDrawingObjects and DrawingObjects Objects*** (on page 1565).
- Pane related property for indicators: `ItSelf.Pane`. See the methods and properties section of ***ItSelf Object*** (on page 1525).
- Properties to help gheap and memory management: `ItSelf.FirstCall, ItSelf.RemoveCall, ItSelf.UniqueId`. See ***Initialization and Finalization*** (on page 1468).
- Property to identify whether a hosting environment is time driven or bar driven: `ItSelf.TimeDriven`. See the methods and properties section of ***ItSelf Object*** (on page 1525).
- Property to identify whether an indicator plot is visible: `ItSelf.PlotVisible`. See the methods and properties section of ***ItSelf Object*** (on page 1525).
- Heap, PHeap, GHeap now support integer list and table (2-dimensional array). See ***Heap, PHeap and GHeap Objects*** (on page 1549).

- Now supports updated by timer indicator. See ***Indicator Updated by Timer*** (on page 1467).
- Update by primary link can now be set at design time. See ***Updated by Primary Link Only*** (on page 1478).
- You can program an indicator to open its own help file. See ***Creating Help for Indicator*** (on page 1451).
- Modified Indicator Setup to include settings for Class, Timer Interval, Primary Link Only, Notify On Removal and Help File. See ***Indicator Specification*** (on page 1627).
- For bar driven charts, indicator drawing tools, `cocExact` provides a way to map bar number to X-axis. See the methods and properties section of ***ChartDrawingObjects and DrawingObjects Objects*** (on page 1565).
- New drawing objects cotUpArrow and cotDownArrow are introduced. See ***ChartDrawingObjects and DrawingObjects Objects*** (on page 1565).
- New properties ArrowStyle, HeadOnly for markers, up arrows and down arrows. See ***ChartDrawingObjects and DrwingObjects, Methods and Properties*** (see "Methods and Properties" on page 1568).
- New Trade object properties DefaultOrderSize and DefaultOrderSizes[index]. See ***Trade Object, System Setting Properties*** (see "System Setting Properties" on page 1604).
- When setting up indicator parameter defaults, you need to specify floating point numbers in a form that is compatible with script (for non-English locales). See ***Locales*** (on page 625).
- When setting up indicator parameter defaults for date/time, you can use special keywords NOW, TODAY for generating parameters when user is using the indicator. See ***Indicator Specification*** (on page 1627).
- DrawingObjects and ChartDrawingObjects added property LabelsVisible[ID] : boolean. See ***ChartDrawingObjects and DrawingObjects, Methods and Properties*** (see "Methods and Properties" on page 1568).

OLE Automation

- Quote window commands – ClearAllSymbols, AddSymbol(Symbol, IgnoreRepeat). See ***FWQuote*** (on page 670).
- RecalcIndicatorOneParamChange (previously undocumented). See ***OLE Automation, FWTimeChart Object*** (see "FWTimeChart Object" on page 672).
- New chart OLE method JumpTo for jumping into a specific time of a chart. See ***OLE Automation, FWTimeChart Object*** (see "FWTimeChart Object" on page 672).
- `UserDefineSymbol.UpdateWithVolume(symbol, price, volume)` – update an user defined symbol symbol with price and volume. See ***OLE Automation, UserDefineSymbol Object*** (see "UserDefineSymbol Object" on page 680).

eSignal

- Symbol usage and limit are shown on main window caption bar. See ***Configuring eSignal Connection*** (on page 19).

- Fundamental data support. See ***Quote Window, Fundamental Data Support*** (see "Fundamental Data Support" on page 885).

System Monitor Window

- Added order id column. See ***System Monitor, Basic Operations*** (see "Basic Operations" on page 940).

Power Indicators

- Volume Profile added parameters Special Break and Starting Pos to control profile breaking and drawing position. See ***Volume Profile*** (on page 1355).

Symbol Info Manager

- New symbol fields: User Field 1 and User Field 2. These fields are used by custom scripts/IDL by retrieving the field values using `NTLIB.GetUserFields` function. See Symbol Info Manager, Field Description.
- Supports time frame setting. See Symbol Info Manager, Field Description.
- UI reorganized to support currency field and more security types. See ***Symbol Info Manager*** (on page 921).

Formula

- Generic function `Choose(condition1, result1, condition2, result2, ... , otherwise)`. See ***If Condition and Choose Function*** (on page 262).
- Generic function ReportFormat(ReportWindow, condition, format1, item1, ...). See ***Generic Functions - Others*** (on page 302).
- Quote Window Formula supports an overriding number of days specification. Overriding is also supported by Dynamic Grid, DDE Link, User Defined Symbol, NeoBreadth, Cluster, Quote Memo, Dynamic Quote Window. See ***Time Frame Specification*** (on page 293).
- New function hex2int(string). Convert a hex number in quoted string to integer, e.g. hex2int(“FFFFFF”). See ***Generic Functions - Math Related*** (on page 298).
- New function rgb(r, g, b) converts RGB color values to internal color value. The parameters r, g, b must be from 0 to 255. See ***Generic Functions - Color Related*** (on page 300).
- New function hsl(h, s, l) converts a color in HSL color space to internal color value. The parameters h, s, l must be from 0 to 1. See ***Generic Functions - Color Related*** (on page 300).
- New function hsv(h, s, v) converts a color in HSV color space to internal color value. The parameters, h, s, v must be from 0 to 1. See ***Generic Functions - Color Related*** (on page 300).

- New properties visualbreak1, visualbreak2, etc. for formulas used in charts. Setting visualbreak to true will make a plot breaks when the plot's visual break style is set to “visual”. See ***Visual Breaks*** (on page 277).
- CurrentBar(N) returns a sequential index starting from 1. When the script is first called to do calculation, usually that is the first bar right after the indicator's Min Bars setting, the current bar is set to 1. See ***Accessing the Current Bar Number*** (on page 287).
- New functions BrokerPosSize and BrokerPosAvgEntry. These are the function equivalents to BPosAvgEntry and BPosSize quote fields. As functions, BrokerPosSize and BrokerPosAvgEntry allows you to specify account type and symbol. See ***Position Information as Quote Fields and Formula Functions*** (on page 782).
- New function MakeIndicator. This allows making a full indicator within a formula, i.e. supports multiple plots and its own time frame. See ***MakeIndicator in Formula*** (on page 287).
- New function DefaultOrderSize. See ***Misc Formula Trading System Functions*** (on page 327).
- New function ParamIs for formula indicators. See ***Accessing User Specified Parameters*** (on page 282).
- Indicator formula added barsnum field for linked data series and compressed series. See ***Accessing Data Links*** (see "Accessing Data Links and Fields" on page 278) and ***Compressing Series to Higher Time Frame*** (on page 283).
- New quote formula functions to access Level 2 data: L2BidOrders, L2AskOrders, L2BidAccOrders, L2AskAccOrders. See ***Context-Driven Functions for Quote Window Formula Columns*** (on page 306).

Level 2/CME Depth/CBT Depth

- Quick Setup. See ***Level 2 Window, Quick Setup*** (see "Quick Setup" on page 621).
- Level 2 window can be configured to display CME/CBT depth data. Currently this is feature is useful for eSignal and MB Trading users only. See ***Level 2 Window, CME/CBT Depth Data*** (see "CME/CBT Depth Data" on page 622).
- Addition options for configuring Bid/Ask table. See ***Level 2 Window, Bid Ask Table*** (see "Bid Ask Table" on page 621).

Holiday List

- Completely rewritten. See ***Holiday List Manager*** (on page 1236).

DTN IQ/IQFeed

- Fundamental data support. See ***Quote Window, Fundamental Data Support*** (see "Fundamental Data Support" on page 885).

QCharts/Quote.com

- Fundamental data support. See ***Quote Window, Fundamental Data Support*** (see "Fundamental Data Support" on page 885).

Symbol Log Tool

- This is a new tool that allows you to write down logs for symbols. See ***Symbol Log Tool Operation Guide*** (on page 934).

Protection

- For write protecting function windows, see ***Function Window Write Protection*** (on page 839).
- For encrypting your work for personal use, see ***Locking Indicators, Windows and Templates*** (on page 841).
- For encrypting your work as a third party developer, see ***Third Party Developer Guide*** (see "Guide To Protection" on page 1659).
- For activating a product from a third party developer, see ***Activating Products from Third Party Developers*** (on page 405).

Misc

- Naming change in Disk Cache location settings to make the setting more obvious. See ***Changing Disk Cache Location*** (on page 448).

Pattern Scanner

- By turning off certain features, you can make pattern scanner not to use any real-time symbols for users with setups that are very close to their symbol limit. See ***Eliminating Real-time Symbol Usage in Scanning*** (on page 814).

Screen Shot Tool

- A new screen shot tool is introduced. You can save screen shots to file, upload to a website, etc. See ***Screen Shots*** (on page 588).

CHAPTER 4

Getting Started

In This Chapter

Welcome

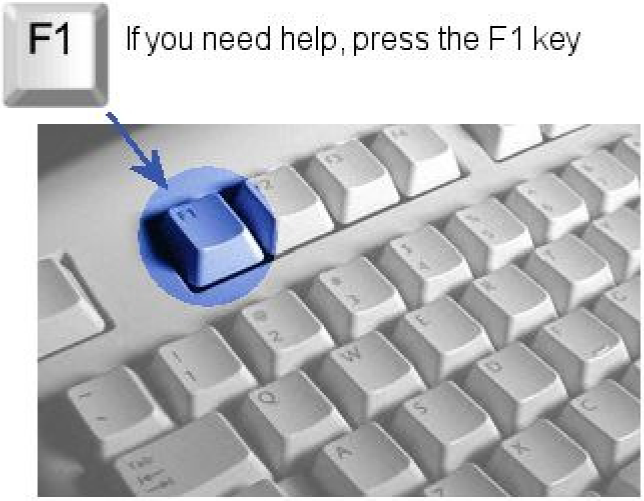
Thank you for using NeoTicker®.

NeoTicker® is TickQuest's advanced real-time trading platform for traders who need their trading tools to be reliable and dependable. Essential real-time tools like quotes, news, charts, time and sales, and alerts are delivered in a single versatile platform. NeoTicker®'s flexible design delivers powerful features from programmability to specialized trading tools.

To find out more, refer to ***What NeoTicker® can do for You*** (on page 3).

New users to NeoTicker® please go to ***Must Read*** (on page 4).

When You Need Help



If you need help, press the F1 key

What NeoTicker® can do for You

NeoTicker® is an advanced trading simulator and real-time trading platform.

We are independent from data vendors and brokers. This allow us to develop NeoTicker® to provide tools that are beyond what data vendors and brokers are willing to offer, satisfying the requirements for the most demanding traders. NeoTicker® is the tool of choice for many professional traders and funds.

Benefits

- NeoTicker® puts you in an advantageous position against other traders. NeoTicker® is feature rich and highly user customizable. You can make faster and better trading decisions.
- NeoTicker® is data vendor and broker independent. You have the freedom to choose the data vendor and broker that best fit your need. Setting up a backup data feed is easy.
- NeoTicker® provides realistic simulation to help you gain familiarity to any trading scenarios.
- We constantly improve NeoTicker®. You will have access to the latest trading technology and are never left behind.
- We provide top quality support for our customers.

To Find Out More

First, you should read ***Must Read*** (on page 4) for information on how to set up and run NeoTicker®.

Then, you can open one of the prebuilt groups to see what NeoTicker® can do. See ***Tour of Prebuilt Groups*** (on page 57).

Must Read

This section contains important information for new users.

System Requirements

Go to ***System Requirements*** (on page 6) to see if your computer can properly run NeoTicker®.

Installation and Launching NeoTicker®

Go to ***Installing and Starting NeoTicker*** (see "Installing and Starting NeoTicker®" on page 7) for more information.

Running NeoTicker® as a Simulator

If you plan to set up NeoTicker® as a simulator, go to ***Setting up NeoTicker® as Simulator*** (on page 9).

Running NeoTicker® as a Real-Time Trading Platform

If you do not have a data feed yet, go to ***Choosing A Data Feed*** (on page 16).

If you already have a data feed, go directly to the setup section listed below.

For DTN satellite feed users, go to ***Configuring Instant Data Server for DTN Satellite Feed*** (on page 21).

For MB Trading customers, go to ***Configuring MB Trading as Real-Time Data Feed*** (on page 33).

For eSignal users, go to ***Configuring eSignal Connection*** (on page 19).

For QCharts/Quote.com users, go to ***Configuring Quote.com Connection*** (see "Configuring QCharts (Quote.com) Connection" on page 39).

For IQFeed users, go to ***Configuring IQFeed*** (on page 27).

For myTrack users, go to ***Configuring myTrack Connection*** (on page 32).

For Interactive Brokers users, go to ***Configuring Interactive Brokers Connection*** (see "Configuring Interactive Brokers as Real-Time Data Feed" on page 23).

If you are using data feed that is not listed above, go to ***Configuring a DDE Data Feed*** (on page 41).

Connecting to a Broker to Place Orders

If you want to connect NeoTicker® to your broker to place orders, go to ***Connecting to Your Broker*** (on page 49).

Tour of Prebuilt Groups

NeoTicker® comes with many prebuilt groups that you can use and modify. You can ***go to Tour of Prebuilt*** Groups (on page 57) to see them.

New User Tutorials

New user tutorials are available to help new users start using NeoTicker® quickly, go ***to New User Tutori***als (on page 99). Of the many tutorials, you should at least go through:

Tutorial: Main Window and Sessions (see "Tutorial: Creating a Great Time Chart" on page 105)

Tutorial: Creating a Great Time Chart (on page 105)

Tutorial: Creating a Great Quote Window (on page 125)

Operation Guide

To learn how to use a specific feature, go to ***Operation Guide*** (on page 395).

Online Help

At anytime using NeoTicker®, you can press the F1 key to use the online help. The online help will check the current feature you are using and open the relevant help page.

Formula

Formula language is an integral part of NeoTicker®. Formulas let you easily and quickly custom configure NeoTicker® for your usage. More information about formula can be found in ***Formula Topics and Tutorials*** (on page 255).

Programming

NeoTicker® supports many scripting and programming languages. For more information, go to ***Programming Guide*** (on page 1361).

When You Need More Help

You can go to our ***Support Forum*** http://www.tickquest.com/forums to get help.

There is a also Yahoo! NeoTicker® ***user group*** (http:\\groups.yahoo.com\group\neoticker), you can exchange idea will fellow NeoTicker® users.

System Requirements

In general, it is not recommended to use NeoTicker® with Microsoft Windows 95, 98, or Me.

Computer Requirements

- Pentium III 450 MHz or faster
- 1024 x 786, 16-bit color
- 256M RAM
- Windows NT4(SP4 or higher), 2000(SP1 or higher), XP Home/Pro(SP1 or higher)

Additional Requirements for Real-time Scanning

- 512M RAM or more

Additional Requirements for Satellite Feed

- Pentium 4 1 GHz or faster
- 512M RAM or more
- 30G free hard disk space

Data Source Requirements

To receive real-time data, you must subscribe to a compatible data vendor. For more information, go to ***Choosing and Configuring a Real-time Data Feed*** (on page 15).

Installing and Starting NeoTicker®

Installing NeoTicker®

To install NeoTicker®, follow the instructions that come with the NeoTicker® package.

Starting NeoTicker®

To launch NeoTicker®, either:

- From Windows' start button, choose: **Start>Programs>TickQuest>NeoTicker 4**
- Double click NeoTicker®'s icon on your desktop.

Starting NeoTicker® without Connecting to a Data Feed

To launch NeoTicker® without connection to a data feed, hold the SHIFT key after NeoTicker®'s splash screen appears.

How to Purchase

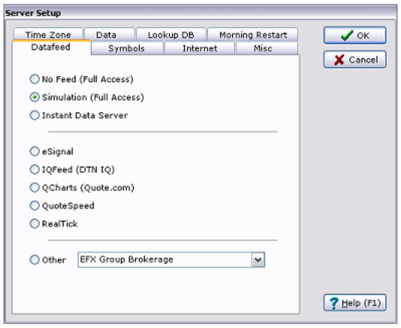
Simply click to TickQuest's NeoTicker® purchase page:

Purchase Online (http:\\www.tickquest.com\firsttimebuy.html)

CHAPTER 5

Setting up NeoTicker® as Simulator

If this is the first time you run NeoTicker®, the Server Setup window will pop up.



○ No Feed (Full Access)
● Simulation (Full Access)
○ Instant Data Server

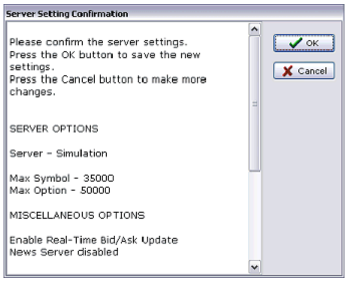
○ eSignal
○ IQFeed (DTN IQ)
○ QCharts (Quote.com)
○ QuoteSpeed
○ RealTick

○ Other EFX Group Brokerage

(If you do not see from NeoTicker®'s main menu, **Program>Server Setup** to open it.)

This window lets you configure NeoTicker® to different data servers. Because we are going to run NeoTicker® as a simulator, choose **Simulation**. Then press the **OK** button.

A Server Setting Confirmation window will open. This window summarizes the new server settings. Notice that in the window, there is a line that says "Server – Simulation". This will be your new server setting.



Please confirm the server settings.
Press the OK button to save the new settings.
Press the Cancel button to make more changes.

SERVER OPTIONS

Server - Simulation

Max Symbol - 35000
Max Option - 50000

MISCELLANEOUS OPTIONS

Enable Real-Time Bid/Ask Update
News Server disabled

Press **OK** button in Server Setting window.

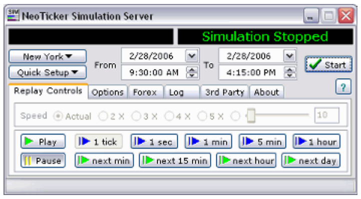
A window will open to remind you that NeoTicker® must be restart for the new server setting to take effect. Press **OK** button and NeoTicker® will close.

You have configured NeoTicker® to use Simulation Server for market data.

Once you have Simulation Server setup, you can restart NeoTicker®.

Starting Simulation

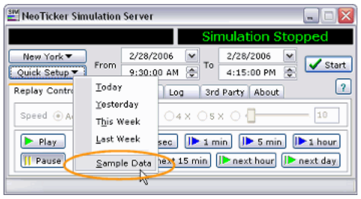
Simulation Server is a separate program from NeoTicker®. When you start NeoTicker®, NeoTicker® will launch Simulation Server. When Simulation Server starts, you will see it:



Note: When minimized, Simulation Server is minimized as a tray icon in the lower right corner of Windows task bar.

Let's try out Simulation Server with the build-in sample data to get a feel how it works.

In Simulation Server, press the **Quick Setup** button, then choose **Sample Data**.



What you have done is to tell Simulation Server to use the data that comes with the installation. This is the data every one has.

The sample data consist of the trading data from for 2003/9/24 to 2003/9/26 for the following symbols (i.e. Symbols you can use in simulation):

$INDU – the Dow Jones Industrial Average Index.

$SPX – S&P 500 Index.

$NDX – Nasdaq 100 Index.

$TICK – A commonly used indicator broadcasted by the New York Stock Exchange.

MMM – Stock symbol for 3M.

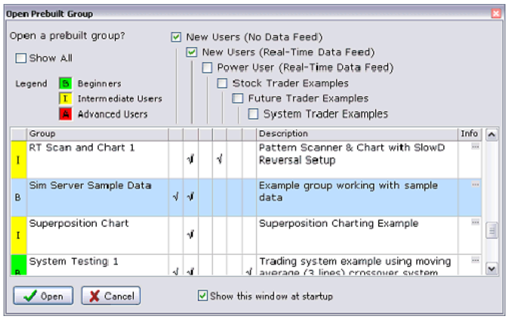
AAPL – Stock symbol for Apple Computer.

ESD_F – ES futures contract.

After you choose using sample data, NeoTicker® will automatically connect to Simulation Server. Your simulation session is set to 2005/9/25 9:30am.

Test Run

NeoTicker® will open the Open Prebuilt Group window. Prebuilt groups are the trading setups that comes with NeoTicker®.



Open Prebuilt Group

Open a prebuilt group?

□ Show All

Legend
🟩 B Beginners
🟨 I Intermediate Users
🟥 A Advanced Users

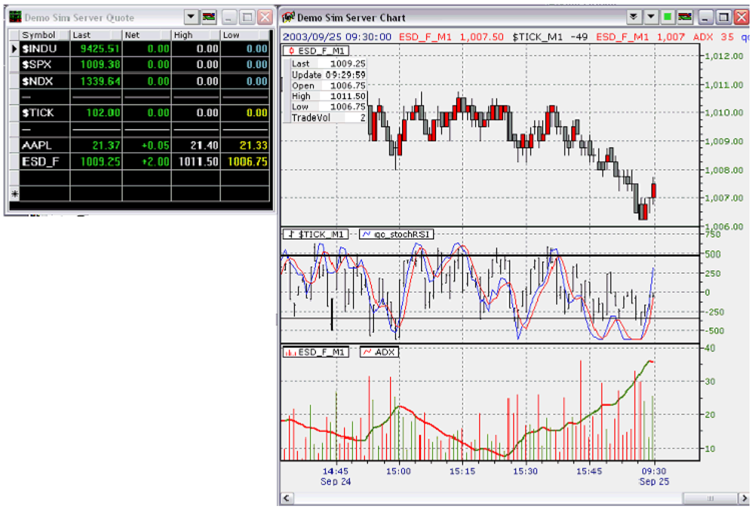
☑ New Users (No Data Feed)
☑ New Users (Real-Time Data Feed)
□ Power User (Real-Time Data Feed)
□ Stock Trader Examples
□ Future Trader Examples
□ System Trader Examples

	Group							Description	Info
I	RT Scan and Chart 1		√		√			Pattern Scanner & Chart with SlowD Reversal Setup	...
B	Sim Server Sample Data	√	√					Example group working with sample data	...
I	Superposition Chart		√					Superposition Charting Example	...
B	System Testing 1	√	√				√	Trading system example using moving average (3 lines) crossover system	...

☑ Show this window at startup

(If you do not see the Open Prebuilt Group window, you can open it by choosing from the main menu, **Group>Open Prebuilt Group**.)

Click on the group **Sim Server Sample Data** to select it and press **Open** button to open the group.



Demo Sim Server Quote

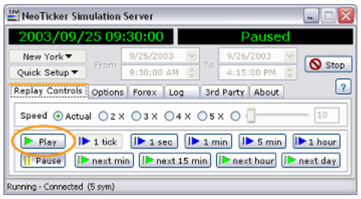
Symbol	Last	Net	High	Low
$INDU	9425.51	0.00	0.00	0.00
$SPX	1009.38	0.00	0.00	0.00
$NDX	1339.64	0.00	0.00	0.00
—				
$TICK	102.00	0.00	0.00	0.00
—				
AAPL	21.37	+0.05	21.40	21.33
ESD_F	1009.25	+2.00	1011.50	1006.75

Demo Sim Server Chart

2003/09/25 09:30:00 ESD_F_M1 1,007.50 $TICK_M1 -49 ESD_F_M1 1,007 ADX 35

Last	1009.25
Update	09:29:59
Open	1006.75
High	1011.50
Low	1006.75
TradeVol	2

In Simulation Server, press the Play button (Simulation Server may be hidden beneath other windows. You can double click on its tray icon to bring it up front).



NeoTicker Simulation Server

2003/09/25 09:30:00 Paused

From 9/25/2003 9:30:00 AM To 9/26/2003 4:15:00 PM

Speed ● Actual ○ 2 X ○ 3 X ○ 4 X ○ 5 X ○ 10

Running - Connected (5 sym)

Notice in quote window and time charts, the price starts to move. You are actually seeing the 2003/9/25 trading session replays right in front of your eyes.

Simulation works exactly like a real trading session. You can perform analysis, place orders, etc. This is the power of simulation. You can learn to trade in an environment that is virtually identical to a real-life session, without worrying about making costly mistakes.

Additional Information

For more information on how to operate Simulation Server, go to ***Simulation Server Operation Guide*** (on page 899).

Trade Simulator simulates a broker and handles orders you place. Trade Simulator is on by default (because NeoTicker® can connect to a real-life broker. If you run NeoTicker® as a Simulator, you should always double check you are using Trade Simulator). For more information on how to operate Trade Simulator, go to ***Trade Simulator*** (on page 755).

Checkout our ***Video Tutorials*** (http://www.tickquest.com/videotutorial.html). The Download Data For Sim Server video tutorial is scheduled to be released in March, 2006.

CHAPTER 6

Choosing and Configuring a Real-time Data Feed

In This Chapter

Choosing a Data Feed

Read this section if you are not currently subscribing to a data feed.

When you use NeoTicker® as real-time trading platform, you will need a data feed that provides real-time market data. Your choices are:

- Brokerage account that can send real-time market data, or
- Subscription to a real-time data provider

Feature Comparison

TickQuest maintain a ***Compatible Data Services*** (http://www.tickquest.com/NeoTicker/data.html) web page to help you select a data service to use with NeoTicker®. The page contains our latest survey of the available services.

Data from Brokerage Account

If you are new to trading, we recommend you to use MB Trading to provide you with real-time data, and use LiveCharts as a back fill service for historical data. This combination provides lots of value and is sufficient for many traders.

There are some limitations:

- Limited number of symbols (about 100 symbols for MB Trading).
- No historical tick data.
- Historical minute data limit to 500 bars.
- Trading data only. No market depth and streaming news service.

To configure MB Trading as a data service, go to ***Configuring MB Trading as Real-Time Data Feed*** (on page 33).

If you trade Forex only, EFX Group is an alternative. See ***Configuring EFX Group as Real-Time Data Feed*** (on page 36).

myTrack is another alternative if you already have a trading account with them. Please note that myTrack will charge an additional monthly SDK fee (the rate various depending on your account status. Consult myTrack for the exact rate) if you plan to use NeoTicker® with myTrack. To configure myTrack as a data service, go to ***Configuring myTrack Connection*** (on page 32).

We currently do not recommend using Interactive Brokers as a data feed.

Real-Time Data Provider

If your data requirements go beyond what brokers can offer, you can subscribe to one of the data service we support. We recommend the following services:

High performance Internet data services:

- eSignal
- QCharts
- QuoteSpeed
- RealTick

Low cost Internet data services:

- IQFeed

Ultra high performance satellite data service:

- DTN Satellite

eSignal, QCharts, QuoteSpeed and RealTick deliver data through Internet. DTN Satellite deliver full market data through satellite. For pros and cons of Internet vs. satellite data deliver, see section below.

For eSignal users, go to ***Configuring eSignal Connection*** (on page 19).

For QCharts users, go to ***Configuring QCharts (Quote.com) Connection*** (on page 39).

For QuoteSpeed, go to ***Configuring QuoteSpeed*** (on page 29).

For RealTick, go to ***Configuring RealTick*** (on page 31).

For DTN satellite feed users, go to ***Configuring Instant Data Server for DTN Satellite Feed*** (on page 21).

If you are looking for a low cost Internet data service, you can consider IQFeed. As a low cost solution, IQFeed has certain limitations but many users find IQFeed acceptable. If you are not sure, before you sign up with IQFeed, please check with support@tickquest.com. We can tell you whether IQFeed fits your need. For configuring IQFeed, see ***Configuring IQFeed*** (on page 27).

Internet Data Delivery vs. Satellite Data Delivery

Internet data delivery is suitable to all but the most demanding traders.

Pros:

- Low cost

- Ease of setup
- Data feed provider maintains the historical data for you
- Portable

Cons:

- Broadband Internet connection is required.
- Internet connection is not as stable as a dedicated channel like satellite
- Often the data feed provider limits the number of symbols you can receive. Even if there is no imposed limit, typical broadband connection is maxed out at around 2000 symbols. So Internet feed is not appropriate for full market scanning.
- Transmission speed can be inconsistent due to Internet traffic
- Historical data is limited by what your data feed vendor is willing to provide

Satellite feed sends you the full market trading data through satellite. Data is locally stored so it offers the best performance. However, additional hardware is required to receive satellite data.

Pros:

- Full market trading data collection
- Data is locally stored. Charting, scanning and heavy computation is extremely fast
- Consistent transmission speed
- Connection problems are rare
- You have full control of how much historical data you want

Cons:

- High cost
- Setup can be difficult
- Non-portable
- Requires local storage of data

Configuring eSignal Connection

This section is not applicable if you do not use eSignal as your real-time server.

Requirements

You need to install eSignal 7.6 or up before NeoTicker® can work. Follow the eSignal instructions to setup your connection to their service. Once that is completed, you can run NeoTicker® and other eSignal programs.

To download the latest version of eSignal, go to eSignal's ***download page*** http://www.esignal.com/download. The download will include both eSignal's data manager and its front end. NeoTicker® is a third party software that works with eSignal's data manager and ignores eSignal's front end.

Setup

To set up eSignal as the real-time data server:

1. Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.
2. Choose **eSignal** under the **Datafeed** tab.
3. Press the **OK** button. Exit and restart NeoTicker®.
4. There are optional information you can enter to fine tune your data feed connection. For more information, refer to ***Server Setup Reference*** (on page 517).

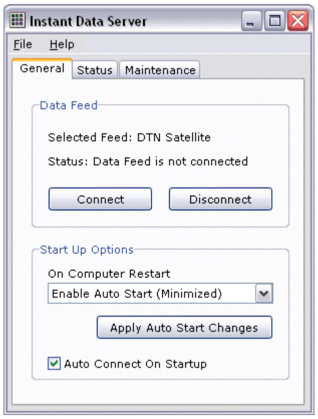
When you use NeoTicker® with eSignal, server connection is automatic. Most users do not need to set up additional parameters beyond telling NeoTicker® to work with eSignal.

eSignal Symbol Usage and Limit

You can check your symbol usage and limit on NeoTicker®'s main window caption bar.

Configuring Instant Data Server for DTN Satellite Feed

Instant Data Server (IDS) is required for connecting NeoTicker® to DTN satellite feed.



Selected Feed: DTN Satellite

Status: Data Feed is not connected

On Computer Restart: Enable Auto Start (Minimized)

☑ Auto Connect On Startup

You can download IDS from NeoTicker® customer area. Consult IDS's own documentation for further setup instructions.

Configuring Interactive Brokers as Real-Time Data Feed

This section is not applicable if you do not use Interactive Brokers as your real-time server.

If you want to configure NeoTicker® to place order to Interactive Brokers, see ***Configuring Order Placement to Interactive Brokers*** (on page 51).

Requirements

Interactive Brokers sends real-time data (including Level 2 data) to their customers. NeoTicker® can use this data for analysis purpose. So you need an IB account (or an IB paper trade account).

For historical data, you have several choices:

- Non-IB historical data such as TickQuest Backfill Server. See ***Setting Up Backfill*** (on page 541) for more information.
- IB historical data. This is a planned feature for 4.1. Please check our blog site for updated news.

Setup

Step 1: Installing and Configuring Interactive Brokers Traders Workstation Software.

You will need to install IB's Traders Workstation and related software. Follow the instructions in ***Tutorial: Installing Interactive Brokers Software*** (on page 243).

Step 2: Installing NT Order Server for Interactive Brokers

You can download NT Order Server for IB from one of the locations below:

Location 1: Full/lease version users can download NT Order Server in ***NeoTicker® Customer Area*** http://http://www.tickquest.com/NeoTicker/neotickercust/, under the section Download Order Server Plugins.

Location 2: All users can download NT Order Server in ***NeoTicker® Download Demo*** http://www.tickquest.com/NeoTicker/downloaddemo.html page, under the section Download Order Server Plugins.

After you download, run the program to install NT Order Server.

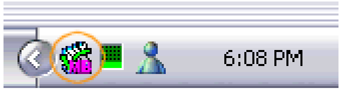
NT Order Server must reside in the OrderApp directory in NeoTicker® installation. If you installed NeoTicker® in a customized location, make sure NT Order Server is installed in the OrderApp directory.

Step 3: Configuring NeoTicker® to use Interactive Brokers as a Data Feed

1. Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.
2. Under **Datafeed** tab, choose **Other**, and select **Interactive Brokers Brokerage**.
3. Under **Backfill** tab, choose one of the backfill options. For the subscription option, fill in your LiveCharts user id and password. For explanation of various backfill options, see ***Setting Up Backfill*** (on page 541).
4. Press the **OK** button. Exit and restart NeoTicker®.

When NeoTicker® starts, it will ask you for order placement agreement (because NeoTicker® is connecting to a brokerage). You must understand and agree to this agreement before IB can be used as a data feed.

Then NeoTicker® will launch a program called NeoTicker® Order Server, which is automatically minimized in Windows' system tray in the lower right corner of your screen.



As well as launching IB's Trader Workstation Software (TWS). TWS is middleman between NeoTicker® and IB. So you will need to log in to TWS using your IB user id and password. Once logged in, you can simply minimize TWS.

When you run TWS for the very first time, NeoTicker® may not be able to connect to TWS. In this case, simply close down TWS and NeoTicker® and restart NeoTicker® again.

Once this is done, you are ready to use IB as a data feed.

Symbology

Interactive Brokers use a rather long symbol syntax. NeoTicker® simplifies the symbology by automatically mapping the much shorter NeoTicker® universal symbol format into Interactive Brokers' format. For more information, refer to ***Interactive Brokers Symbology*** (see "Interactive Brokers DDE (Obsolete)" on page 26).

Automating Order Placement Agreement

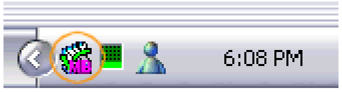
By default, when connecting to a real-life broker, NeoTicker® wants you to manually confirm the order placement agreement. You can configure auto confirmation by:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Connection** tab.
3. Check **Accept Order Placement Usage Agreement by Default** under **Options**.
4. Press **OK** button.

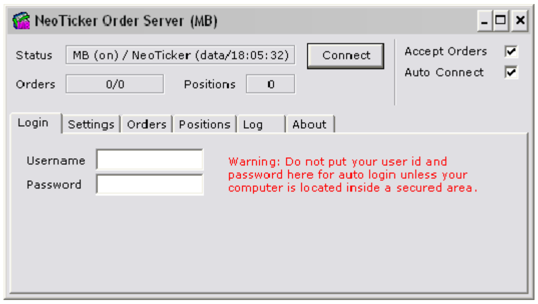
Automatic Login to TWS

If your computer is located in a safe area, you can set up NeoTicker® to auto login to TWS:

1. Double click on the NeoTicker® Order Server icon in the system tray to open NT Order Server.



2 In NT Order Server, unger **Login** tab, enter your IB user name and password.



Status MB (on) / NeoTicker (data/18:05:32)

Orders 0/0 Positions 0

Accept Orders ☑

Auto Connect ☑

Username

Password

Warning: Do not put your user id and password here for auto login unless your computer is located inside a secured area.

Other Server Settings

There are optional information you can enter to fine tune your data feed connection. For more information, refer to ***Server Setup Reference*** (on page 517).

Interactive Brokers Symbology

This section is now ***online*** (http://www.tickquest.com/ibsymbol.html).

Interactive Brokers DDE (Obsolete)

Interactive Brokers DDE is obsoleted. Its functionality has been superseded by the current implementation of IB connection. If you need reference, refer to manual of previous version of NeoTicker®.

Configuring IQFeed

This section is not applicable if you do not use IQFeed as your real-time server.

Requirements

To use IQFeed with NeoTicker® you need:

- Subscription to IQFeed Basic Service or higher.
- Download the IQFeed Client 4.1.1.1 from IQFeed, then install it. Do not install earlier versions of IQFeed Client.

As of NeoTicker® 4.1, older IQFeed client are no longer supported. You must install IQFeed Client 4.1.1.1 or higher.

Limitation

The current version of IQFeed does not provide second information in historical ticks. Thus it is not suitable for tick and second charts. If you require these features, please contact our support for alternative solution.

Downloading IQFeed Client

IQFeed Client does not come with NeoTicker®. You need to download it separately.

You can download IQFeed Client from the following locations:

- NeoTicker®'s Yahoo group file area, under ***DTN_IQFeed*** http://groups.yahoo.com/group/neoticker/files/DTN_IQFeed/.
- NeoTicker® ***customer area*** http://www.tickquest.com/NeoTicker/neotickercust/, under the section DTNIQ and IQFeed Users.
- On the NeoTicker® CD (version 3.2 or later), under the section IQFeed folder.

Setup

To set up IQFeed as the real-time data server:

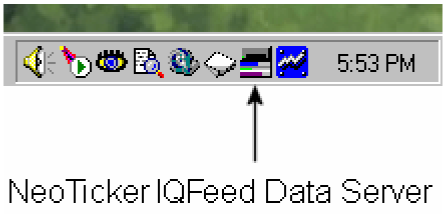
1. Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.
2. Under the **Datafeed** tab, and select **DTN IQ/DTN IQFeed**.
3. Press the **OK** button. Exit and restart NeoTicker®.

4 IQFeed will start automatically. You will need to enter your IQFeed user id and password to connect.

5 There are optional information you can enter to fine tune your data feed connection. For more information, refer to ***Server Setup Reference*** (on page 517).

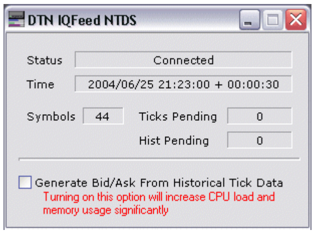
How NeoTicker® Connects to IQFeed (Optional Reading)

NeoTicker®'s IQFeed connections uses a new generation of data feed connection technology from TickQuest. When NeoTicker® decides to connect to IQFeed, NeoTicker® will launch a program called *NeoTicker IQFeed Data Server*. You can see this program in your icon tray:



Because NeoTicker IQFeed Data Server is a separate program, TickQuest can easily upgrade IQFeed users's connection without going through a full NeoTicker® release cycle. Also, any data feed related issues are isolated to the server program and will not affect NeoTicker®.

NeoTicker IQFeed Data Server is completely transparent to you. The only time you need to configure it is if your analysis requires historical bid/ask data (majority of analysis does not require historical bid/ask data). In which case, you can double click the icon and a window will open. Then you can check the **Generate Bid/Ask Ticks From Historical Tick Data** option.


DTN IQFeed NTDS

Status Connected

Time 2004/06/25 21:23:00 + 00:00:30

Symbols 44 Ticks Pending 0

Hist Pending 0

□ Generate Bid/Ask From Historical Tick Data

Turning on this option will increase CPU load and memory usage significantly

Configuring QuoteSpeed

This section is not applicable if you do not use QuoteSpeed as your real-time server.

Requirements

To use QuoteSpeed with NeoTicker® you need:

- Subscription to QuoteSpeed.
- Have QuoteSpeed Workstation 4.0.7 or higher installed on your computer.

Setup

To set up QuoteSpeed as the real-time data server:

1 Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.

2 Under the **Datafeed** tab, and select **QuoteSpeed**.

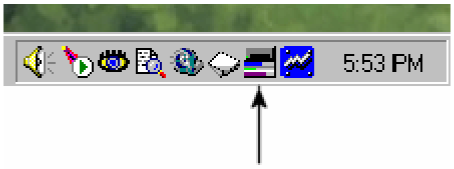
3 Press the **OK** button. Exit NeoTicker®.

4 Start QuoteSpeed Workstation. QuoteSpeed Workstation must be running before NeoTicker® starts.

5 Start NeoTicker®.

How NeoTicker® Connects to QuoteSpeed (Optional Reading)

NeoTicker®'s QuoteSpeed connections uses a new generation of data feed connection technology from TickQuest. When NeoTicker® decides to connect to QuoteSpeed NeoTicker® will launch a program called *NeoTicker QuoteSpeed Data Server*. You can see this program in your icon tray:



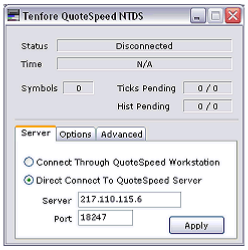
Because QuoteSpeed Data Server is a separate program, TickQuest can easily update QuoteSpeed support without going through a full NeoTicker® release cycle. Also, any data feed related issues are isolated to the server program and will not affect NeoTicker®.

Direct Connection Without Using QuoteSpeed Workstation

QuoteSpeed Data Server by default retrieves data from QuoteSpeed Workstation. This is why you need to start QuoteSpeed before running NeoTicker®. This is the preferred setup because QuoteSpeed Workstation offers some QuoteSpeed specific features.

It is possible to configure Data Server to connect to a QuoteSpeed Server directly, bypassing QuoteSpeed Workstation. A directly connection offers slight performance gain and saves you from starting QuoteSpeed Workstation before NeoTicker®.

1 In QuoteSpeed Data Server, under **Server** tab, choose **Direct Connect To QuoteSpeed Server**.



Tenfore QuoteSpeed NTDS

Status: Disconnected
Time: N/A
Symbols: 0
Ticks Pending: 0 / 0
Hist Pending: 0 / 0

○ Connect Through QuoteSpeed Workstation
● Direct Connect To QuoteSpeed Server
Server: 217.110.115.6
Port: 18247

2 Specify the IP address for a QuoteSpeed Server. You can look up the IP address from QuoteSpeed Workstation, under **Setup>Login/Configuration**.

3 Leave the **Port** number at 18247. Press **Apply** button.

4 In NeoTicker®, Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.

5 In Server Setup, under the **Login** tab, specify your QuoteSpeed login id and password.

6 Restart NeoTicker®.

Configuring RealTick

This section is not applicable if you do not use RealTick as your real-time server.

Requirements

To use RealTick with NeoTicker® you need:

- Subscription to RealTick
- Have RealTick 8 or higher installed on your computer.

Setup

To set up RealTick as the real-time data server:

1 Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.

2 Under the **Datafeed** tab, and select **RealTick**.

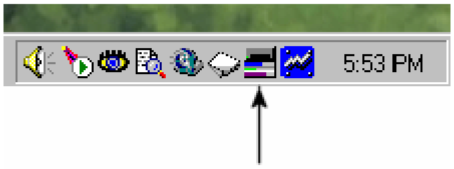
3 Press the **OK** button. Exit NeoTicker®.

4 Restart NeoTicker®.

5 RealTick will be started automatically. It will prompt you for your RealTick user id and password.

How NeoTicker® Connects to RealTick (Optional Reading)

NeoTicker®'s RealTick connections uses a new generation of data feed connection technology from TickQuest. When NeoTicker® decides to connect to RealTick, NeoTicker® will launch a program called *NeoTicker RealTick Data Server*. You can see this program in your icon tray:



Because RealTick Data Server is a separate program, TickQuest can easily update RealTick support without going through a full NeoTicker® release cycle. Also, any data feed related issues are isolated to the server program and will not affect NeoTicker®.

Configuring myTrack Connection

This section is not applicable if you do not use myTrack as your real-time server.

Requirements

To use myTrack with NeoTicker®, you need to enable the SDK option of your subscription with myTrack. There is additional charges for this option and you can find out more about this additional cost using Chat option in the myTrack program.

Setup

To set up myTrack as the real-time data server:

1. Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.
2. Choose **myTrack** under the **Datafeed** tab.
3. Enter your myTrack user id and password under the **Log in** tab, i.e. the user id and password supplied by myTrack.
4. Press the **OK** button. Exit and restart NeoTicker®.
5. There are optional information you can enter to fine tune your data feed connection. For more information, refer to ***Server Setup Reference*** (on page 517).

Forex Symbols

myTrack supports two format of Forex symbol. First format looks like USDCAD, USDJPY. Second format looks like CU.CD, CU.JY.

The first format is problematic for historical data. So NeoTicker® has these symbols blocked out. You should use the second format only.

Configuring MB Trading as Real-Time Data Feed

This section is not applicable if you do not use MB Trading as your real-time server.

If you want to configure NeoTicker® to place order through MB Trading, see ***Configuring Order Placement to MB Trading*** (on page 53).

Requirements

MB Trading http://www.mbtrading.com sends real-time data (including Level 2 data) to their customers. NeoTicker® can use this data for analysis purpose. So you need a MB Trading account.

However, MB Trading does not provide historical data. The solution is NeoTicker® can use historical data from another source - Quote.com's ***LiveCharts*** http://www.livecharts.com or TickQuest Backfill Server.

With LiveCharts. You can use the free delay data from LiveCharts. The free data is for demo purpose only, you will see a 15-minute data hole in the chart and from time to time, their server will be too busy to provide data to you.

The much better option is subscribe to LiveCharts. LiveCharts is reasonably priced and offers excellent data service.

Another alternative is to use free delay data hosted on our TickQuest Backfill Server. It is a reasonably fast service with limited symbol. It is suitable for research, simulation and some real-time use.

Setup

To set up MB Trading as the real-time data server:

Step 1: Installing and Configuring MBT Navigator

You will need to install MB Trading's MBT Navigator in order for NeoTicker® to receive data from MB Trading.

1. Go to ***MB Trading Website*** http://www.mbtrading.com.
2. In MB Trading's website, go to **Trading System>MBT Navigator>Download**.
3. Press the **Download** button to download MBT Navigator installation program to your hard drive.
4. Install MBT Navigator.

Step 2: Installing NT Order Server for MB Trading

You can download NT Order Server for MB from one of the locations below:

Location 1: Full/lease version users can download NT Order Server in ***NeoTicker® Customer Area*** http://http://www.tickquest.com/NeoTicker/neotickercust/, under the section Download Order Server Plugins.

Location 2: All users can download NT Order Server in ***NeoTicker® Download Demo*** http://www.tickquest.com/NeoTicker/downloaddemo.html page, under the section Download Order Server Plugins.

After you download, run the program to install NT Order Server.

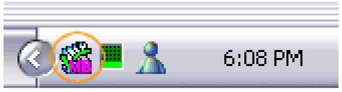
NT Order Server must reside in the OrderApp directory in NeoTicker® installation. If you installed NeoTicker® in a customized location, make sure NT Order Server is installed in the OrderApp directory.

Step 3: Configuring NeoTicker® to use MB Trading as a Data Feed

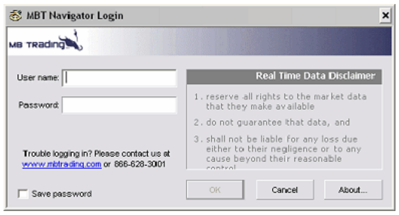
1. Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.
2. Under **Datafeed** tab, choose **Other**, and select **MB Trading Brokerage**.
3. Under **Backfill** tab, choose one of the backfill options. For the subscription option, fill in your LiveCharts user id and password. For explanation of various backfill options, see ***Setting Up Backfill*** (on page 541).
4. Press the **OK** button. Exit and restart NeoTicker®.

When NeoTicker® starts, it will ask you for order placement agreement (because NeoTicker® is connecting to a brokerage). You must understand and agree to this agreement before MB Trading can be used as a data feed.

Then NeoTicker® will launch a program called NeoTicker® Order Server, which is automatically minimized in Windows' system tray in the lower right corner of your screen.



As well as launching MB Navigator. In MB Navigator, you will need to fill in your MB Trading user name and password here.



1. reserve all rights to the market data that they make available
2. do not guarantee that data, and
3. shall not be liable for any loss due either to their negligence or to any cause beyond their reasonable control

Trouble logging in? Please contact us at www.mbtrading.com or 866-628-3001

Once this is done, you are ready to use MB Trading as a data feed.

Automating Order Placement Agreement

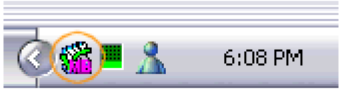
By default, when connecting to a real-life broker, NeoTicker® wants you to manually confirm the order placement agreement. You can configure auto confirmation by:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Connection** tab.
3. Check **Accept Order Placement Usage Agreement by Default** under **Options**.
4. Press **OK** button.

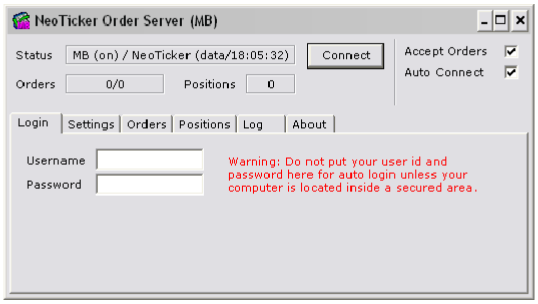
Automatic Login to MB Navigator

If your computer is located in a safe area, you can set up NeoTicker® to auto login to MB Trading:

1. Double click on the NeoTicker® Order Server icon in the system tray to open NT Order Server.



2. In NT Order Server, under **Login** tab, enter your MB Trading user name and password.



Status: MB (on) / NeoTicker (data/18:05:32)

Orders: 0/0 Positions: 0

Accept Orders ☑

Auto Connect ☑

Username

Password

Warning: Do not put your user id and password here for auto login unless your computer is located inside a secured area.

Other Server Settings

There are optional information you can enter to fine tune your data feed connection. For more information, refer to ***Server Setup Reference*** (on page 517).

Configuring EFX Group as Real-Time Data Feed

This section is not applicable if you do not use EFX Group as your real-time server.

If you want to configure NeoTicker® to place order through EFX Group, see ***Configuring Order Placement to EFX Group*** (on page 53).

Requirements

EFX Group (http://www.efxgroup.com) sends real-time data to their customers. NeoTicker® can use this data for analysis purpose. So you need an EFX Group account.

However, EFX Group does not provide historical data. The solution is NeoTicker® can use historical data from another source - Quote.com's ***LiveCharts*** http://www.livecharts.com or TickQuest Backfill Server.

With LiveCharts. You can use the free delay data from LiveCharts. The free data is for demo purpose only, you will see a 15-minute data hole in the chart and from time to time, their server will be too busy to provide data to you.

The much better option is subscribe to LiveCharts. LiveCharts is reasonably priced and offers excellent data service.

Another alternative is to use free delay data hosted on our TickQuest Backfill Server. It is a reasonably fast service with limited symbol. It is suitable for research, simulation and some real-time use.

Setup

To set up EFX Group as the real-time data server:

Step 1: Installing and Configuring EFX Navigator

You will need to install EFX Group's EFX Navigator in order for NeoTicker® to receive data from EFX Group.

1. Go to ***EFX Trading*** (http://www.efxgroup.com)'s website.
2. In MB Trading's website, go to Trading **System>Download**.
3. Press the **Download** button to download EFX Navigator installation program to your hard drive.
4. Install EFX Navigator.

Step 2: Installing NT Order Server for MB Trading

You can download NT Order Server for EFX from one of the locations below:

Location 1: Full/lease version users can download NT Order Server in ***NeoTicker® Customer Area*** http://http://www.tickquest.com/NeoTicker/neotickercust/, under the section Download Order Server Plugins.

Location 2: All users can download NT Order Server in ***NeoTicker® Download Demo*** http://www.tickquest.com/NeoTicker/downloaddemo.html page, under the section Download Order Server Plugins.

After you download, run the program to install NT Order Server.

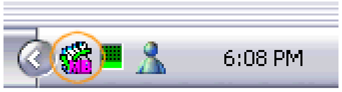
NT Order Server must reside in the OrderApp directory in NeoTicker® installation. If you installed NeoTicker® in a customized location, make sure NT Order Server is installed in the OrderApp directory.

Step 3: Configuring NeoTicker® to use MB Trading as a Data Feed

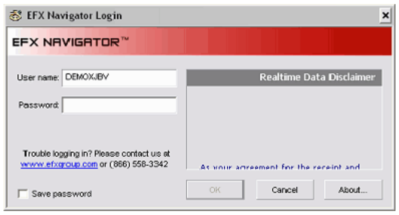
1. Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.
2. Under **Datafeed** tab, choose **Other**, and select **EFX Group Brokerage**.
3. Under **Backfill** tab, choose one of the backfill options. For the subscription option, fill in your LiveCharts user id and password. For explanation of various backfill options, see ***Setting Up Backfill*** (on page 541).
4. Press the **OK** button. Exit and restart NeoTicker®.

When NeoTicker® starts, it will ask you for order placement agreement (because NeoTicker® is connecting to a brokerage). You must understand and agree to this agreement before EFX Group can be used as a data feed.

Then NeoTicker® will launch a program called NeoTicker® Order Server, which is automatically minimized in Windows' system tray in the lower right corner of your screen.



As well as launching EFX Navigator. In EFX Navigator, you will need to fill in your EFX Group user name and password here.



User name: DEMOXJBV

Password:

Trouble logging in? Please contact us at www.efxgroup.com or (866) 558-3342

As your agreement for the receipt and

Once this is done, you are ready to use EFX Group as a data feed.

Automating Order Placement Agreement

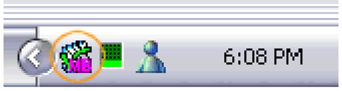
By default, when connecting to a real-life broker, NeoTicker® wants you to manually confirm the order placement agreement. You can configure auto confirmation by:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Connection** tab.
3. Check **Accept Order Placement Usage Agreement by Default** under **Options**.
4. Press **OK** button.

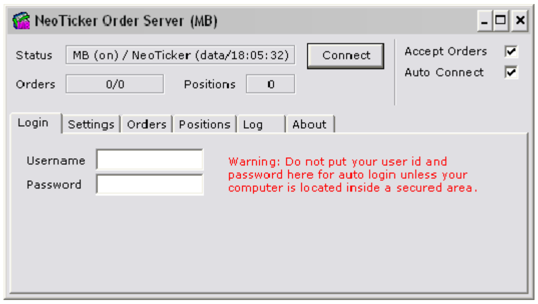
Automatic Login to EFX Navigator

If your computer is located in a safe area, you can set up NeoTicker® to auto login to EFX Group:

1. Double click on the NeoTicker® Order Server icon in the system tray to open NT Order Server.



2. In NT Order Server, under **Login** tab, enter your EFX Group user name and password.



NeoTicker Order Server (MB)

Status MB (on) / NeoTicker (data/18:05:32)

Orders 0/0 Positions 0

Accept Orders ☑

Auto Connect ☑

Username

Password

Warning: Do not put your user id and password here for auto login unless your computer is located inside a secured area.

Other Server Settings

There are optional information you can enter to fine tune your data feed connection. For more information, refer to ***Server Setup Reference*** (on page 517).

Configuring QCharts (Quote.com) Connection

This section is not applicable if you do not use QCharts/Quote.com as your real-time server.

Requirements

You need to subscribe to Quote.com's QCharts or QDP service.

Important to Windows NT, Windows 2000 and Windows XP Professional Users: we recommend the user to have administrator privilege. Otherwise, login time to QCharts will be significantly slower.

Setup

To set up QCharts as the real-time data server:

1. Open the Server Setup dialog by choosing **Program>Server Setup** in the main window.
2. Choose **QCharts (Quote.com)** under the **Datafeed** tab.
3. Enter your QCharts user id and password under the **Log in** tab.
4. Press the **OK** button. Exit and restart NeoTicker®.
5. There are optional information you can enter to fine tune your data feed connection. For more information, refer to ***Server Setup Reference*** (on page 517).

Server Connection

When NeoTicker® makes a connection to QCharts, NeoTicker® makes a guess on the best server available. If a connection attempt failed, NeoTicker® will rotate to the next best server.

If a connection takes too long, you can force NeoTicker® to go into off-line mode by holding the SHIFT key on the keyboard.

Server Rotation

You can use the CTRL-ALT-N key sequence to rotate to the next Quote.com server.

Considerations for Experienced Quote.com Users

ContinuumClient.ini

The `ContinuumClient.ini` file is located in the NeoTicker®'s installation directory under the folder `config` (e.g. `C:\Program Files\TickQuest\NeoTicker 4\config`). This file lists all the servers that NeoTicker® will try to access. If you intended to constrain which servers NeoTicker® should try to work with, then you can adjust this list accordingly.

If you intended to enforce the specific order on which server NeoTicker® will try to connect to QCharts, then change the line `AutoUpdateServerList=true` to `AutoUpdateServerList=false`.

If you have special Internet settings that needs to be used for connecting to QCharts, you can add those special lines to this `ContinuumClient.ini` file and NeoTicker® will utilize those settings.

Best Server List

NeoTicker®'s best server list is generated when NeoTicker® starts. You can make NeoTicker® to regenerate the best server list without exiting the program.

First, delete the `ContinuumClient.ini` file located in the NeoTicker® installation directory.

IMPORTANT: do not delete the `ContinuumClient.ini` file in the `config` folder.

Then disconnect and reconnect to the server, or press CTRL-ALT-N, to tell NeoTicker® to rotate server. The best server list will be regenerated.

Configuring a DDE Data Feed

DDE feed should only be used if you have no other choice regarding to real-time data service. DDE feed does not have the quality of a regular data feed.

NeoTicker® can receive data from a DDE data feed. Many data feeds can send data to Microsoft Excel by a DDE link. NeoTicker® can read DDE data the same way Excel does, allowing you to connect to a data feed that is not directly supported by NeoTicker®.

What you need to know

You need to find out the server, topic and item of the data feed. Because every data feed configure DDE link differently, you should have a general idea how data is represented in the data feed.

Example: 101Quote DDE

The 101Quote DDE feed is part of NeoTicker® installation. You can simply choose the feed and use it. This example serves as an illustration on how to configure a DDE feed that has a format similar to 101Quote.

101Quote is TickQuest's simple quote window that pulls quotes from popular Internet websites. 101Quote can act as an DDE server to NeoTicker®.

101Quote uses the following DDE format:

```
TQQ|MSFT!LAST
```

This is how you specify a DDE feed for 101Quote.

1 Open the DDE Data Feeds Manager by choosing **Manager>DDE Data Feeds** from the main window.

2 Enter a name into the **Name** field. This is a name you choose, e.g. 101QUOTE

3 Press the **Server/Topic** tab.

4 In the **Server Name** field, enter `TQQ`.

5 For **Stream Topic**, choose **Symbol**.

6 Press the **Item Rules** tab.

7 Create a rule called `Last Price`, by setting **Request Msg** to `<SYMBOL>,LAST`, **Receive Msg** to `LAST`, then press the **Add/Replace** button.

8 Additional rules can be specified for each visible column in 101Quote.

Example: Bloomberg DDE

When Bloomberg sends data to Excel, it uses the following format:

```
BLP|M!'CSCO Equity,[LAST_PRICE]'
```

where `CSCO Equity` is the symbol.

Thus the server is BLP; topic is M; item is symbol followed by the field, enclosed in square bracket.

This is how you specify a DDE feed for Bloomberg.

1 Open the DDE Data Feeds Manager by choosing **Manager>DDE Data Feeds** from the main window.

2 Enter a name into the **Name** field. This is a name you choose, e.g. `Bloomberg DDE`.

3 Press the **Server/Topic** tab.

4 In the **Server Name** field, enter `BLP`.

5 For **Stream Topic**, choose **Predefined**, and enter `M`.

6 Press the **Item Rules** tab.

7 Create a rule called `Last Price`, by setting **Request Msg** to `<SYMBOL>,[LAST_PRICE]`, **Receive Msg** to `LAST`, then press the **Add/Replace** button.

You should use Bloomberg's DDE Monitor to check for symbol format and other fields that are available. Additional fields can be specified by adding more rules, or by using comma to separate different fields, e.g. by setting **Request Msg** to `<SYMBOL>,[LAST_PRICE,BID,ASK,VOLUME]`, **Receive Msg** to `LAST,BID,ASK,VOLUME`.

Example: Reuters DDE

When Reuters sends data to Excel, it uses the following format:

```
REUTER|IDN!'BISC.MI,LAST'
```

Where BISC.MI is the symbol

Thus the server is REUTER; topic is IDN; items is symbol followed by the field.

This is how you specify a DDE feed for Reuters:

1 Open DDE Data Feeds Manager by choosing **Manager>DDE Data Fees** from the main window.

2 Enter a name into the **Name** field. This is a name you choose, e.g. Reuters DDE.

3 Press the **Server/Topic** tab.

4 In the **Server Name** field, enter `REUTER`.

5 For **Stream Topic**, choose **Predefined** and enter `IDN`.

6 Press the **Item Rules** tab.

7 Create a rule called `Last`, by setting **Request Msg** to `<SYMBOL>,LAST` , **Receive Msg** to `LAST` , then press the **Add/Replace** button.

You should use Reuters Terminal Windows (RTW) to check for symbol format and other fields that are available. Additional fields can be specified by adding more rules, or by using comma to separate different fields and enclose them with square brackets, e.g. by setting **Request Msg** to `<SYMBOL>,[LAST,ASK,BID,TURNOVER]` , **Receive Msg** to `LAST,ASK,BID,TURNOVER`.

Example: eSignal DDE

While NeoTicker® supports a direct eSignal feed, eSignal's DDE feed illustrates another type of DDE format.

eSignal's Data Manager sends out DDE link in the following format:

```
WINROS|LAST!MSFT
```

Note that unlike most DDE feeds, eSignal specifies the field in the DDE topic, and symbol in the DDE item. To specify eSignal DDE feed:

1 Open the DDE Data Feeds Manager by choosing **Manager>DDE Data Feeds** from the main window.

2 Enter a name into the **Name** field. This is a name you choose, e.g. `ESIGNAL`.

3 Press the **Server/Topic** tab.

4 In the **Server Name** field, enter `WINROS`.

5 For **Stream Topic**, choose **Item as Topic, Symbol as Item**.

6 Press the **Item Rules** tab.

7 Create a rule called `Last Price`, by setting **Request Msg** to `<SYMBOL>,LAST_PRICE`, **Receive Msg** to `LAST`, then press the **Add/Replace** button.

To add additional rules, consult eSignal Data Manager's help manual for additional fields.

Selecting a DDE Feed

To select a DDE feed:

1 Choose **Program>Server Setup** from the main window.

2 Choose **Other** as the data source.

3 Select the desired DDE feed from the drop down.

4 Press the **OK** button and restart NeoTicker®.

DDE Feed Considerations

Historical Data

DDE feeds do not provide historical data. Data is accumulated into the disk cache as ticks come in. Therefore you cannot chart until sufficient data is available.

Monitoring Symbols

Once you've requested a symbol, NeoTicker® will continue to receive data until NeoTicker® or the DDE server quits.

For example, if you quote MSFT in a quote window, and later delete the quote window. Data for MSFT will still be sent to NeoTicker®. So if you chart MSFT later, the data will still be continuous.

Also, the symbol will be added to the pre-load symbol list of RAM Cache. What this means is when you restart NeoTicker®, the said symbol will be automatically requested from your DDE server.

DDE Updates

Most DDE feeds do not provide tick-by-tick update. Instead, data is sent to the client application (e.g. NeoTicker® on timer).

101Quote is a good example. It sends data to NeoTicker® every 10 seconds.

This has several consequences.

- First, the bars constructed are approximation. The longer the time the bar is, the more accurate the approximation. In 101Quote's example, 1-minute bar is less accurate than 5-minute bar.
- Second, it does not make sense to use tick chart because the data are not true ticks.
- Third, trade volume is undefined. Trade volume is the accumulation of volume each trade. Since the trades are undefined, trade volume is undefined.

Locale and Time Zone Considerations

By default, DDE feed uses your PC's time zone. However, some DDE feed sends out data in US decimal format and time even if your computer is set to a different locale (e.g. a computer in Germany receives DDE data in New York time, using a period for decimal). In this case, you can specify DDE Data Feeds Manager to use the US locale without having to set your computer's time zone to New York time.

1 In DDE Data Feeds Manager, press **Advance** tab.

2 Change **Time Stamp** and **Numeric Values** to the locale of your data.

CHAPTER 7

Connecting to Your Broker

Disclaimer

Past performance is not a guarantee of future results. Only risk capital should be used to trade futures, stocks, options on futures or stocks, mutual funds or any other type of financial instruments. Whether this product is used in conjunction with futures, stocks, stock indices or mutual funds, all involve a high degree of inherent financial risk and the possibility of loss is great. TickQuest Inc. does not assume any responsibilities, make any guarantees whatsoever, or make any trading recommendations in NeoTicker®. All such investment vehicles carry risks and all trading decisions are ultimately made by you. You are solely and individually responsible for those decisions and the results of those decisions.

When you connect to a real-life broker, all orders are sent to the broker for real-life execution. **You need an account at the broker and real money is involved**. We strongly recommend our users to first practice trading with Trade Simulator, especially for orders generated by trading systems, before routing the orders to a real broker.

Introduction

You can place orders to your broker directly from NeoTicker®.

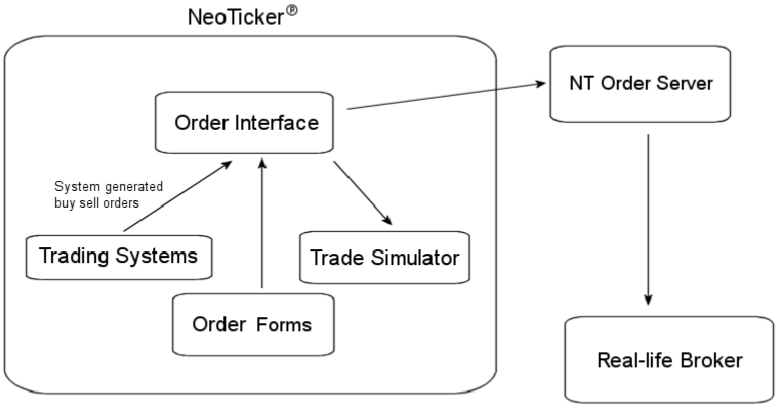
If you use a mechanical trading system (computer programs that trade) from within NeoTicker®, the trading system can also directly place orders to your broker.

Currently the following brokers are supported:

- MB Trading
- Interactive Brokers
- Brokers supported by NinjaTrader. In this case, NinjaTrader is used as the middle-ware to connect to your broker.

NeoTicker® uses an extensible architecture so it is possible to support additional brokers. If you are interested in having a broker supported, please contact our sales team.

How It Works



When you place an order in NeoTicker®, a module called Order Interface is responsible for handling and routing the order.

So how does Order Interface send the order to your broker? It doesn't. Because every broker works differently, Order Interface sends the order to a plug-in called NT Order Server. NT Order Server will in turn send the order to your broker.

You can think of NeoTicker® and your broker speak different languages. They cannot directly communicate and NT Order Server acts an interpreter between the two. For this reason, you will need a different NT Order Server for each broker.

Once configured, NT Order Server will be largely invisible to you.

The rest of this chapter tells you how to:

- Installing NT Order Server for your broker
- Configuring Order Interace to talk to your broker through NT Order Server

For an in depth discussion of how orders work in NeoTicker®, refer to the section ***How All Order Related Items Work Together*** (on page 695).

In This Chapter

Configuring Order Placement to Interactive Brokers

This section is online. Go to ***Online Tutorial: Configuration Order Placement to Interactive Brokers*** http://www.tickquest.com/NeoTicker/brokerIBconfig.html.

Configuring Order Placement to MB Trading

This section is online. Go to ***Online Tutorial: Configuring Order Placement to MB Trading*** http://www.tickquest.com/NeoTicker/brokerMBconfig.html.

Configuring Order Placement to EFX Group

This section is online. Go to ***Online Tutorial: Configuring Order Placement to EFX Group*** (http://www.tickquest.com/NeoTicker/brokerEFXconfig.html).

Configuring Order Placement to NinjaTrader

This section is online. Go to ***Online Tutorial: Configuring Order Placement to NinjaTrader*** http://www.tickquest.com/NeoTicker/brokerNTconfig.html.

Connection Related Options

See ***Order Interface, Connection Related Options*** (see "Connection Related Options" on page 752).

Troubleshooting Broker Connections

This section is online. Go to ***Online Tutorial: Troubleshooting Broker Connections*** http://www.tickquest.com/NeoTicker/brokertroubleshoot.html

Configuring Order Placement to Other Brokers

If you want NeoTicker® to place orders to your broker, it is possible to develop an order placement plug-in, provide that your broker also has a programming interface. The development can be done by TickQuest, your broker or a third party. For more information, contact our sales team.

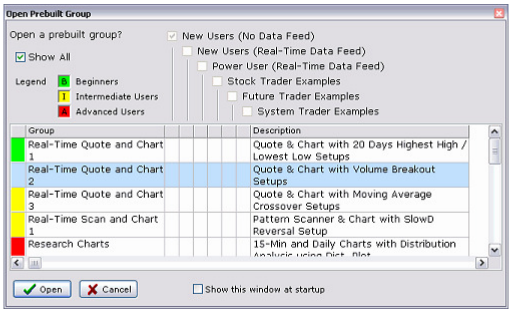
C H A P T E R 8

Tour of Prebuilt Groups

If you would like to see what NeoTicker® can do without first browsing through this manual, you can open the prebuilt groups to see a demonstration of some of the features that NeoTicker® offers.

To open a prebuilt group, choose from the main program window, **Group>Open Prebuilt Group**. Click on the name of the group and press the **Open** button.

Some of the group requires real-time data subscription. See ***Choosing and Configuring a Real-time Data Feed*** (on page 15).



	Group	Description
🟩	Real-Time Quote and Chart 1	Quote & Chart with 20 Days Highest High / Lowest Low Setups
	Real-Time Quote and Chart 2	Quote & Chart with Volume Breakout Setups
🟨	Real-Time Quote and Chart 3	Quote & Chart with Moving Average Crossover Setups
🟨	Real-Time Scan and Chart 1	Pattern Scanner & Chart with SlowD Reversal Setup
🟥	Research Charts	15-Min and Daily Charts with Distribution Analysis using Dist. Plot

Prebuilt groups are organized according to their usage. You can toggle the check boxes to see only the groups that interest you.

Usage Type	Description
New Users (No Data Feed)	For new users who do not yet have a real-time data feed.
New Users (Real-Time Data Feed)	For new users. A real-time data feed is required.
Power Users (Real-Time Data Feed)	For power users. A real-time data feed is required.
Stock Trader Examples	For stock traders.
Future Trader Examples	For futures traders.
System Trader Examples	Examples with trading systems (mechanical models of trading).

In addition, color legend is used to the level of NeoTicker® knowledge required to understand the group. Green indicates the group is suitable for new traders, yellow for intermediate traders, red for advanced traders.

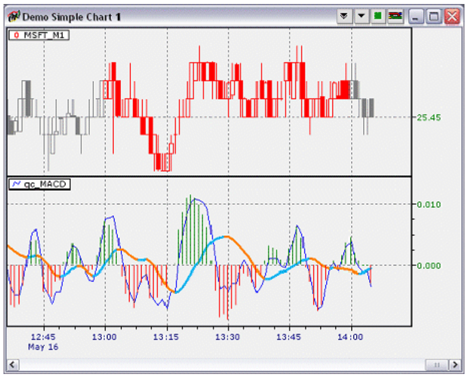
The rest this chapter describes the prebuilt groups in more details.

In This Chapter

Chart 1

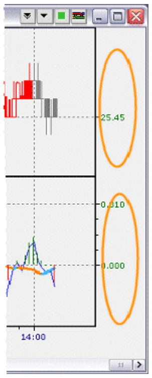
Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

Chart 1 demonstrates basic chart customization.

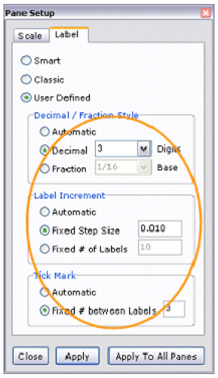


Customized Pane Labelling

Check them out by double click at the Price Axis area to bring up the Pane Setup Window.



Under the Label tab, you will find that both panes are set to **User Defined** with customized options chosen.



Pane Setup

Scale | Label

○ Smart
○ Classic
● User Defined

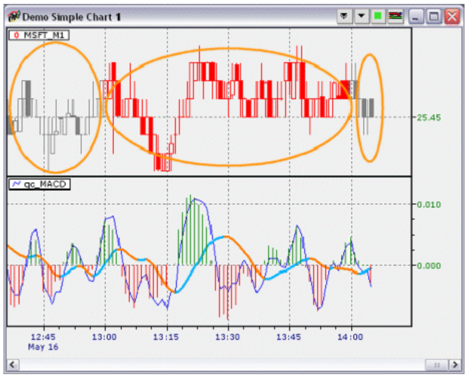
Decimal / Fraction Style
○ Automatic
● Decimal 3 Digits
○ Fraction 1/16 Base

Label Increment
○ Automatic
● Fixed Step Size 0.010
○ Fixed # of Labels 10

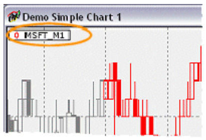
Tick Mark
○ Automatic
● Fixed # between Labels 3

Data Series Formula Coloring

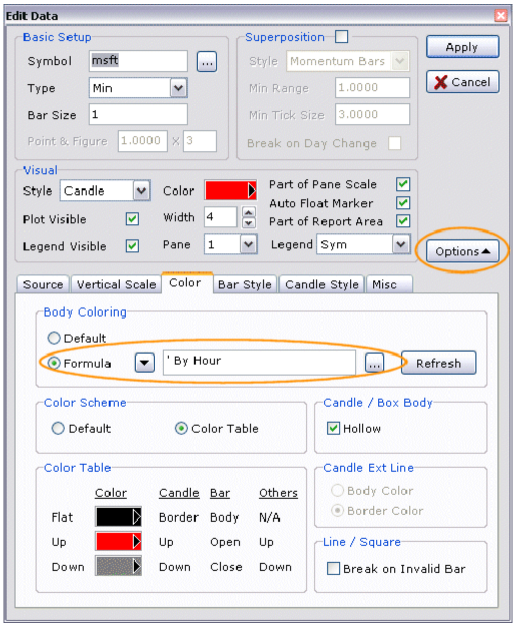
Data series in this example is colored by the hour.



To try out other color formula, double-click on the data legend to bring up the Edit Data Setup window.



Then click the **Options** button to show the extended options. Under **Color** tab, you will find the **Body Coloring** option set to use **Formula** and the **By Hour** formula is used.



Edit Data

Basic Setup
Symbol: msft
Type: Min
Bar Size: 1
Point & Figure: 1.0000 × 3

Superposition ☐
Style: Momentum Bars
Min Range: 1.0000
Min Tick Size: 3.0000
Break on Day Change ☐

Visual
Style: Candle
Color: 🟥
Width: 4
Pane: 1
Legend: Sym
Plot Visible ☑
Legend Visible ☑
Part of Pane Scale ☑
Auto Float Marker ☑
Part of Report Area ☑

Body Coloring
○ Default
● Formula: ' By Hour

Color Scheme
○ Default
● Color Table

Candle / Box Body
☑ Hollow

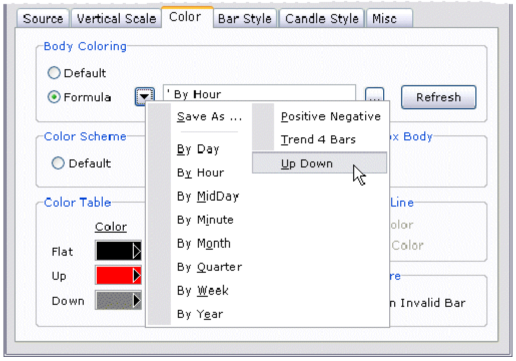
Color Table

	Color	Candle	Bar	Others
Flat	■	Border	Body	N/A
Up	🟥	Up	Open	Up
Down	▨	Down	Close	Down

Candle Ext Line
○ Body Color
● Border Color

Line / Square
☐ Break on Invalid Bar

To switch to any pre-defined formula, click on the drop down button to choose from the available choices.

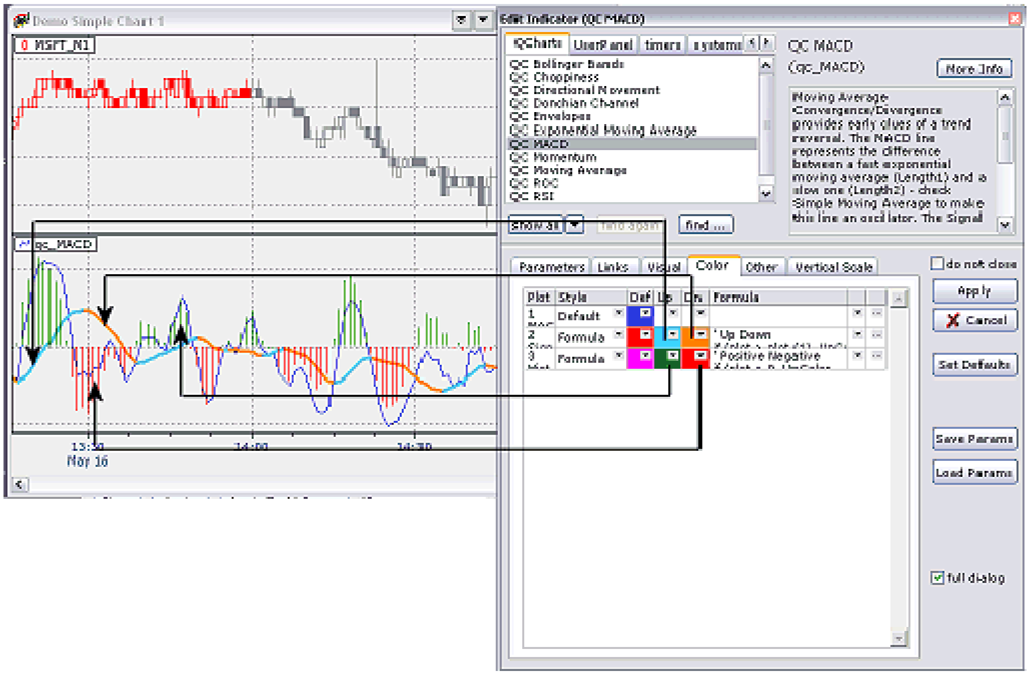


Body Coloring
○ Default
● Formula: ' By Hour

Indicator Formula Coloring

The qc_MACD indicator is using 2 different color formulas on 2 of its 3 plots.

To see how it works, you can open the Edit Indicator window by double clicking the indicator legend. Look for the **Color** tab, you will see that plot 2 is set to use the **Up Down** color formula, while plot 3 is set to use **Positive Negative** color formula.



Moving Average Convergence/Divergence provides early clues of a trend reversal. The MACD line represents the difference between a fast exponential moving average (Length1) and a slow one (Length2) - check Simple Moving Average to make this line an oscillator. The Signal

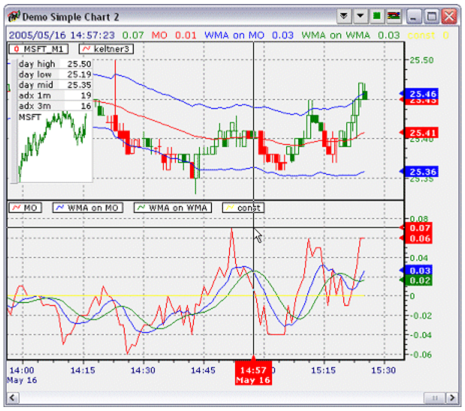
Plot	Style	Def	Up	Dn	Formula
1	Default	🟦			
2	Formula	🟥	🟦	🟧	'Up Down
3	Formula	🟪	🟩	🟥	'Positive Negative

To try out other color formulas, just choose from the drop down menu by clicking on the drop down icon button at the right side of the formula column.

Chart 2

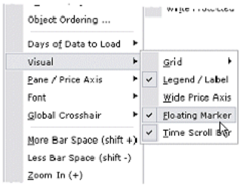
Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

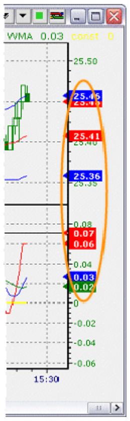
Chart 2 demonstrates time chart features such as float markers, indicator pane scale settings, indicator-on-indicator and Dynamic Grid.



Float Markers

You can enable float markers by right-clicking on the chart to open popup menu, then choose **Visual>Float Marker**.

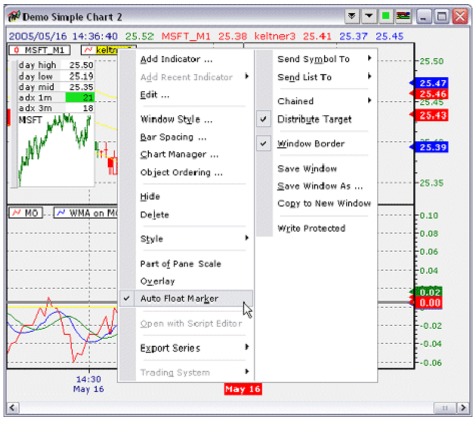




Individual Float Marker Settings

Float marker settings can be turn on/off for each individual data series or indicator.

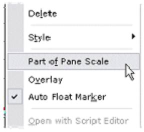
Right click on the data or indicator series itself, or its legend to open the popup menu, then choose **Auto Float Marker** item.



Part of Pane Scale Setting

The Keltner Channel indicator has Part of Pane Scale disabled. This means when the pane's price axis is scaling, the Kelter Channel values are not taken into account.

To enable/disable part of pane scale setting, right click on the indicator to open popup menu, then choose Part of Pane Scale.

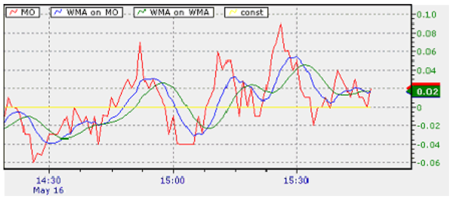


Indicator-on-Indicator

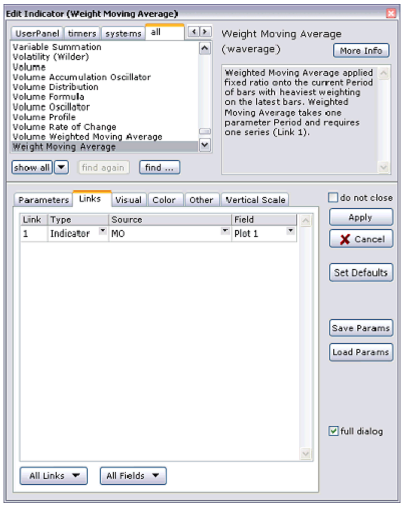
In the second pane of the chart, the Momentum indicator calculates the momentum of the data series.

The first weighted average (WMA on MO) is linked to the momentum, i.e. it calculates the moving average of the momentum indicator.

Second weighted average (WMA on WMA) is linked to WMA on MO, i.e. it calculates the moving average of the moving average.



To adjust indicator links, double click on the indicator or its legend to open the Edit Indicator window, the settings is under the **Links** tab.


Edit Indicator (Weight Moving Average)

Weight Moving Average
(waverage)

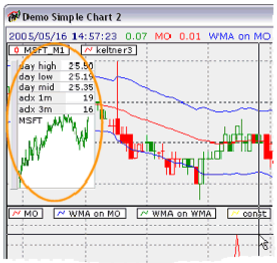
Weighted Moving Average applied fixed ratio onto the current Period of bars with heaviest weighting on the latest bars. Weighted Moving Average takes one parameter Period and requires one series (Link 1).

Link	Type	Source	Field
1	Indicator	MO	Plot 1

□ do not close

☑ full dialog

Dynamic Grid

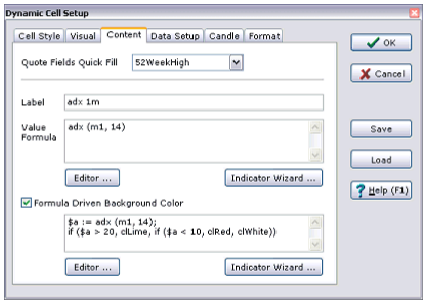

Demo Simple Chart 2

2005/05/16 14:57:23 0.07 MO 0.01 WMA on MO

day high	25.50
day low	25.19
day mid	25.35
adx 1m	19
adx 3m	16

Each cell has a different formula using combination of indicators and various time frames. The cells "adx 1m" and "adx 3m" utilize background coloring that highlight the cell in different colors based on the current value of the adx indicators.

To see how those formulas look like, double click on the cells to open the Dynamic Cell Setup window, then switch to the **Content** tab.



Quote Fields Quick Fill: 52WeekHigh

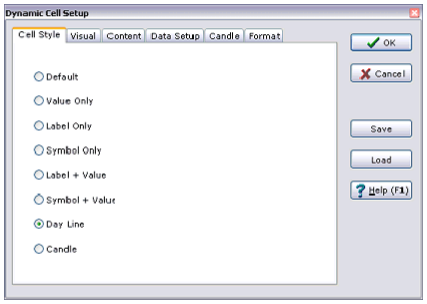
Label: adx 1m

Value Formula: adx (m1, 14)

☑ Formula Driven Background Color

```
$a := adx (m1, 14);
if ($a > 20, clLime, if ($a < 10, clRed, clWhite))
```

The last cell in the dynamic quote window display the "Day Line" graph, a useful tool for summarizing the whole trading day. To set a day line graph, double click on a cell to open the Dynamic Cell Setup window, then switch to the **Cell Style** tab, change the setting to **Day Line**.

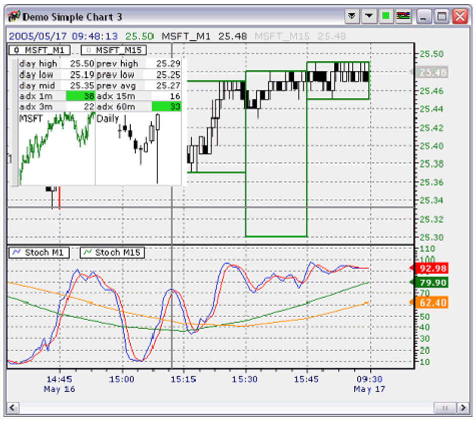


○ Default
○ Value Only
○ Label Only
○ Symbol Only
○ Label + Value
○ Symbol + Value
● Day Line
○ Candle

Chart 3

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

Chart 3 demonstrates Dynamic Grid and multiple time frame.

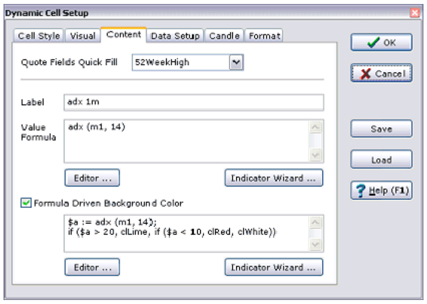


day high	25.50	prev high	25.29
day low	25.19	prev low	25.25
day mid	25.35	prev avg	25.27
adx 1m	38	adx 15m	16
adx 3m	22	adx 60m	33

Dynamic Grid Usage

Each cell has a different formula using combination of indicators and various time frames. The cells "adx 1m", "adx 3m", "adx 15m", and "adx 60m" utilize background coloring that highlight the cell in different colors based on the current value of the adx indicators.

If you double click on the "adx 1m" and "adx 15m" cells to open Dynamic Cell Setup window, and compare the formula under **Content** tab.



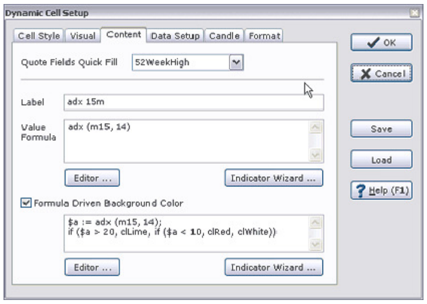
Quote Fields Quick Fill: 52WeekHigh

Label: adx 1m

Value Formula: adx (m1, 14)

☑ Formula Driven Background Color

```
$a := adx (m1, 14);
if ($a > 20, clLime, if ($a < 10, clRed, clWhite))
```



Quote Fields Quick Fill: 52WeekHigh

Label: adx 15m

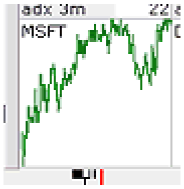
Value Formula: adx (m15, 14)

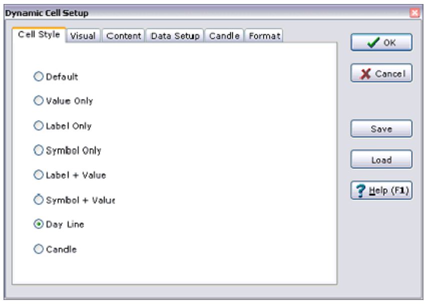
☑ Formula Driven Background Color

```
$a := adx (m15, 14);
if ($a > 20, clLime, if ($a < 10, clRed, clWhite))
```

You can see that the formulas for the two cells are almost identical, except they work on different time frames (1 minute vs. 15 minutes).

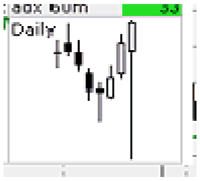
The lower left hand corner in the dynamic quote window display the "Day Line" graph, a useful tool for summarizing the whole trading day. To set a day line graph, double click on a cell to open the Dynamic Cell Setup window, then switch to the **Cell Style** tab, change the setting to **Day Line**.



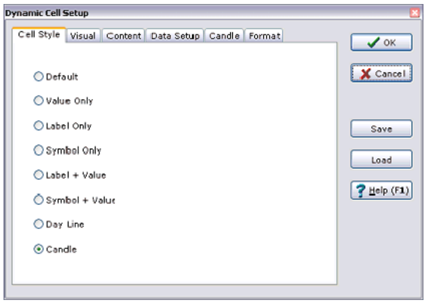


○ Default
○ Value Only
○ Label Only
○ Symbol Only
○ Label + Value
○ Symbol + Value
● Day Line
○ Candle

At the lower right hand corner a candlestick chart based on daily data is displayed. This will remind the trader visually the daily price levels of the past few days.

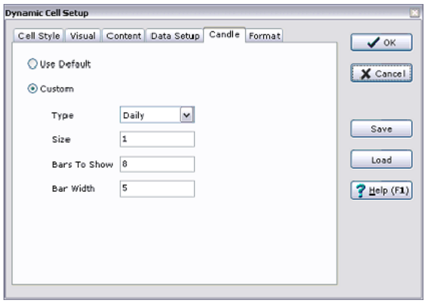


To set up candlestick chart in Dynamic Grid, double click on the cell to open Dynamic Cell Setup, under **Cell Style** tab, change the setting to **Candle**.



○ Default
○ Value Only
○ Label Only
○ Symbol Only
○ Label + Value
○ Symbol + Value
○ Day Line
● Candle

Then under the **Candle** tab, setup the type of candlesticks you want.



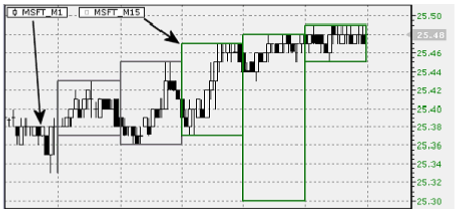
○ Use Default
● Custom

Type: Daily
Size: 1
Bars To Show: 8
Bar Width: 5

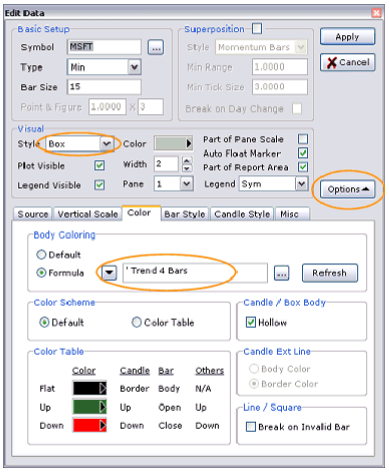
You may also want to check under the **Data Setup** tab for setting up trading time.

Overlay of Multiple Time Frames

The top pane is showing both 1-min bar in candlestick format and 15-min bar in box style.



The 15-min boxes are colored using the color formula 4 Bar Trend. You can edit the box style and coloring formula by double clicking on the MSFT_M15 legend to open Edit Data window. In the window, box style is set under the **Visual** group, **Style** item. To set coloring options, press the **Option** button to expand the window, and set the coloring options under **Color** tab.


Edit Data

Basic Setup
Symbol: MSFT
Type: Min
Bar Size: 15
Point & Figure: 1.0000 X 3

Superposition □
Style: Momentum Bars
Min Range: 1.0000
Min Tick Size: 3.0000
Break on Day Change □

Visual
Style: Box
Color
Plot Visible ☑
Legend Visible ☑
Width: 2
Pane: 1
Part of Pane Scale □
Auto Float Marker ☑
Part of Report Area ☑
Legend: Sym

Body Coloring
○ Default
● Formula: ' Trend 4 Bars

Color Scheme
● Default   ○ Color Table

Candle / Box Body
☑ Hollow

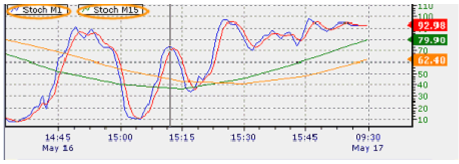
Color Table

	Color	Candle	Bar	Others
Flat	■	Border	Body	N/A
Up	🟩	Up	Open	Up
Down	🟥	Down	Close	Down

Candle Ext Line
○ Body Color
● Border Color

Line / Square
□ Break on Invalid Bar

The second pane is showing the corresponding stochastics indicators linked to the 1-min and 15-min data, i.e. the stochastics is calculated using 1-minute and 15-minute data respectively.

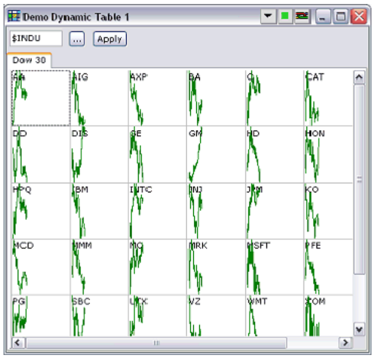


Parameters for stochastics and their links can be set by double clicking on the indicator legend to open Edit Indicator window. The values are set under the **Parameters** tab and **Links** tab.

Dynamic Table 1

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

Dynamic Table 1 demonstrates using Dynamic Table to display Day Line charts. Because each chart requires loading historical data from data vendor, this group can take up to several minutes to populate. The exact speed depends on the speed of your data vendor.

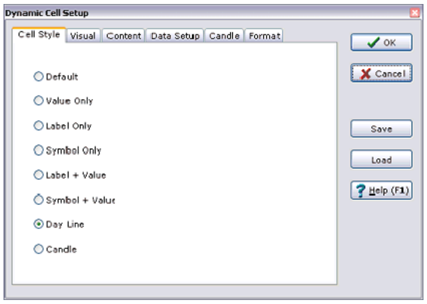


Demo Dynamic Table 1

$INDU

Dow 30 component symbols are shown in Day Line style over each cell. Day Line is the price movement of a stock in a trading day. The figure is taken at 11:00am ET. So only about one third of the cell is filled.

To edit a cell, double click the Dynamic Cell Setup window. Day Line is shown by setting the **Cell Style** to **Day Line**.

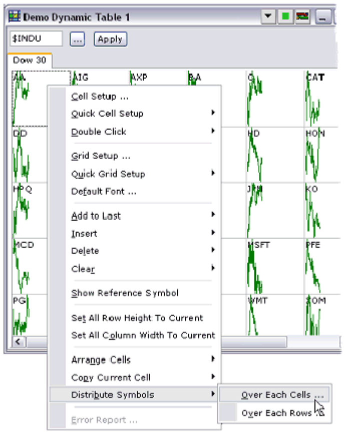


Dynamic Cell Setup

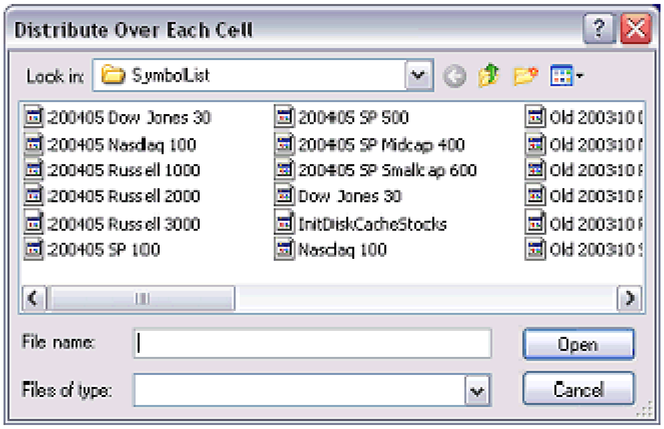
○ Default
○ Value Only
○ Label Only
○ Symbol Only
○ Label + Value
○ Symbol + Valuє
● Day Line
○ Candle

To make a cell displaying the symbol name, under the **Content** tab,simply clear the **Label** field. Symbol name is the default display for Day Line.

To replace the cells with a different set of symbols, you can right-click in the dynamic table to bring up its popup menu. Choose **Distribute Symbols>Over Each Cells**.



A dialog will display all the pre-defined symbol list that you can use.

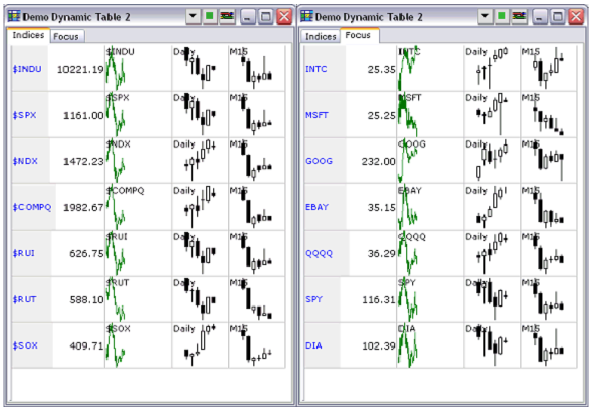


If you have other function windows (e.g. time chart) within the same group as this dynamic table, then you can double click a cell to send the reference symbol of the cell to the other function windows.

Dynamic Table 2

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

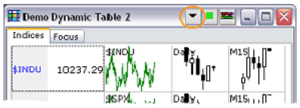
Dynamic Table 2 demonstrates multiple tab sheets in a Dynamic Table. The charts in Dynamic Tables 2 require loading historical data from data vendor. This group can take up to several minutes to populate. The exact speed depends on the speed of your data vendor.



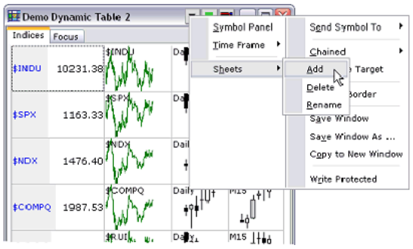
$INDU	10221.19
$SPX	1161.00
$NDX	1472.23
$COMPQ	1982.67
$RUI	626.75
$RUT	588.10
$SOX	409.71

INTC	25.35
MSFT	25.25
GOOG	232.00
EBAY	35.15
QQQQ	36.29
SPY	116.31
DIA	102.39

In Dynamic Table, you can add as many sheets as you want through the right-click popup menu **Sheets>Add**. Note that when you right click on a cell, you are bringing up the popup menu of the cell. To bring up the popup menu of Dynamic Table, you need to right click in an empty area of the window. So it may be easier to left click on the pop up menu button to bring up the menu instead of using right click.

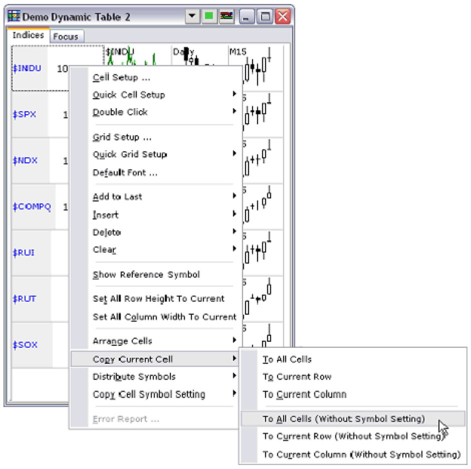


$INDU	10237.29



$INDU	10231.38
$SPX	1163.33
$NDX	1476.40
$COMPQ	1987.53

To reuse your formulas, visual settings, etc. you can use the various grid edit command to copy settings to other cells easily. To reuse settings of a cell, right click on the cell, and choose one of the item under **Copy Current Cell**.

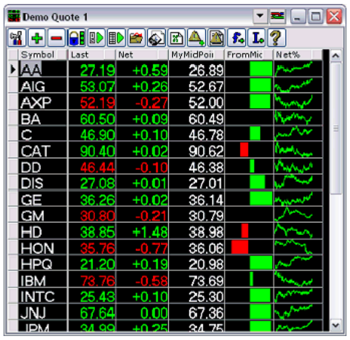


To copy entire sheet, use Dynamic Table's popup menu item **Sheets>Copy**.

Quote 1

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

Quote 1 demonstrates quote window's basic setup and usage.

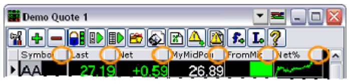


Symbol	Last	Net	MyMidPoi	FromMic	Net%
AA	27.19	+0.59	26.89	🟩	
AIG	53.07	+0.26	52.67	🟩	
AXP	52.19	-0.27	52.00	🟩	
BA	60.50	+0.09	60.49		
C	46.90	+0.10	46.78	🟩	
CAT	90.40	+0.02	90.62	🟥	
DD	46.44	-0.10	46.38	🟩	
DIS	27.08	+0.01	27.01	🟩	
GE	36.26	+0.02	36.14	🟩	
GM	30.80	-0.21	30.79		
HD	38.85	+1.48	38.98	🟥	
HON	35.76	-0.77	36.06	🟥	
HPQ	21.20	+0.19	20.98	🟩	
IBM	73.76	-0.58	73.69	🟩	
INTC	25.43	+0.10	25.30	🟩	
JNJ	67.64	0.00	67.36	🟩	
JPM	34.99	+0.25	34.75	🟩	

Rearranging Columns and Rows

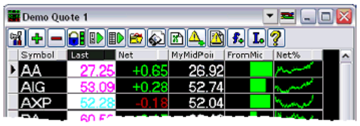
A quote window delivers the basic updated information to the user in a compact table format.

You can sort a quote window column by clicking on the sort button in column's heading.



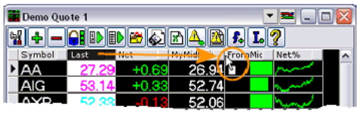
	Symbol	Last	Net	MyMidPoi	FromMic	Net%
▶	AA	27.19	+0.59	26.89	🟩	

You can also rearrange the columns by moving the column headings. First, left click on a column heading to select a column:



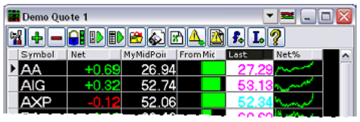
	Symbol	Last	Net	MyMidPoi	FromMic	Net%
▶	AA	27.25	+0.65	26.92	🟩	
	AIG	53.09	+0.28	52.74	🟩	
	AXP	52.28	-0.18	52.04	🟩	

Then, left click on the column again and hold the mouse button. Move the drag cursor to desire position.



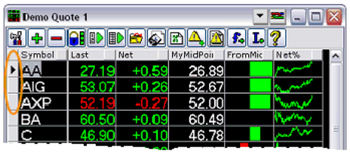
	Symbol	Last	Net	MyMid	FromMic	Net%
▶	AA	27.29	+0.69	26.94	🟩	
	AIG	53.14	+0.33	52.74	🟩	
	AXP	52.33	-0.13	52.06	🟩	

Release left mouse button to complete the move.



	Symbol	Net	MyMidPoi	From Mic	Last	Net%
▶	AA	+0.69	26.94	🟩	27.29	
	AIG	+0.32	52.74	🟩	53.13	
	AXP	-0.12	52.06	🟩	52.34	

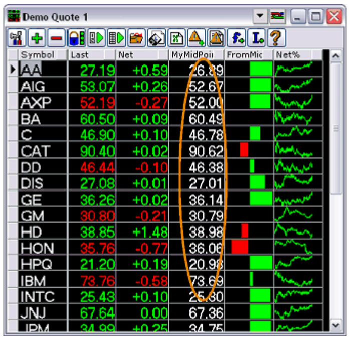
Rows can be rearranged in the same manner through the row handle at the far left.



	Symbol	Last	Net	MyMidPoi	FromMic	Net%
▶	AA	27.19	+0.59	26.89	🟩	
	AIG	53.07	+0.26	52.67	🟩	
	AXP	52.19	-0.27	52.00	🟩	
	BA	60.50	+0.09	60.49		
	C	46.90	+0.10	46.78	🟩	

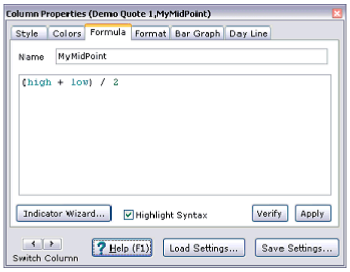
Formula Column

To display customized information, you can use the formula column to display simply calculations to complex indicator combinations. The values under the MyMidPoint column is an example.



Symbol	Last	Net	MyMidPoi	FromMic	Net%
AA	27.19	+0.59	26.89	🟩	
AIG	53.07	+0.26	52.67	🟩	
AXP	52.19	-0.27	52.00	🟩	
BA	60.50	+0.09	60.49		
C	46.90	+0.10	46.78	🟩	
CAT	90.40	+0.02	90.62	🟥	
DD	46.44	-0.10	46.38	🟩	
DIS	27.08	+0.01	27.01	🟩	
GE	36.26	+0.02	36.14	🟩	
GM	30.80	-0.21	30.79		
HD	38.85	+1.48	38.98	🟥	
HON	35.76	-0.77	36.06	🟥	
HPQ	21.20	+0.19	20.98	🟩	
IBM	73.76	-0.58	73.69	🟩	
INTC	25.43	+0.10	25.80	🟩	
JNJ	67.64	0.00	67.36	🟩	
JPM	34.99	+0.25	34.75	🟩	

Double-click on any cell in the MyMidPoint column to open the Column Properties window. Under the **Formula** tab, you can see how this formula column is defined.



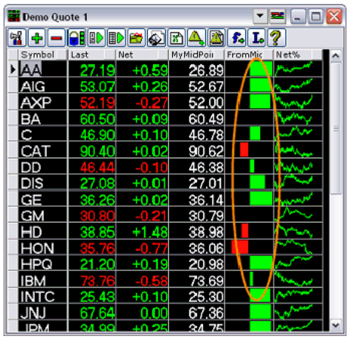
Name: MyMidPoint

```
(high + low) / 2
```

☑ Highlight Syntax

Bar Graphs

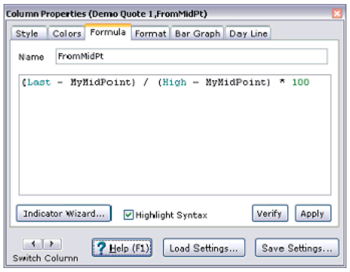
The column FromMidPt is set as a Bar Graph.



Symbol	Last	Net	MyMidPoi	FromMic	Net%
AA	27.19	+0.59	26.89	🟩	
AIG	53.07	+0.26	52.67	🟩	
AXP	52.19	-0.27	52.00	🟩	
BA	60.50	+0.09	60.49		
C	46.90	+0.10	46.78	🟩	
CAT	90.40	+0.02	90.62	🟥	
DD	46.44	-0.10	46.38	🟩	
DIS	27.08	+0.01	27.01	🟩	
GE	36.26	+0.02	36.14	🟩	
GM	30.80	-0.21	30.79		
HD	38.85	+1.48	38.98	🟥	
HON	35.76	-0.77	36.06	🟥	
HPQ	21.20	+0.19	20.98	🟩	
IBM	73.76	-0.58	73.69	🟩	
INTC	25.43	+0.10	25.30	🟩	
JNJ	67.64	0.00	67.36	🟩	
JPM	34.99	+0.25	34.75	🟩	

To see how this column is set up, double click on any cell in the FromMidPt column to open the Column Properties window.

Under the **Formula** tab, you can see this column is defined as a ratio between last price to midpoint and high price from midpoint.



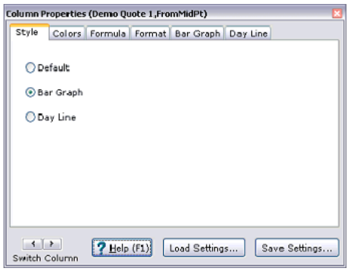
Column Properties (Demo Quote 1,FromMidPt)

Name: FromMidPt

```
(Last - MyMidPoint) / (High - MyMidPoint) * 100
```

☑ Highlight Syntax

Under **Style** tab, this column is set to display as **Bar Graph**.



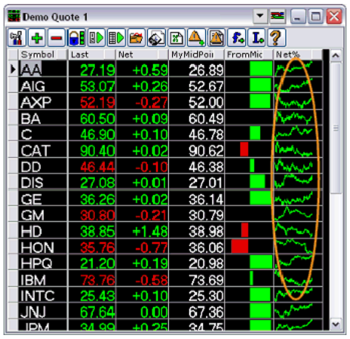
Column Properties (Demo Quote 1,FromMidPt)

○ Default
● Bar Graph
○ Day Line

Bar graph options are set under the **Bar Graph** tab.

Day Line Graph

The column Net% has been turn into a Day Line graph.



Demo Quote 1

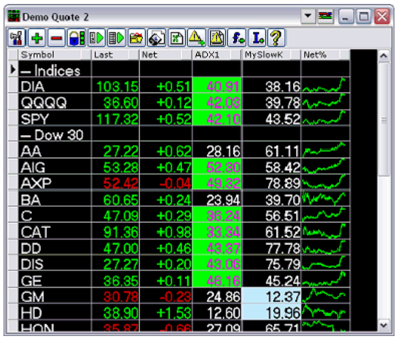
Symbol	Last	Net	MyMidPoi	FromMic	Net%
AA	27.19	+0.59	26.89	🟩	
AIG	53.07	+0.26	52.67	🟩	
AXP	52.19	-0.27	52.00	🟩	
BA	60.50	+0.09	60.49		
C	46.90	+0.10	46.78	🟩	
CAT	90.40	+0.02	90.62	🟥	
DD	46.44	-0.10	46.38	🟩	
DIS	27.08	+0.01	27.01	🟩	
GE	36.26	+0.02	36.14	🟩	
GM	30.80	-0.21	30.79		
HD	38.85	+1.48	38.98	🟥	
HON	35.76	-0.77	36.06	🟥	
HPQ	21.20	+0.19	20.98	🟩	
IBM	73.76	-0.58	73.69	🟩	
INTC	25.43	+0.10	25.30	🟩	
JNJ	67.64	0.00	67.36	🟩	
JPM	34.99	+0.25	34.75	🟩	

Day Line graph shows the price movement during a trading day. Day Line graph is not related to the actual column definition. You can turn any column into a Day Line graph by opening the Column Properties window and choose the **Day Line** option under the **Style** tab.

Quote 2

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

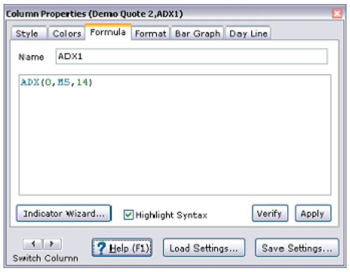
Quote 2 demonstrates how to display indicator values in quote window, and the use of coloring rule to highlight specific conditions.



Symbol	Last	Net	ADX1	MySlowK	Net%
– Indices					
DIA	103.15	+0.51	40.91	38.16	
QQQQ	36.60	+0.12	42.03	39.78	
SPY	117.32	+0.52	42.10	43.52	
– Dow 30					
AA	27.22	+0.62	28.16	61.11	
AIG	53.28	+0.47	52.30	58.42	
AXP	52.42	-0.04	49.32	78.89	
BA	60.65	+0.24	23.94	39.70	
C	47.09	+0.29	36.24	56.51	
CAT	91.36	+0.98	33.34	61.52	
DD	47.00	+0.46	43.37	77.78	
DIS	27.27	+0.20	43.06	75.79	
GE	36.35	+0.11	46.16	45.24	
GM	30.78	-0.23	24.86	12.37	
HD	38.90	+1.53	12.60	19.96	
HON	35.87	-0.66	27.09	65.71	

Indicator Calculation

Indicator calculation is part of quote formula. ADX1 is a formula column displaying the ADX indicator based on 5-minute data and using the standard 14-period parameter. You can see how ADX1 is written by double clicking a cell under ADX1 column to open Column Properties window.

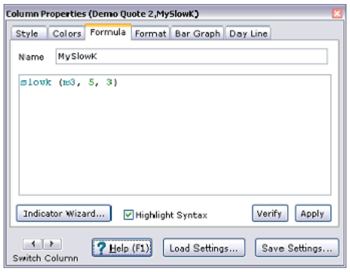


Name: ADX1

```
ADX(0,M5,14)
```

☑ Highlight Syntax

MySlowK is a formula column displaying the SlowK indicator based on 3-minute data with the standard 5, 3 parameters. You can see how MySlowK is written by double clicking a cell under MySlowK column to open Column Properties window.



Name: MySlowK

```
slowk (m3, 5, 3)
```

☑ Highlight Syntax

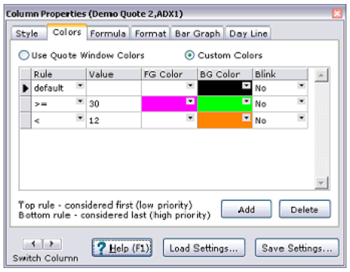
Notice that there is no restriction on how many time frames you are using within a quote window. We are using two time frames here (5-min and 3-min) and you can experiment with adding more formula columns using different time frames.

Coloring

Just showing the indicator values is neat. But that cannot help us identify potential setup at a glance. Thus, we've highlighted the ADX values when they are greater than or equal to 30, an indication of strong trend.

We've also set the column to show in orange color when the ADX value fall below 12, an indication of choppy or trendless environment.

Coloring settings are under **Colors** tab in Column Properties window.

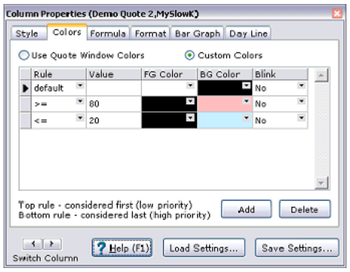


○ Use Quote Window Colors ● Custom Colors

Rule	Value	FG Color	BG Color	Blink
default			⬛	No
>=	30	🟪	🟩	No
<	12		🟧	No

Top rule - considered first (low priority)
Bottom rule - considered last (high priority)

The column MySlowK has its own set of coloring rules.



○ Use Quote Window Colors ● Custom Colors

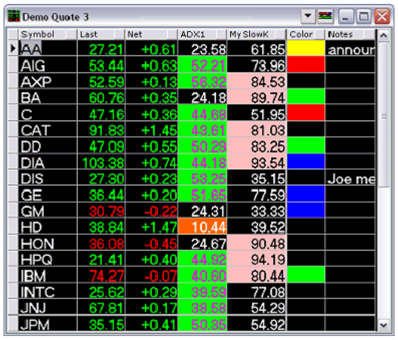
Rule	Value	FG Color	BG Color	Blink
default			⬛	No
>=	80	⬛	🟥	No
<=	20	⬛	🟦	No

Top rule - considered first (low priority)
Bottom rule - considered last (high priority)

Quote 3

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

Quote 3 demonstrates the use of color code, notes and row filtering. It is an extension of Quote 2. For information about how ADX1 and MySlowK are defined, see ***Tour of Prebuilt Groups, Quote 2*** (see "Quote 2" on page 79).



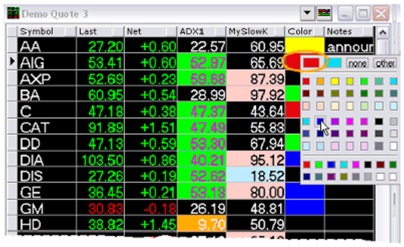
Demo Quote 3

Symbol	Last	Net	ADX1	My SlowK	Color	Notes
AA	27.21	+0.61	23.58	61.85	🟨	annour
AIG	53.44	+0.63	52.21	73.96	🟥	
AXP	52.59	+0.13	56.32	84.53		
BA	60.76	+0.35	24.18	89.74	🟩	
C	47.16	+0.36	44.68	51.95	🟥	
CAT	91.83	+1.45	43.61	81.03		
DD	47.09	+0.55	50.29	83.25	🟩	
DIA	103.38	+0.74	44.18	93.54	🟦	
DIS	27.30	+0.23	53.25	35.15		Joe me
GE	36.44	+0.20	51.65	77.59	🟦	
GM	30.79	-0.22	24.31	33.33	🟦	
HD	38.84	+1.47	10.44	39.52		
HON	36.08	-0.45	24.67	90.48		
HPQ	21.41	+0.40	44.92	94.19		
IBM	74.27	-0.07	40.60	80.44	🟩	
INTC	25.62	+0.29	39.59	77.08		
JNJ	67.81	+0.17	38.58	54.29		
JPM	35.15	+0.41	50.35	54.92		

Color Code

You can use quote window to track symbols with your own input. An example of that is using the ColorCode column.

When you left-click on the ColorCode column, you will be able to choose a color to be assigned to the symbol.



Demo Quote 3

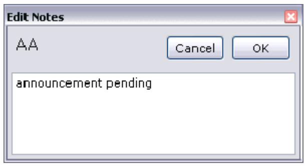
Symbol	Last	Net	ADX1	MySlowK	Color	Notes
AA	27.20	+0.60	22.57	60.95	🟨	annour
AIG	53.41	+0.60	52.97	65.69	🟥	
AXP	52.69	+0.23	59.68	87.39		
BA	60.95	+0.54	28.99	97.92	🟩	
C	47.18	+0.38	47.37	43.64	🟥	
CAT	91.89	+1.51	47.49	55.83		
DD	47.13	+0.59	53.30	67.94	🟩	
DIA	103.50	+0.86	40.21	95.12	🟦	
DIS	27.26	+0.19	52.62	18.52		
GE	36.45	+0.21	53.13	80.00	🟦	
GM	30.83	-0.18	26.19	48.81	🟦	
HD	38.82	+1.45	9.70	50.79		

You can then sort the ColorCode column based on color.

The ColorCode column can also be used as a simple tool that highlights symbols of special interest to you at the moment.

Notes Column

You can enter notes for a symbol under the **Notes** column. Simply double click a cell to open the editor to enter notes.



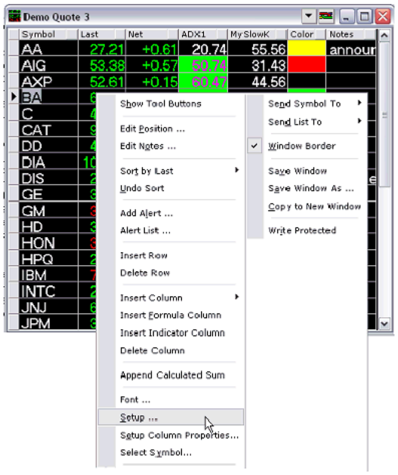
AA

announcement pending

Row Filtering

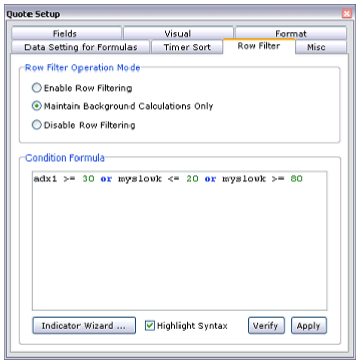
Quote window can filter its rows and display the ones that meet your criteria.

Take a look at the example row filter formula by right-click on the quote window to bring up the popup menu, then choose **Setup**.



Symbol	Last	Net	ADX1	My SlowK	Color	Notes
AA	27.21	+0.61	20.74	55.56	🟨	annour
AIG	53.38	+0.57	50.74	31.43	🟥	
AXP	52.61	+0.15	60.47	44.56		
BA	6					
C	4					
CAT	9					
DD	4					
DIA	10					
DIS	2					e
GE	3					
GM	3					
HD	3					
HON	3					
HPQ	2					
IBM	7					
INTC	2					
JNJ	6					
JPM	3					

You will then open the Quote Setup window. Filtering settings are under **Row Filter** tab.



Row Filter Operation Mode

○ Enable Row Filtering
● Maintain Background Calculations Only
○ Disable Row Filtering

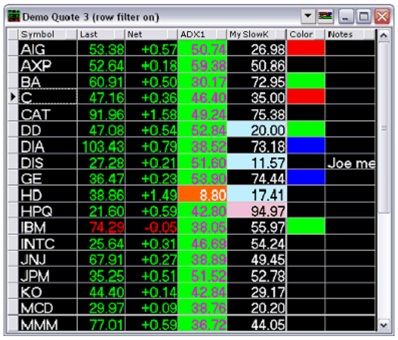
Condition Formula

```
adx1 >= 30 or myslowk <= 20 or myslowk >= 80
```

☑ Highlight Syntax

When filtering is on, only symbols that makes the formula evaluates to true are displayed in quote window. In the example, the filtering formula will display only symbols that are trending (high adx value) or stochastics at the overbought or oversold levels.

To activate the row filtering, you can choose **Enable Row Filtering** in the Quote Setup window, or, if the Quote Setup window is not opened, choose from the Quote window popup menu, **Start Row Filtering**, to enable the function.



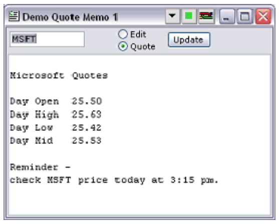
Symbol	Last	Net	ADX1	My SlowK	Color	Notes
AIG	53.38	+0.57	50.74	26.98	🟥	
AXP	52.64	+0.18	59.38	50.86		
BA	60.91	+0.50	30.17	72.95	🟩	
C	47.16	+0.36	46.40	35.00	🟥	
CAT	91.96	+1.58	49.24	75.38		
DD	47.08	+0.54	52.84	20.00	🟩	
DIA	103.43	+0.79	38.52	73.18	🟦	
DIS	27.28	+0.21	51.60	11.57		Joe me
GE	36.47	+0.23	53.90	74.44	🟦	
HD	38.86	+1.49	8.80	17.41		
HPQ	21.60	+0.59	42.80	94.97		
IBM	74.29	-0.05	38.05	55.97	🟩	
INTC	25.64	+0.31	46.69	54.24		
JNJ	67.91	+0.27	38.89	49.45		
JPM	35.25	+0.51	51.52	52.78		
KO	44.40	+0.14	42.84	29.17		
MCD	29.97	+0.09	38.76	20.20		
MMM	77.01	+0.59	36.72	44.05		

In the screen shot above, row filter is on. Notice that the symbols that do not meet the filtering criteria (e.g. AA) are not displayed.

Quote Memo 1

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

Quote Memo 1 demonstrates Quote Memo, a memo window that can display quotes, formulas and indicator values.



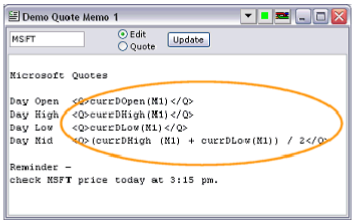
```
Microsoft Quotes

Day Open  25.50
Day High  25.63
Day Low   25.42
Day Mid   25.53

Reminder -
check MSFT price today at 3:15 pm.
```

Quote Memo is good for displaying information that needs less restrictions than within the context of quote window or dynamic grids.

Quote Memo is also a better function window for experimenting with formulas before putting them into quote columns because you can limit your test to a single symbol with extra calculations displayed to help you construct your formulas.

In general, it is also a good tool for typing comments, notes, etc. just like what you see here in this window.

To edit content of quote memo, choose **Edit** in Quote Memo. Notice that quotes, formulas and indicators are enclosed by `<Q>` and `</Q>`.



```
Microsoft Quotes

Day Open  <Q>currDOpen(M1)</Q>
Day High  <Q>currDHigh(M1)</Q>
Day Low   <Q>currDLow(M1)</Q>
Day Mid   <Q>(currDHigh (M1) + currDLow(M1)) / 2</Q>

Reminder -
check MSFT price today at 3:15 pm.
```

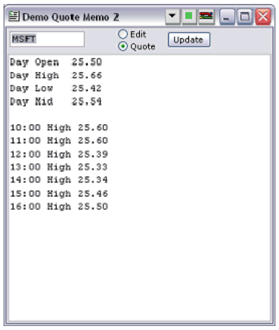
Choose **Quote** to start displaying quotes.

Quote Memo 2

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

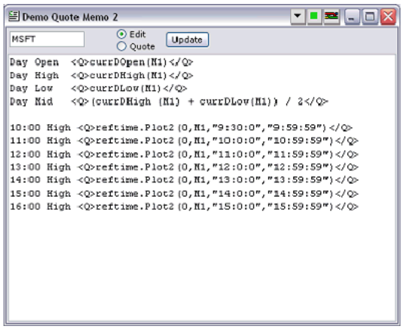
Quote Memo 2 is the second example of Quote Memo. It demonstrates the use of the reftime indicator.

Quote Memo 2 in Quote mode:



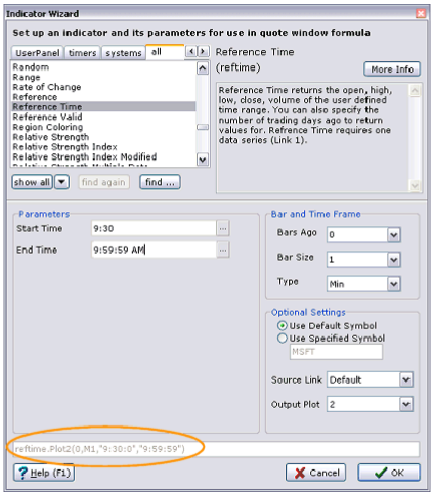
```
Day Open  25.50
Day High  25.66
Day Low   25.42
Day Mid   25.54

10:00 High 25.60
11:00 High 25.60
12:00 High 25.39
13:00 High 25.33
14:00 High 25.34
15:00 High 25.46
16:00 High 25.50
```

Quote Memo 2 in Edit mode:



```
Day Open  <Q>currDOpen(M1)</Q>
Day High  <Q>currDHigh(M1)</Q>
Day Low   <Q>currDLow(M1)</Q>
Day Mid   <Q>(currDHigh (M1) + currDLow(M1)) / 2</Q>

10:00 High <Q>reftime.Plot2(0,M1,"9:30:0","9:59:59")</Q>
11:00 High <Q>reftime.Plot2(0,M1,"10:0:0","10:59:59")</Q>
12:00 High <Q>reftime.Plot2(0,M1,"11:0:0","11:59:59")</Q>
13:00 High <Q>reftime.Plot2(0,M1,"12:0:0","12:59:59")</Q>
14:00 High <Q>reftime.Plot2(0,M1,"13:0:0","13:59:59")</Q>
15:00 High <Q>reftime.Plot2(0,M1,"14:0:0","14:59:59")</Q>
16:00 High <Q>reftime.Plot2(0,M1,"15:0:0","15:59:59")</Q>
```

Notice that Quote Memo 2 is referencing the Plot2 of reftime. Plot2 is the high value. The syntax of indicator can get quite complex, but there is no need for you to memorize the syntax. You can right click on Quote Memo to open pop menu, and choose **Indicator Wizard** (or press Ctrl-W) to open Indicator Wizard. Indicator Wizard is a tool to help you construct indicator syntax.



Indicator Wizard

Set up an indicator and its parameters for use in quote window formula

Random
Range
Rate of Change
Reference
Reference Time
Reference Valid
Region Coloring
Relative Strength
Relative Strength Index
Relative Strength Index Modified

Reference Time
(reftime)

Reference Time returns the open, high, low, close, volume of the user defined time range. You can also specify the number of trading days ago to return values for. Refrence Time requires one data series (Link 1).

Parameters
Start Time: 9:30
End Time: 9:59:59 AM

Bar and Time Frame
Bars Ago: 0
Bar Size: 1
Type: Min

Optional Settings
● Use Default Symbol
○ Use Specified Symbol: MSFT
Source Link: Default
Output Plot: 2

reftime.Plot2(0,M1,"9:30:0","9:59:59")

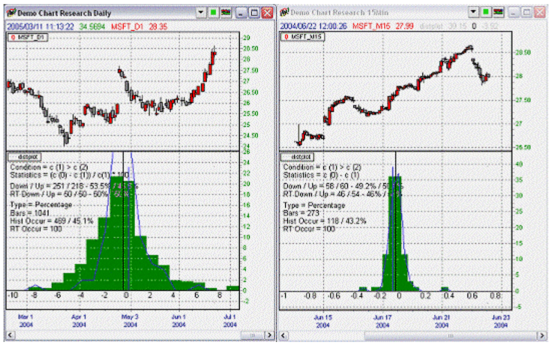
Research Charts

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

This group consists of 2 time chart windows. It demonstrates the use of the Distribution Plot power indicator (***Power Indicator Guide, Distribution Plot*** (see "Distribution Plot" on page 1307))

Demo Chart Research Daily shows the distribution of MSFT price changes in percent one day after an up close day.

Demo Chart Research 15Min shows the distribution of MSFT price changes in actual amount after an up close bar set up.



If you are interested in conducting your own research, you can modify the condition and value parameter of distribution plot indicator to test for almost anything.

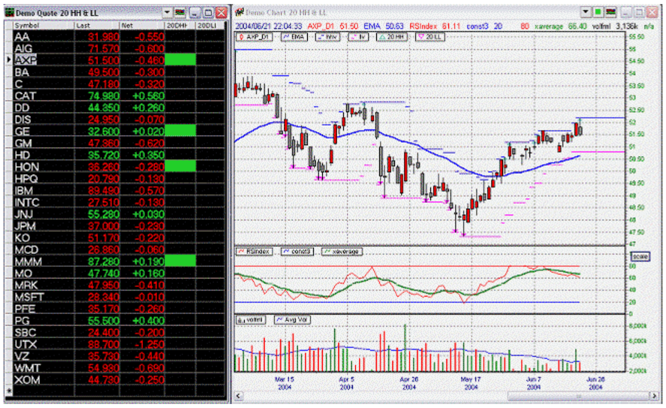
RT Quote and Chart 1

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

This group consists of 2 function windows. It demonstrates the use of quote window formula and connectivity with a time chart window.

The quote window Demo Quote 20 HH & LL has 2 formula columns, one detects a 20 days highest high breakout and the other one detects a 20 days lowest low breakdown.

When the criteria is meet, the corresponding cell will change color so that you can easily spot the signal. If you would like to see the chart, then all you have to do is double click the corresponding symbol in the quote window and the chart will load the data your have chosen.



Demo Quote 20 HH & LL

Symbol	Last	Net	20DHI	20DLI
AA	31.980	-0.550		
AIG	71.570	-0.600		
AXP	51.500	-0.460	🟩	
BA	49.500	-0.300		
C	47.180	-0.320		
CAT	74.980	+0.560		
DD	44.350	+0.260		
DIS	24.950	-0.070		
GE	32.600	+0.020	🟩	
GM	47.360	-0.620		
HD	35.720	+0.350		
HON	36.260	-0.280	🟩	
HPQ	20.790	-0.130		
IBM	89.490	-0.570		
INTC	27.510	-0.130		
JNJ	55.280	+0.030		
JPM	37.000	-0.230		
KO	51.170	-0.220		
MCD	26.860	-0.060		
MMM	87.280	+0.190	🟩	
MO	47.740	+0.160		
MRK	47.950	-0.410		
MSFT	28.340	-0.010		
PFE	35.170	-0.260		
PG	55.500	+0.400		
SBC	24.400	-0.200		
UTX	88.700	-1.250		
VZ	35.730	-0.440		
WMT	54.930	-0.690		
XOM	44.730	-0.250		

RT Quote and Chart 2

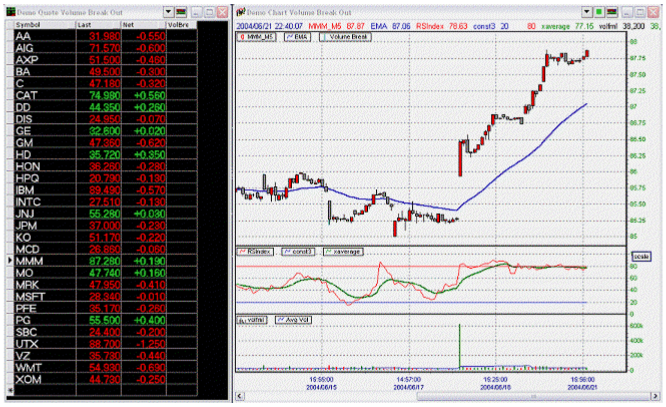
Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

This group consists of 2 function windows. It demonstrates the use of quote window formula with minute data and the use of highlight bar formula indicator within a chart.

The quote window Demo Quote Volume Break Out has a formula column checking for 5-min bar volume break out. You can double click within the column to take a look at how simple the formula is.

The formula column will change color if the particular symbol has a volume break out happening now. You can double click on the symbol to send it to the chart.

The volume breakout bar is highlighted from low to high with user changeable color. If you want to modify the settings of the Volume Break, simply double-click on the legend of the Volume Break indicator. The edit indicator window will show up and you can even modify the formula to get different results.



Demo Quote Volume Break Out

Symbol	Last	Net	VolBre
AA	31.980	-0.550	
AIG	71.570	-0.600	
AXP	51.500	-0.460	
BA	49.500	-0.300	
C	47.180	-0.320	
CAT	74.980	+0.560	
DD	44.350	+0.260	
DIS	24.950	-0.070	
GE	32.600	+0.020	
GM	47.360	-0.620	
HD	35.720	+0.350	
HON	36.260	-0.280	
HPQ	20.790	-0.130	
IBM	89.490	-0.570	
INTC	27.510	-0.130	
JNJ	55.280	+0.030	
JPM	37.000	-0.230	
KO	51.170	-0.220	
MCD	26.860	-0.060	
MMM	87.280	+0.190	
MO	47.740	+0.160	
MRK	47.950	-0.410	
MSFT	28.340	-0.010	
PFE	35.170	-0.260	
PG	55.500	+0.400	
SBC	24.400	-0.200	
UTX	88.700	-1.250	
VZ	35.730	-0.440	
WMT	54.930	-0.690	
XOM	44.730	-0.250	

Demo Chart Volume Break Out

2004/06/21 22:40:07 MMM_M5 87.87 EMA 87.06 RSIndex 78.63 const3 20 80 xaverage 77.15 volfml 38,200 38,

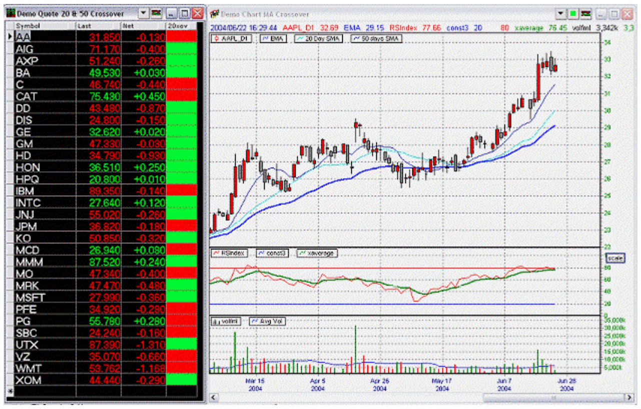
RT Quote and Chart 3

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

This group consists of 2 function windows. It demonstrates the use of quote window formula with multiple indicators.

The quote window Demo Quote 20 & 50 Crossover has a formula column that compares the positions of 2 moving averages relative to each other. If the fast one is above the slower one, the cell color will change to green; otherwise it will display in red.

You can modify the formula column by double-clicking any cell within the column "20xover50". You can try modify the formula to use different moving average period and see what happen.



Demo Quote 20 & 50 Crossover

Symbol	Last	Net	20xov
AA	31.850	-0.130	🟥
AIG	71.170	-0.400	🟩
AXP	51.240	-0.260	🟩
BA	49.530	+0.030	🟩
C	46.740	-0.440	🟥
CAT	75.430	+0.450	🟥
DD	43.480	-0.870	🟩
DIS	24.800	-0.150	🟩
GE	32.620	+0.020	🟩
GM	47.330	-0.030	🟩
HD	34.790	-0.930	🟩
HON	36.510	+0.250	🟩
HPQ	20.800	+0.010	🟩
IBM	89.350	-0.140	🟥
INTC	27.640	+0.120	🟩
JNJ	55.020	-0.260	🟩
JPM	36.820	-0.180	🟥
KO	50.850	-0.320	🟩
MCD	26.940	+0.080	🟥
MMM	87.520	+0.240	🟩
MO	47.340	-0.400	🟥
MRK	47.470	-0.480	🟩
MSFT	27.990	-0.360	🟩
PFE	34.920	-0.280	🟥
PG	55.780	+0.280	🟥
SBC	24.240	-0.160	🟥
UTX	87.390	-1.310	🟩
VZ	35.070	-0.660	🟥
WMT	53.762	-1.168	🟥
XOM	44.440	-0.290	🟩

RT Scan and Chart 1

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

This group consists of 2 function windows. It demonstrates the use of pattern scanner with time chart.

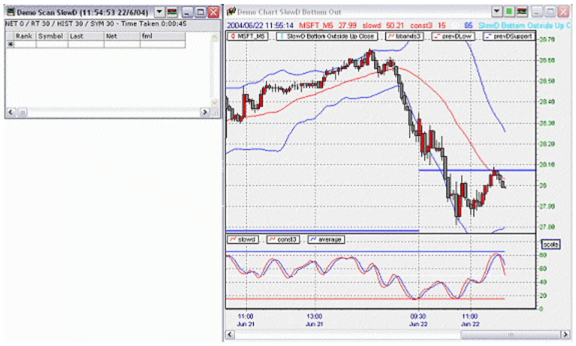
The pattern scanner Demo Scan SlowD scan the Nasdaq 100 for the setup of stochastic slowk at the bottom while the price bar started to rise.

You can change the pattern scanner to scan other symbol list by making the pattern scanner window active and press the space bar key to get the pattern setup window.

You can also modify the indicators used in this scanner to something you want.

To start scanning, simply right-click on the pattern scanner window and choose **Scan Now**.

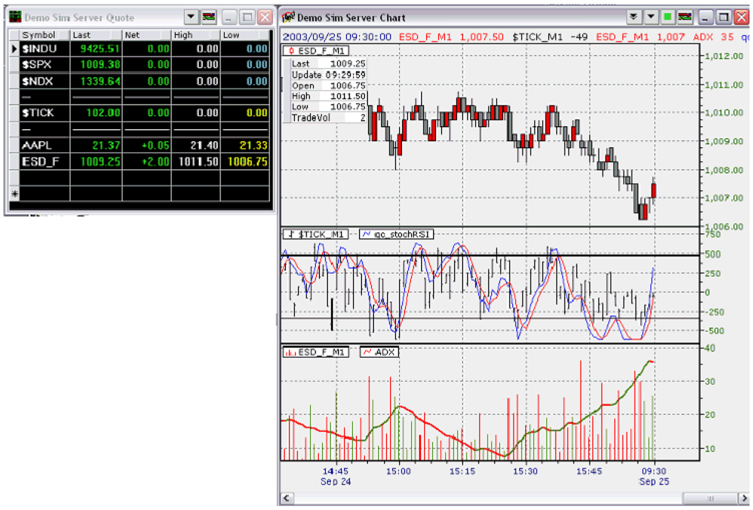
Once scanning is completed, you can double-click on a symbol to send it to the chart for more detail examination.


Demo Scan SlowD (11:54:53 22/6/04)
NET 0 / RT 30 / HIST 30 / SYM 30 - Time Taken 0:00:45
Rank
Symbol
Last
Net
fml
Demo Chart SlowD Bottom Out
2004/06/22 11:56:14 MSFT_M5 27.99 slowd 50.21 const3 15 86 SlowD Bottom Outside Up C
MSFT_M5
SlowD Bottom Outside Up Close
bbands3
prevDLow
prevDSupport
28.70
28.60
28.50
28.40
28.30
28.20
28.10
28
27.90
27.80
slowd
const3
average
scale
80
60
40
20
0
11:00 Jun 21
13:00 Jun 21
09:30 Jun 22
11:00 Jun 22

Sim Server Demo Data

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

Sim Server Demo Data demonstrates NeoTicker®'s simulation capability. This group is designed to use with the sample data that comes with every NeoTicker® installation.



Demo Sim Server Quote

Symbol	Last	Net	High	Low
$INDU	9425.51	0.00	0.00	0.00
$SPX	1009.38	0.00	0.00	0.00
$NDX	1339.64	0.00	0.00	0.00
—				
$TICK	102.00	0.00	0.00	0.00
—				
AAPL	21.37	+0.05	21.40	21.33
ESD_F	1009.25	+2.00	1011.50	1006.75
*				

Demo Sim Server Chart

2003/09/25 09:30:00 ESD_F_M1 1,007.50 $TICK_M1 -49 ESD_F_M1 1,007 ADX 35 qc

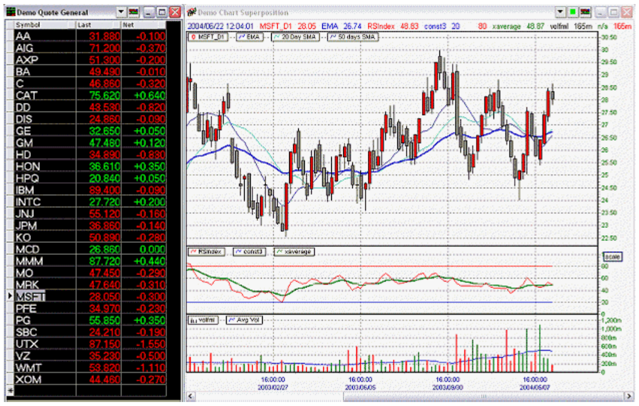
Last	1009.25
Update	09:29:59
Open	1006.75
High	1011.50
Low	1006.75
TradeVol	2

For more information about setting up NeoTicker® as a simulator and using this group, see ***Setting up NeoTicker® as Simulator*** (on page 9).

Superposition Charts

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

This group consists of 2 function windows. It demonstrates the use of price range chart, a type of superposition charts.



Demo Quote General

Symbol	Last	Net
AA	31.880	-0.100
AIG	71.200	-0.370
AXP	51.300	-0.200
BA	49.490	-0.010
C	46.860	-0.320
CAT	75.620	+0.640
DD	43.530	-0.820
DIS	24.860	-0.090
GE	32.650	+0.050
GM	47.480	+0.120
HD	34.890	-0.830
HON	36.610	+0.350
HPQ	20.840	+0.050
IBM	89.400	-0.090
INTC	27.720	+0.200
JNJ	55.120	-0.160
JPM	36.860	-0.140
KO	50.890	-0.280
MCD	26.860	0.000
MMM	87.720	+0.440
MO	47.450	-0.290
MRK	47.640	-0.310
MSFT	28.050	-0.300
PFE	34.970	-0.230
PG	55.850	+0.350
SBC	24.210	-0.190
UTX	87.150	-1.550
VZ	35.230	-0.500
WMT	53.820	-1.110
XOM	44.460	-0.270

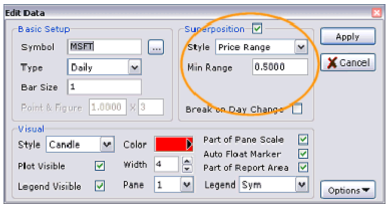
Demo Chart Superposition

2004/06/22 12:04:01 MSFT_D1 28.05 EMA 26.74 RSIndex 48.83 const3 20 80 xaverage 48.87 volfml 165m n/a 165m

In a price range chart, only when there is sufficient price movement, a bar is generated. This reduces much of the noise in price movements, making the chart easier to read. Notice that in a superposition chart, the time axis scaling is non-linear because the noisy bars have been filtered out.

You can double click on a symbol on the quote window to change the symbol in the chart. You can also open a non-superposition chart side-by-side for a comparison.

In this particular example, a bar is generated for every $0.5 movement. You can edit the data to see how the price range is set up. For intraday charts, you may want to reduce the value for movement.



Edit Data

Basic Setup
Symbol: MSFT
Type: Daily
Bar Size: 1
Point & Figure: 1.0000 X 3

Superposition ☑
Style: Price Range
Min Range: 0.5000
Break on Day Change □

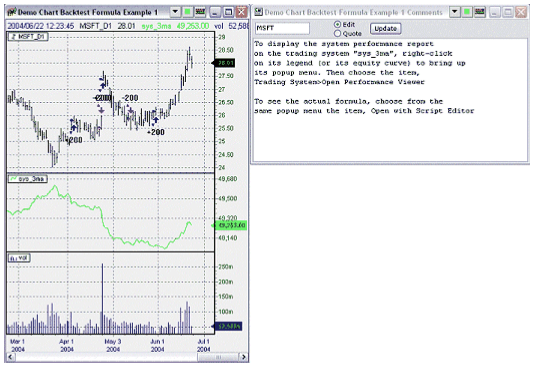
Visual
Style: Candle
Color
Plot Visible ☑
Width: 4
Legend Visible ☑
Pane: 1
Part of Pane Scale ☑
Auto Float Marker ☑
Part of Report Area ☑
Legend: Sym

For more information on different types of Superposition charts, see ***Superposition Chart*** (on page 1178).

System Testing 1 Formula System

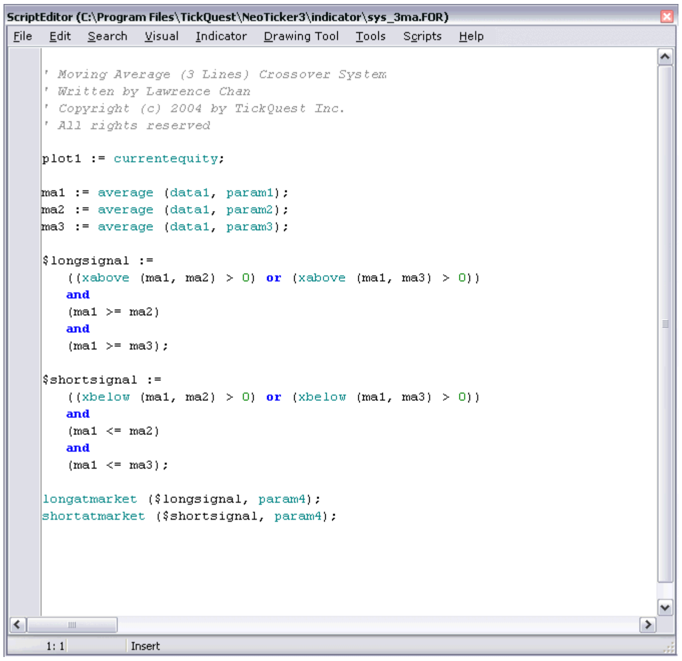
Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

This sample group demonstrates a moving average crossover trading system written in NeoTicker® formula language. The trading system is "sys_3ma" shown in the chart, displaying the equity curve of the system.



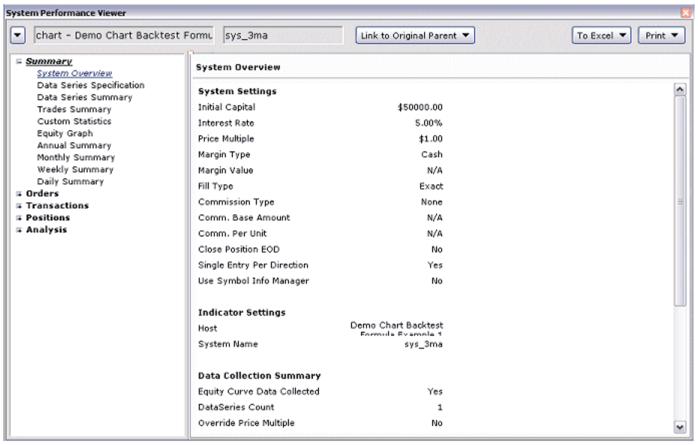
```
To display the system performance report
on the trading system "sys_3ma", right-click
on its legend (or its equity curve) to bring up
its popup menu. Then choose the item,
Trading System>Open Performance Viewer

To see the actual formula, choose from the
same popup menu the item, Open with Script Editor
```

To see the formula that implements this trading system, right click on the "sys_3ma" legend to open pop up menu, choose **Open with Script Editor**. The formula demonstrates how to call indicators within formula with user parameters and how to issue orders.


ScriptEditor (C:\Program Files\TickQuest\NeoTicker3\indicator\sys_3ma.FOR)
File
Edit
Search
Visual
Indicator
Drawing Tool
Tools
Scripts
Help
1: 1
Insert

```
' Moving Average (3 Lines) Crossover System
' Written by Lawrence Chan
' Copyright (c) 2004 by TickQuest Inc.
' All rights reserved

plot1 := currentequity;

ma1 := average (data1, param1);
ma2 := average (data1, param2);
ma3 := average (data1, param3);

$longsignal :=
   ((xabove (ma1, ma2) > 0) or (xabove (ma1, ma3) > 0))
   and
   (ma1 >= ma2)
   and
   (ma1 >= ma3);

$shortsignal :=
   ((xbelow (ma1, ma2) > 0) or (xbelow (ma1, ma3) > 0))
   and
   (ma1 <= ma2)
   and
   (ma1 <= ma3);

longatmarket ($longsignal, param4);
shortatmarket ($shortsignal, param4);
```

To see the performance of the system, right click on the "sys_3ma" legend to open pop up menu, choose **Trading System>Open Performance Viewer**.



System Overview

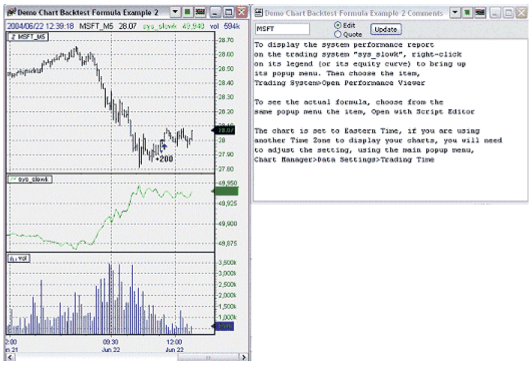
System Settings	
Initial Capital	$50000.00
Interest Rate	5.00%
Price Multiple	$1.00
Margin Type	Cash
Margin Value	N/A
Fill Type	Exact
Commission Type	None
Comm. Base Amount	N/A
Comm. Per Unit	N/A
Close Position EOD	No
Single Entry Per Direction	Yes
Use Symbol Info Manager	No
Indicator Settings	
Host	Demo Chart Backtest Formula Example 1
System Name	sys_3ma
Data Collection Summary	
Equity Curve Data Collected	Yes
DataSeries Count	1
Override Price Multiple	No

System Testing 2 Formula System

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

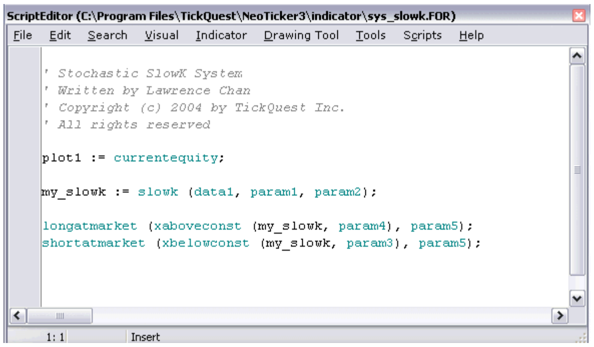
This sample group demonstrates a buy at oversold, sell at overbought trading system in NeoTicker® formula language. Buying is triggered by price level crossing below oversold condition defined using slowk indicator. Selling is triggered by price level crossing above overbought condition defined using slowk indicator.

The trading system is "sys_slowk" shown in the chart, displaying the equity curve of the system.



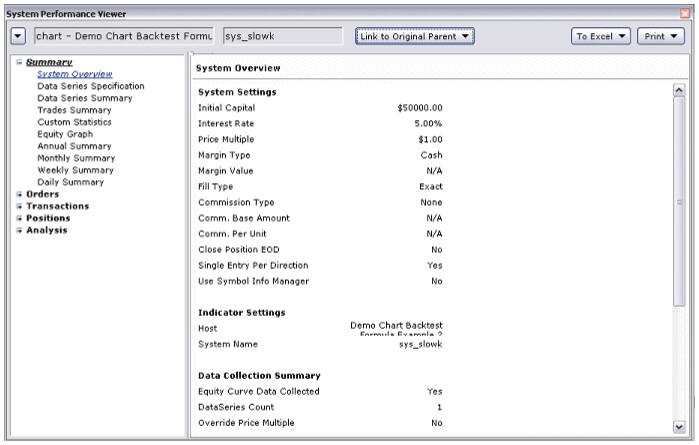
```
To display the system performance report
on the trading system "sys_slowk", right-click
on its legend (or its equity curve) to bring up
its popup menu. Then choose the item,
Trading System>Open Performance Viewer

To see the actual formula, choose from the
same popup menu the item, Open with Script Editor

The chart is set to Eastern Time, if you are using
another Time Zone to display your charts, you will need
to adjust the setting, using the main popup menu,
Chart Manager>Data Settings>Trading Time
```

To see the formula that implements this trading system, right click on the "sys_slowk" legend to open pop up menu, choose **Open with Script Editor**. The formula demonstrates how to call indicators within formula with user parameters and how to issue orders.


ScriptEditor (C:\Program Files\TickQuest\NeoTicker3\indicator\sys_slowk.FOR)
File Edit Search Visual Indicator Drawing Tool Tools Scripts Help
1: 1
Insert

```
' Stochastic SlowK System
' Written by Lawrence Chan
' Copyright (c) 2004 by TickQuest Inc.
' All rights reserved

plot1 := currentequity;

my_slowk := slowk (data1, param1, param2);

longatmarket (xaboveconst (my_slowk, param4), param5);
shortatmarket (xbelowconst (my_slowk, param3), param5);
```

To see the performance of the system, right click on the "sys_slowk" legend to open pop up menu, choose **Trading System>Open Performance Viewer**.



System Overview

System Settings	
Initial Capital	$50000.00
Interest Rate	5.00%
Price Multiple	$1.00
Margin Type	Cash
Margin Value	N/A
Fill Type	Exact
Commission Type	None
Comm. Base Amount	N/A
Comm. Per Unit	N/A
Close Position EOD	No
Single Entry Per Direction	Yes
Use Symbol Info Manager	No
Indicator Settings	
Host	Demo Chart Backtest Formula Example 2
System Name	sys_slowk
Data Collection Summary	
Equity Curve Data Collected	Yes
DataSeries Count	1
Override Price Multiple	No

System Testing 3 Text Report

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

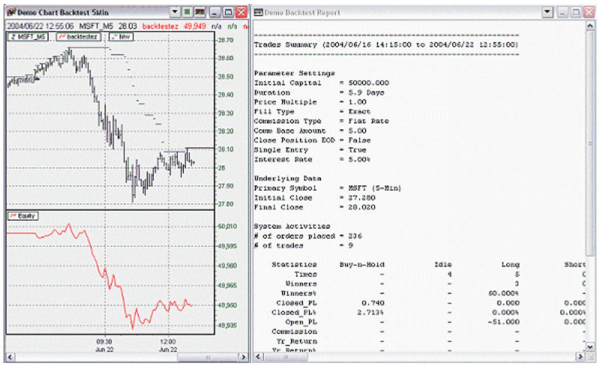
This group consists of 1 time chart window and 1 report window. It demonstrates:

- How to display performance statistics in text that can easily copy and paste to other programs.
- How to utilizes the different plots of Backtest EZ.

If you open the trading system "Backtest EZ". In the editor, under the **Report** tab. It shows how to set up the system to output to a report window.

The default setting for Backtest EZ is to display the equity curve. In this example, we modify Backtest EZ (under the **Visual** tab) to display its other plots, namely buy sell markers.

The indicator "Equity" extracts the equity curve from Backtest EZ and displays it in another pane.



```
------------------------------------------------------------
Trades Summary (2004/06/16 14:15:00 to 2004/06/22 12:55:00)
------------------------------------------------------------

Parameter Settings
Initial Capital     = 50000.000
Duration            = 5.9 Days
Price Multiple      = 1.00
Fill Type           = Exact
Commission Type     = Flat Rate
Comm Base Amount    = 5.00
Close Position EOD  = False
Single Entry        = True
Interest Rate       = 5.00%

Underlying Data
Primary Symbol      = MSFT (5-Min)
Initial Close       = 27.280
Final Close         = 28.020

System Activities
# of orders placed  = 236
# of trades         = 9

    Statistics     Buy-n-Hold          Idle          Long        Short
         Times              -             4             5            0
       Winners              -             -             3            0
      Winners%              -             -       60.000%            -
     Closed_PL          0.740             -         0.000        0.000
    Closed_PL%         2.713%             -        0.000%       0.000%
       Open_PL              -             -       -51.000        0.000
    Commission              -             -             -            -
     Yr_Return              -             -             -            -
    Yr_Return%              -             -             -            -
```

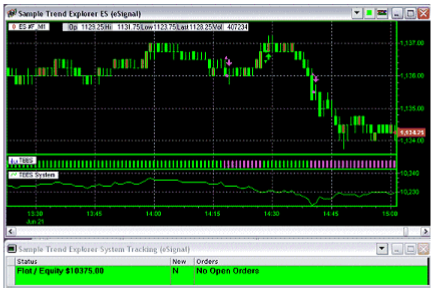
Trend Explorer ES

Trend Explorer ES is not available to demo version of NeoTicker®.

This group uses commodities symbol, so it is data vendor dependent. Open the group that matches the current data vendor you are using.

Your computer must set to Eastern Time, e.g. New York time for this prebuit group to work.

This group demonstrates Trend Explorer ES signal that comes with NeoTicker®. A green bar indicates a bullish signal, a red bar indicates a bearish signal, a gray bar indicates a neutral signal.



Sample Trend Explorer ES (eSignal)

Sample Trend Explorer System Tracking (eSignal)

Status	New	Orders
Flat / Equity $10375.00	N	No Open Orders

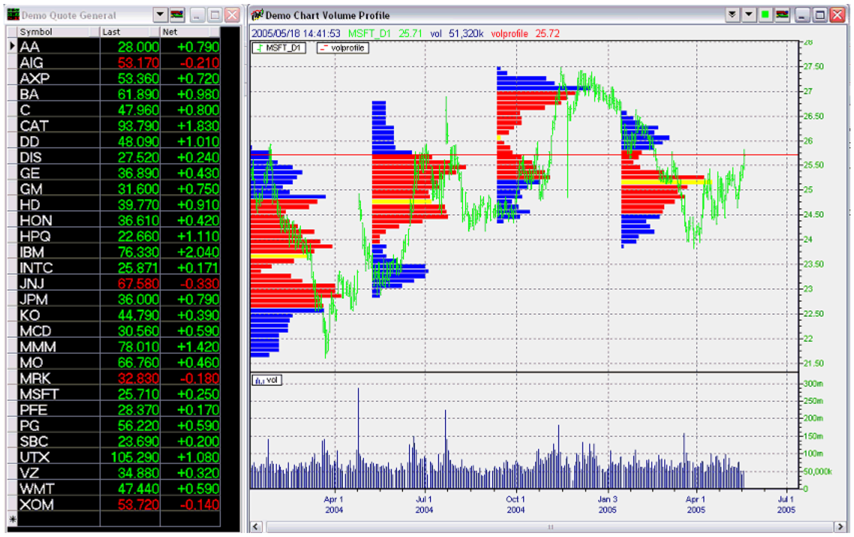
For more information about Trend Explorer ES, see ***Trend Explorer ES Indicator*** (on page 1243).

Volume Profile Chart

Demo version users: please choose **Group>Close All Groups** in main window to close all groups before opening a new group.

This group demonstrates the use of Volume Profile indicator.

Volume Profile is a NeoTicker® power indicator that displays total volume traded at different price levels. Volume Profile indicator marks support resistance levels.



Symbol	Last	Net
AA	28.000	+0.790
AIG	53.170	-0.210
AXP	53.360	+0.720
BA	61.890	+0.980
C	47.960	+0.800
CAT	93.790	+1.830
DD	48.090	+1.010
DIS	27.520	+0.240
GE	36.890	+0.430
GM	31.600	+0.750
HD	39.770	+0.910
HON	36.610	+0.420
HPQ	22.660	+1.110
IBM	76.330	+2.040
INTC	25.871	+0.171
JNJ	67.580	-0.330
JPM	36.000	+0.790
KO	44.790	+0.390
MCD	30.560	+0.590
MMM	78.010	+1.420
MO	66.760	+0.460
MRK	32.830	-0.180
MSFT	25.710	+0.250
PFE	28.370	+0.170
PG	56.220	+0.590
SBC	23.690	+0.200
UTX	105.290	+1.080
VZ	34.880	+0.320
WMT	47.440	+0.590
XOM	53.720	-0.140

2005/05/18 14:41:53 MSFT_D1 25.71 vol 51,320k volprofile 25.72

For more information about setting up Volume Profile indicator, see ***Power Indicators Guide, Volume Profile*** (see "Volume Profile" on page 1355).

CHAPTER 9

New User Tutorials

This chapter is a collection of tutorials to help you gain familiarity with NeoTicker®.

If you are new to NeoTicker®, the three tutorials you definitely should take a look at are:

Tutorial: Main Window and Sessions (see "Tutorial: Creating a Great Time Chart" on page 105)

Tutorial: Creating a Great Time Chart (on page 105)

Tutorial: Creating a Great Quote Window (on page 125)

In This Chapter

Tutorial: Main Window and Sessions

In this tutorial, we will show you NeoTicker®'s main window and how to save and load sessions in NeoTicker®. We will also show you how to create some charts.

Prerequisite

You should already have NeoTicker® installed and running. For more information, go ***to Getting Star***ted (on page 1).

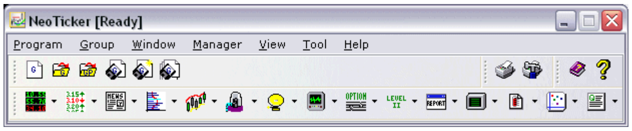
Goal

After you finish this tutorial, you will be able to:

- Understand main window's role
- Open charts and add data series
- Save a session
- Re-open a session after the program restarts

Looking at The Main Window

Once NeoTicker® is running, you will see the main window. The main window is the centralized control of NeoTicker®. The main window looks like:



NeoTicker® is a multiple windows application. Each individual window provides certain functionality to the user. For example, a chart windows provides charting and a quote window provides quote service. You can freely place the windows anywhere on your desktop. The main window lets you coordinate and manage these different windows.

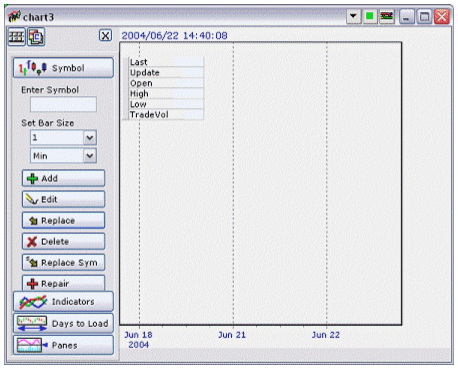
It is possible that the main window becomes hidden beneath another window. In this case, you can click on the caption of any NeoTicker® window, and press the F5 key to bring the main window to the top.

Opening Blank Chart

Press the time chart tool button on the main window.

Time chart is the type of charts most people associated with technical analysis. The horizontal axis represents time and the vertical axis represents a value (e.g. price, volume). NeoTicker® supports other chart types. In general, when the word "chart" is used in this manual, it means time chart.

A new blank chart will be created. An blank chart looks like:



Let's briefly explain what has happened here. After you have pressed the time chart tool button, NeoTicker® creates a new chart.

In NeoTicker®, time chart is a function window. Function windows implement features in NeoTicker®. Examples of function windows are quote windows, time sales windows and chart windows.

Function windows are organized into groups. A group is a collection of function windows. By having multiple groups, you can quickly switch between different function windows. After NeoTicker® is launched, it does not have any groups. Since the chart you have just created needs a group to reside in, NeoTicker® will automatically create a new group for the chart.

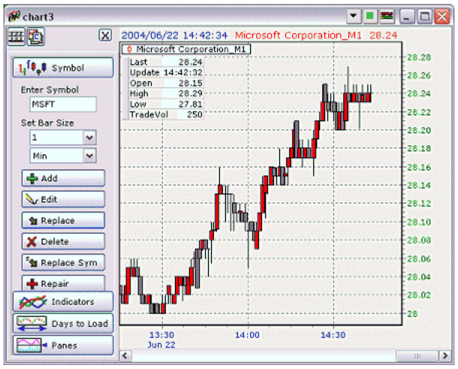
Multiple groups consist of a session in NeoTicker®. In this case, your NeoTicker® session has exactly one group, which contains the blank chart you just created.

Adding Data Series to Chart

A data series is the data from a security. The security can be a stock, index, option or future. There are many ways to add a data series to a chart. In this tutorial, we will use the User Panel to add data. User Panel is the left side panel in the chart.

Type a symbol into the user panel. In this example, we use `MSFT`. Then press the Add button or the ENTER key to add the symbol as a data series to the chart.

After you add the data series, the chart should look like:



2004/06/22 14:42:34 Microsoft Corporation_M1 28.24

Microsoft Corporation_M1

Last 28.24
Update 14:42:32
Open 28.15
High 28.29
Low 27.81
TradeVol 250

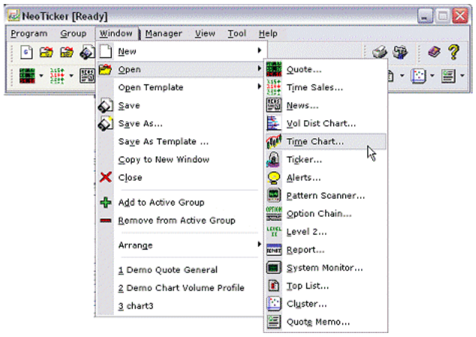
Enter Symbol: MSFT
Set Bar Size: 1, Min

TickQuest does not guarantee the accuracy of the data NeoTicker® found on the Internet or the data you retrieved from your data files/services. You are responsible to ensure and verify the accuracy of such data.

Adding More Charts

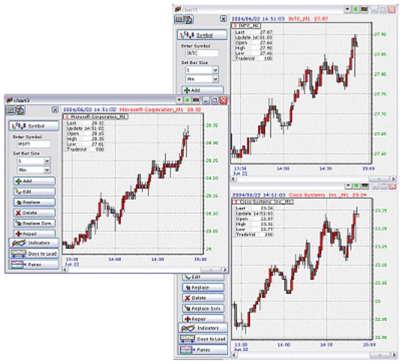
Demo version of NeoTicker® has a limit on the number of windows you can open. If you reach the demo limit, you can skip to the next step in this tutorial.

In this step, we will add more time charts. This time, we will use an alternative method. You can create time chart by choosing **Window>New>Time Chart** from the main window.



If you use the window menu, NeoTicker® will ask you to save a file. This allows you to name the new chart. You can keep the default name the saving dialog gives you and press the **Save** button.

Resize and re-position the new chart to your liking. Add a data series to the new chart. This time we will use the symbol `INTC`. Add a third chart by repeating the procedures above. This time, add the symbol `CSCO`. In the following figure, we have re-positioned the charts again.



Saving Your Work

The easiest way to save your work is to choose **Group>Save All Groups** from the main window.

Although NeoTicker® will prompt you to save your work before you exit the program, it is generally a good practice to save your work after you have made significant changes.

Exiting

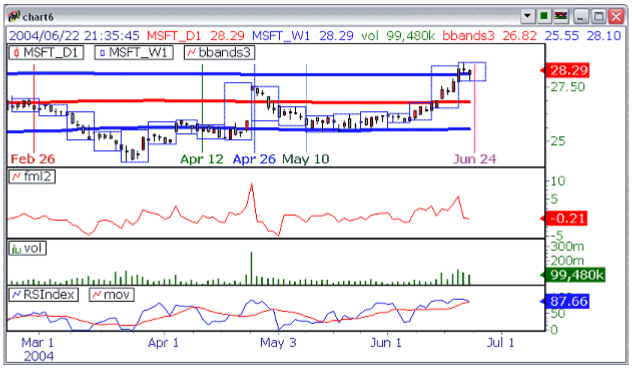
Choose **Program>Exit** from the main window to exit NeoTicker®. NeoTicker® will prompt you to save your work. Press the **Yes** button if you haven't saved your work.

Restarting Previous Session

Click on the NeoTicker® icon to restart NeoTicker®. To continue your previous session, choose **Group>Open Groups from Previous Session**. The three chart windows will be reopened.

Tutorial: Creating a Great Time Chart

In this tutorial, we will create a time chart with many bells and whistles. The finished chart looks like:



The chart created is a daily chart, but the techniques used are equally applicable to real-time charts.

Prerequisite

You know how to create a chart.

Goal

After you finish this tutorial, you will have a good grasp of time chart's features. This tutorial explores the following features:

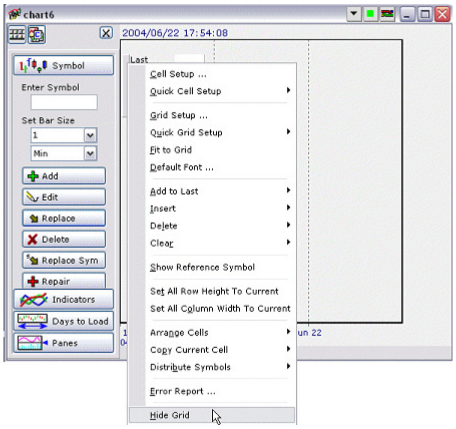
- Multiple time frame analysis
- Indicator-on-indicator
- Chart customization
- Formula
- Float markers
- Drawing tools
- Templates
- Switching symbols

There are many valuable features in time chart that are not covered in this tutorial. We suggest you to read ***Time Chart Operation Guide*** (on page 1011) to explore more usages.

Adding Data Series to Time Chart

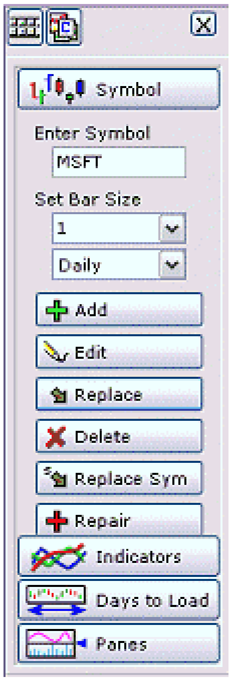
Create a blank chart.

In this tutorial, we are not using Dynamic Grid - the area that displays quotes in the chart. If you see the Dynamic Grid showing, turn it off by right clicking on the Dynamic Grid and choose **Hide**.



Set Bar Size
1
Min

Using User Panel on the left hand side, create a 1-day MSFT data series. User Panel is the area at the left hand side of the chart:



Enter Symbol
MSFT
Set Bar Size
1
Daily

The chart should look like:

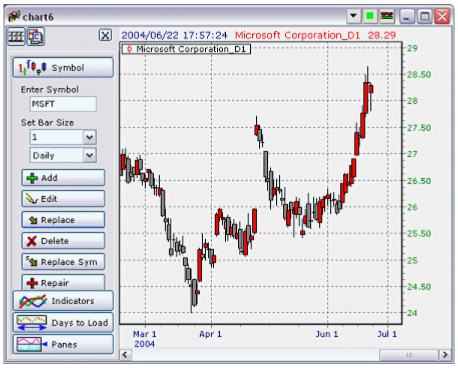


chart6

2004/06/22 17:57:24 Microsoft Corporation_D1 28.29

Enter Symbol: MSFT

Set Bar Size: 1, Daily

Feel free to drag the scroll the chart to see different area of the data series. You can scroll the chart by the scroll bar, or by directly dragging the time axis (see ***Chart Scrolling*** (on page 1020)).

Change the period drop down from **Daily** to **Weekly.** Then press the **Add** button. This will add a 1-week MSFT data series in addition to the 1-day MSFT data series. It is important you use the **Add** button. The chart should look like:

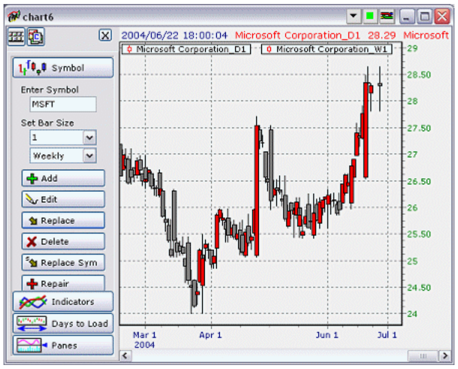


chart6

2004/06/22 18:00:04 Microsoft Corporation_D1 28.29 Microsoft

Enter Symbol: MSFT

Set Bar Size: 1, Weekly

Notice that in the chart, you have 2 legends for MSFT, this indicates there are 2 data series in the chart. The 1-week MSFT pretty much overlap the 1-day MSFT. To make the display better, we will change the visual style of the 1-week MSFT.

First select the 1-week MSFT by clicking on the **MSFT_W1** legend. Notice that the selected data series is highlighted in yellow. This is only a highlight color and does not affect the true color of the selected data series. After the selection, the chart should look like:

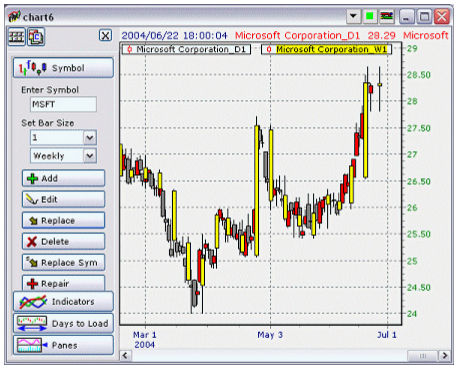


chart6

2004/06/22 18:00:04 Microsoft Corporation_D1 28.29 Microsoft

Microsoft Corporation_D1

Microsoft Corporation_W1

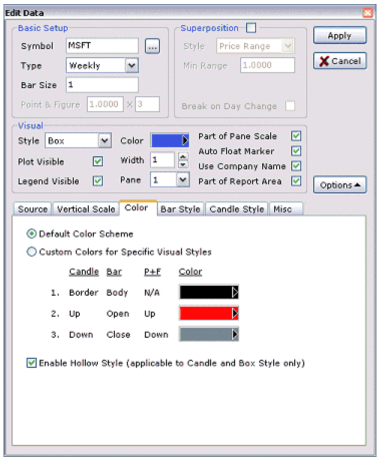
Enter Symbol: MSFT

Set Bar Size: 1, Weekly

Now press the **Edit** button or the space bar on the keyboard to open the edit data dialog. In the edit data dialog, press the **Options** button to review all the options in the edit data dialog. You will need to change:

- **Style** to **Box**
- **Width** to 1
- **Color** to blue
- Check **Enable Hollow Style** under the **Color** tab

All of these options work immediately on the chart. You can see how these options affect the chart as you change the values. You can simply close the dialog after you make the changes.



Edit Data

Basic Setup

Symbol: MSFT

Type: Weekly

Bar Size: 1

Point & Figure: 1.0000 X 3

Superposition □

Style: Price Range

Min Range: 1.0000

Break on Day Change □

Visual

Style: Box

Color: 🟦

Plot Visible ☑

Width: 1

Legend Visible ☑

Pane: 1

Part of Pane Scale ☑

Auto Float Marker ☑

Use Company Name ☑

Part of Report Area ☑

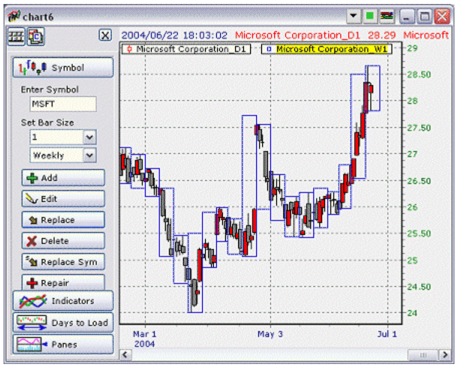
● Default Color Scheme

○ Custom Colors for Specific Visual Styles

	Candle	Bar	P+F	Color
1.	Border	Body	N/A	■
2.	Up	Open	Up	🟥
3.	Down	Close	Down	▨

☑ Enable Hollow Style (applicable to Candle and Box Style only)

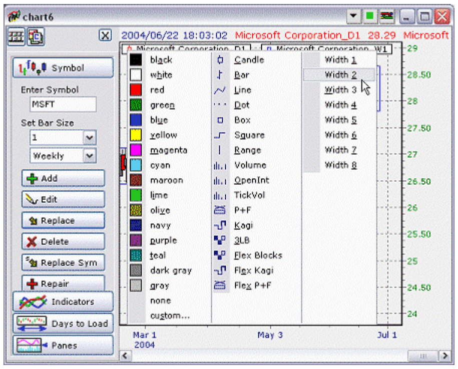
Afterward, the chart should look like:



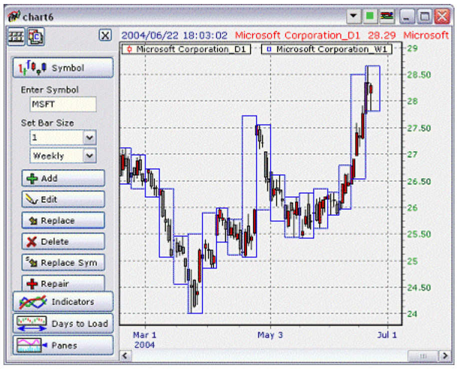
The weekly bars are drawn as boxes on the chart. The boxes create a nice reference when you look at the daily bars.

The 1-day candlesticks are still a bit tight on the chart. What we will do here is to make the candlesticks narrower.

Click on the legend icon of the daily MSFT series. The legend icon is the mini candlestick 🕯 in the legend. A pop up menu will be displayed to let you to quickly change the visual style of the data series. Set the width to 2.



The chart should look like:



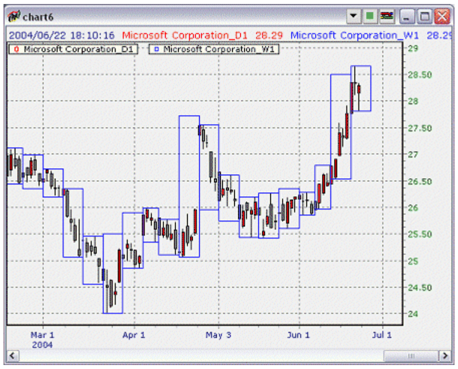
Turning off User Panel for Extra Space

From this point on, Instead of using the User Panel on the right, we will use pop up menus.

Working without the User Panel is essential if you plan to work with a lot of charts and screen space becomes a valuable real estate.

First turn off the User Panel by pressing close button ☒ on the User Panel. The chart will remind you that you can turn on the User Panel again using pop up menus. Simply press **OK.**

With the User Panel off, the chart should look like:



For more information on hiding and showing user panel, see ***Hiding and Showing User Panel*** (on page 1056).

Adding Indicators to Time Chart

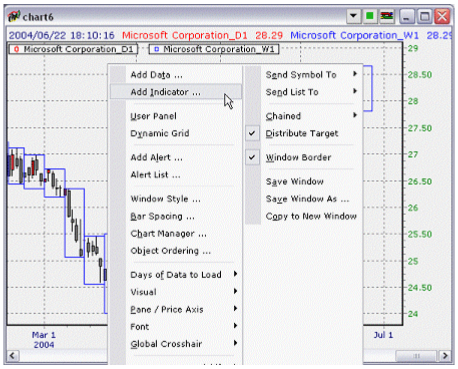
We will be adding indicators to the chart. First, press the ESC key on the keyboard to de-select any data series. Any yellow highlight should be gone.

Time chart allows you to link arbitrary indicator and data series as data source to indicator. The general rule of thumb is:

- If nothing is selected, the first data series in the chart is the data source for the indicator
- If a data series or an indicator is selected, it is the data source for the indicator

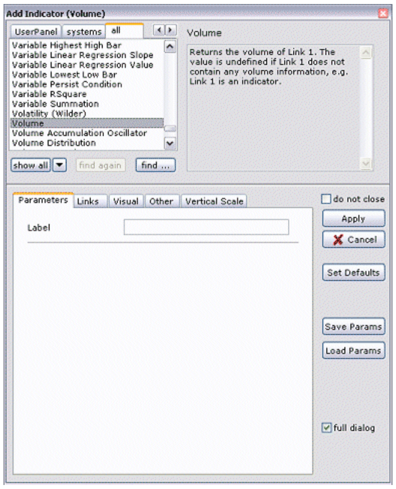
By de-selecting the data series. You basically tell time chart to use the first data series (1-day MSFT) as the default data source.

Right click on the chart, a pop up menu will be opened. Choose **Add Indicator**.



The add indicator dialog is opened. Choose the **Volume** indicator under the **all** tab (press the **show all** button if you can't find it).

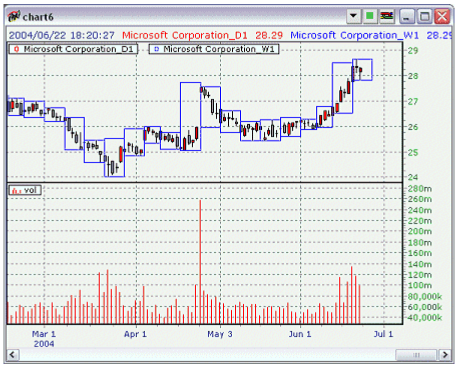
The volume indicator does not take any parameter, so you can just press the **Apply** button to create the indicator.


Add Indicator (Volume)

Volume

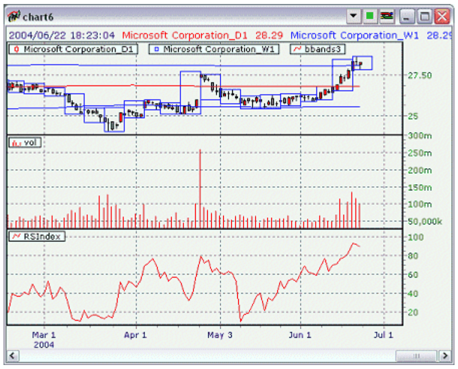
Returns the volume of Link 1. The value is undefined if Link 1 does not contain any volume information, e.g. Link 1 is an indicator.

The chart should look like:



Similarity, add Bollinger Bands 3 (bbands3) and Relative Strength Index (RSI) to the chart. The Bollinger Bands parameters are set to a **period** of 200 and **offset** of 1. The RSI **period** parameter is set to 10.

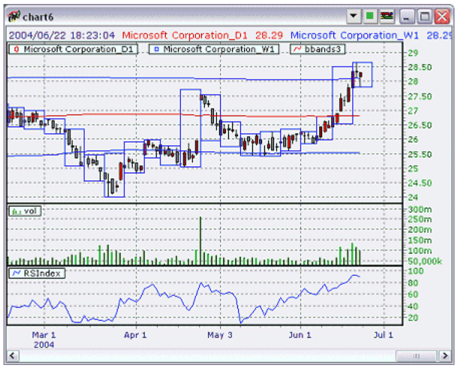
The chart should look like:



Indicators have a long name (e.g. Bollinger Bands 3) and a short name (e.g. bbands3). The long name is more descriptive and is used when you select indicator from the add indicator dialog. Typically, chart displays the indicator in its short name, for example, in the indicator legend.

You can drag the separator between panes (the vertical segments in the chart) to scale the panes to give you more space for the data. For more information of panes, refer to working with panes.

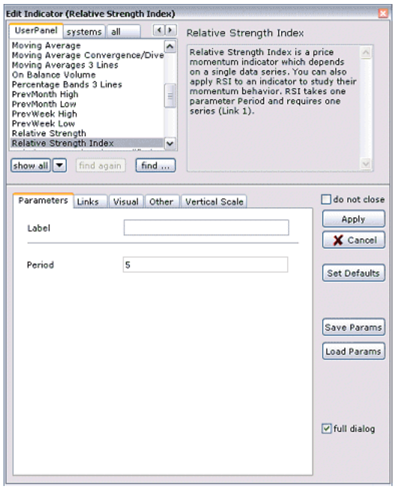
Just like data series, you can also use the legend icon to change the visual appearance of the indicators. After re-spacing and change of visual style, the chart looks like:



If you want to change indicator parameter, you can use the edit indicator dialog. Suppose we want to change the RSI period from 10 to 5. You can either:

- Selected the RSI indicator by clicking on its legend, then press the space bar, or
- Double click on the RSI indicator legend

Simply change the period parameter in the edit indicator dialog to 5 and press the **Apply** button.



Relative Strength Index

Relative Strength Index is a price momentum indicator which depends on a single data series. You can also apply RSI to an indicator to study their momentum behavior. RSI takes one parameter Period and requires one series (Link 1).

Label

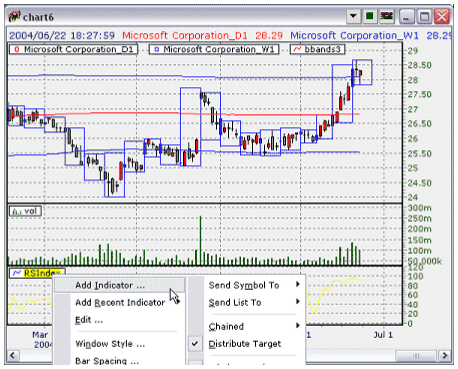
Period 5

Adding Indicator on Indicator

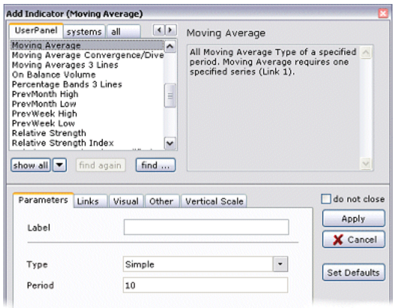
You can apply an indicator on another indicator. We will use the capability to apply a 10-period simple moving average to the 5 period RSI indicator from the last step.

To tell time chart which is the source indicator, click on the legend of the RSI indicator to select it. The RSI indicator will be highlighted in yellow.

Right click on the chart to open the pop up menu, and choose **Add Indicator**.



Set up a 10-period simple moving average as shown in the dialog below, and press the **Apply** button to add the indicator (if you can't find the moving average indicator, use the **find** button).



Moving Average

All Moving Average Type of a specified period. Moving Average requires one specified series (Link 1).

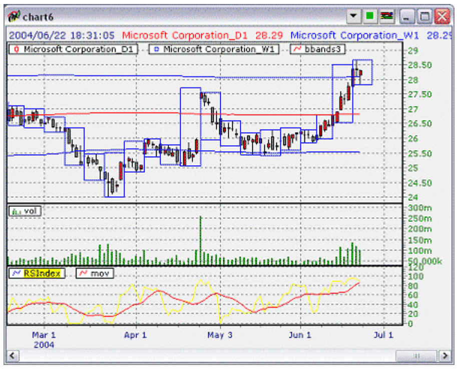
☐ do not close

Label

Type: Simple

Period: 10

The chart should look like:



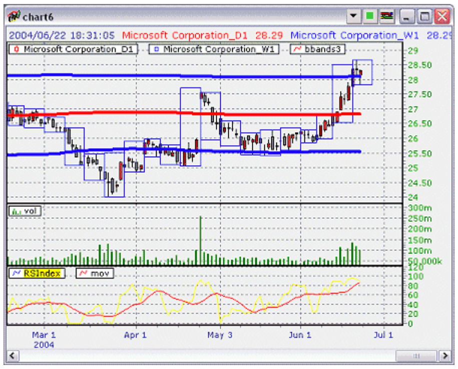
Note that the 10-period moving average is applied on the RSI indicator.

You can create arbitrary levels of indicators. Multiple input indicators such as addition and multiplication are available. You can use technique to perform arbitrary complex calculations without a single line of programming or formula.

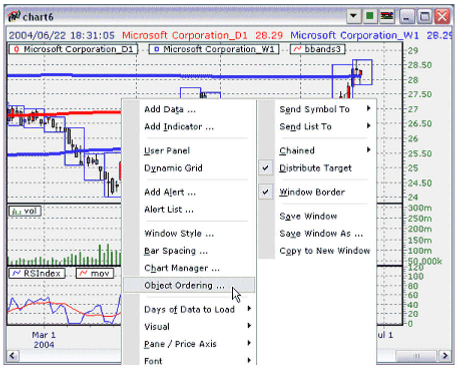
Changing Object Display Order

You can change the display ordering of data series and indicators.

Change the width of the Bollinger Bands to 4. This will thicken the lines and they start to obscure the display of data series.



Press the ESC key to make sure all items are de-selected. Right click on the pop up menu and choose **Object Ordering**.



The chart object ordering dialog will be open. The dialog shows the order the data series and indicators are drawn on the chart. It shows that the Bollinger Bands (**bbands3**) are drawn on top of the data series MSFT weekly and daily series.

Click on **bbands3**. Then press the **Down** button until **bbands3** is at the bottom.

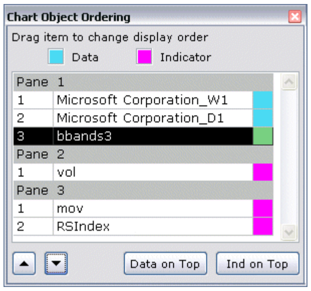


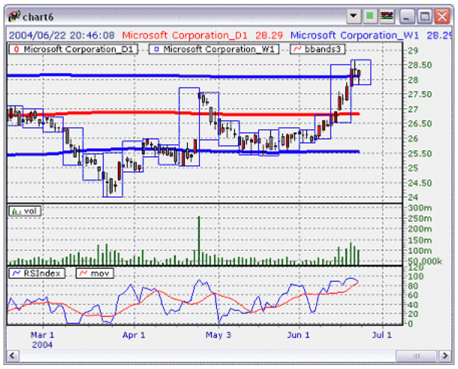
Chart Object Ordering

Drag item to change display order

🟦 Data 🟪 Indicator

Pane 1		
1	Microsoft Corporation_W1	🟦
2	Microsoft Corporation_D1	🟦
3	bbands3	🟩
Pane 2		
1	vol	🟪
Pane 3		
1	mov	🟪
2	RSIndex	🟪

The chart looks like:



Notice that the Bollinger Bands are no longer blocking the data series.

Customizing the Look

We will customize the look of the chart. Our goal is to make a white background chart with no grid. Also, we want to hide the scroll bar in the time axis because we know we can scroll the chart by directly dragging the time axis. We want to make the font a bit larger so the chart is easier to read. Finally we want to display the chart using ticker symbol, not company name.

All of this can be accomplished by using the chart manager. Right click on the chart to open the pop up menu and choose **Chart Manager**. The chart manager will be opened. Press the **Visual 1** tab.

Perform the following changes:

- Change **Show Grid** to **None**
- Change the **Background** color to white
- Check off **Time Scroll Bar**
- Press the **Medium** button under **Fonts**

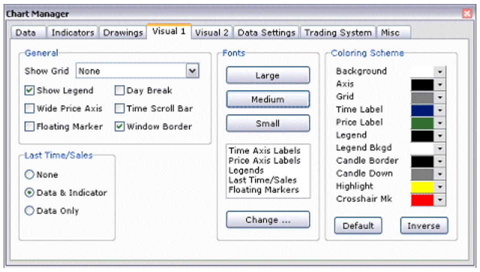


Chart Manager

Data | Indicators | Drawings | Visual 1 | Visual 2 | Data Settings | Trading System | Misc

General

Show Grid: None

☑ Show Legend ☐ Day Break

☐ Wide Price Axis ☐ Time Scroll Bar

☐ Floating Marker ☑ Window Border

Last Time/Sales

○ None

● Data & Indicator

○ Data Only

Fonts

Large

Medium

Small

Time Axis Labels
Price Axis Labels
Legends
Last Time/Sales
Floating Markers

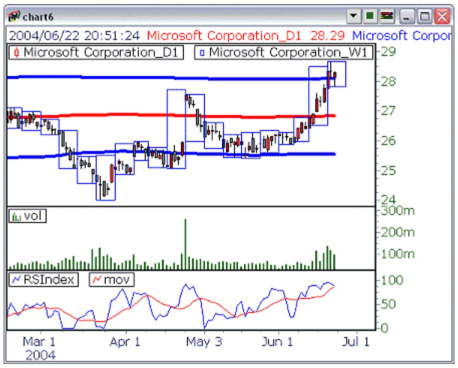
Change ...

Coloring Scheme

Background	⬜
Axis	⬛
Grid	⬛
Time Label	🟦
Price Label	🟩
Legend	⬛
Legend Bkgd	⬜
Candle Border	⬛
Candle Down	⬛
Highlight	🟨
Crosshair Mk	🟥

Default Inverse

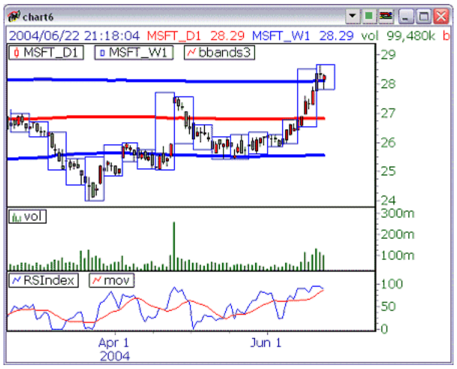
The chart will change as you work on the chart manager. The resulting chart looks like:



You can close the chart manager once you complete the changes.

When NeoTicker® connects to some data vendors, company names of stocks are provided. NeoTicker® will use the company name in charts. Sometimes the names can get quite long and distract the chart. You can choose to use the ticker symbol instead.

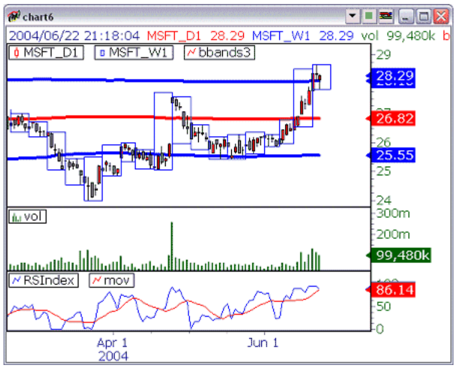
Right click on MSFT's daily series to open pop up menu, and choose **Edit**. In the editor, besides **Legend**, choose **Sym**. Do the same with MSFT's week series.



Enabling Float Markers

Float markers are markers on the price axis. The marker will mark the value of a data series or an indicator as you scroll the chart, or when real-time data is received in the chart.

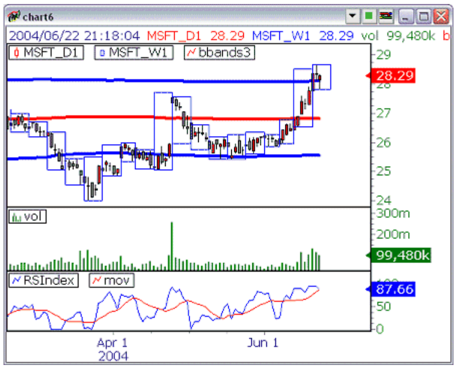
Turn on the float markers by choosing **Visual>Floating Markers** from the pop up menu.



Now we want to turn off some of the float markers we don't need. Select the following items one by one, and for each item, right click on the chart to open the pop up menu for the item, and use the pop up menu to toggle off **Auto Float Marker**:

- The 1-week MSFT data series
- The Bollinger Bands indicator (bbands3)
- The moving average on RSI index (mov)

The resulting chart looks like:

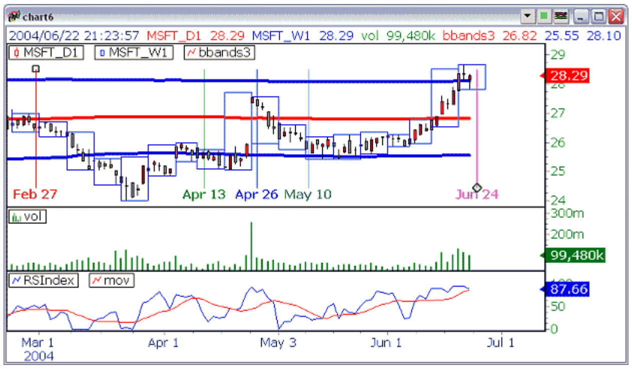


Drawing Fibonacci Lines

To do drawing on the chart, you need to use the tool bar. You can open the tool bar by pressing the F4 key.

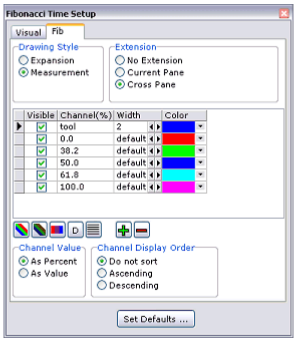
Press the Fibonacci Time button on the tool bar. The chart will change its cursor to indicate that you can draw Fibonacci time lines.

Press on a point on the left side of the chart, then drag to the right side. A set of Fibonacci time lines will be drawn. We've scaled up the chart a bit so everything will fit. The chart looks like (the Fibonacci lines may looks somewhat different):



Notice that the drawing tool has handles on them (square handle and diamond handle on the Feb 27 and Jul 24 lines). When handles are shown on a drawing tool, the drawing tool is selected. You can drag the handles to modify the tool.

Right click on the chart to open the pop up menu for the drawing tool, choose **Edit**. The tool's editor will be opened.



Fibonacci Time Setup

Visual | Fib

Drawing Style
○ Expansion
● Measurement

Extension
○ No Extension
○ Current Pane
● Cross Pane

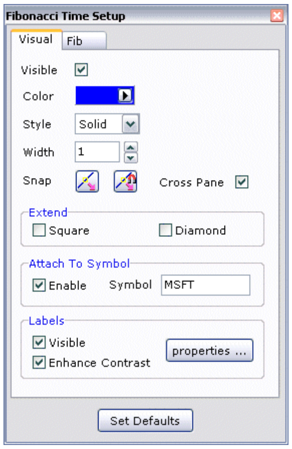
	Visible	Channel(%)	Width	Color
▶	☑	tool	2	🟦
	☑	0.0	default	🟥
	☑	38.2	default	🟩
	☑	50.0	default	🟦
	☑	61.8	default	🟦
	☑	100.0	default	🟪

Channel Value
● As Percent
○ As Value

Channel Display Order
● Do not sort
○ Ascending
○ Descending

Here you can edit various attributes of the Fibonacci lines. Note that we've switch the drawing style to **Measurement**. When you are making changes to the drawing tool, the drawing tool in the chart will change immediately. There is no need to press any okay or apply buttons to confirm the changes.

Now press the **Visual** tab in the editor.



Fibonacci Time Setup

Visual | Fib

Visible ☑
Color 🟦
Style Solid
Width 1
Snap
Cross Pane ☑

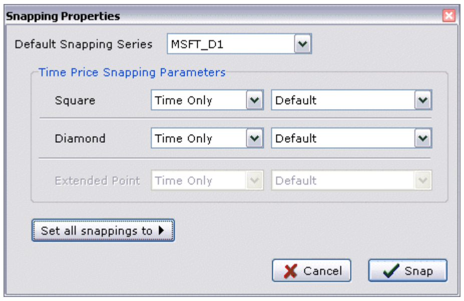
Extend
☐ Square
☐ Diamond

Attach To Symbol
☑ Enable
Symbol MSFT

Labels
☑ Visible
☑ Enhance Contrast

Press the snap property button on the editor. The Snapping Properties editor is opened.

Press the **Set all snappings to** button to open a menu and choose **Time Only** from the pop up menu. Press the **Snap** button. This will ensures the square and diamond handles are always snap to exact time of a bar.


Snapping Properties

Default Snapping Series: MSFT_D1

Time Price Snapping Parameters

Square	Time Only	Default
Diamond	Time Only	Default
Extended Point	Time Only	Default

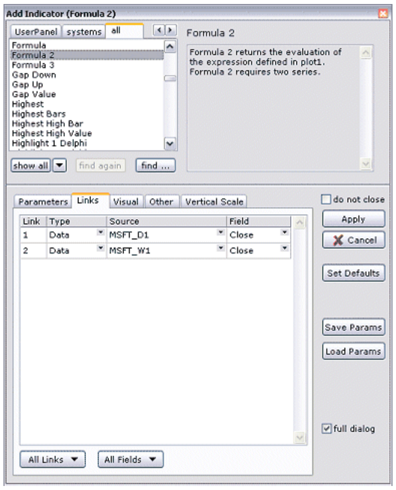
Calculating Daily % Gain with Formula

We will illustrate how to use formula to calculate the daily percentage gain on last week close price. The basic formula is:

```
(daily price – last week price) / last week price * 100
```

First, press ESC to de-select everything in the chart, then choose **Add Indicator** from the pop up menu. Choose the **Formula 2** indicator. This is an indicator that allows you to add a formula calculation to the chart. Formula 2 can link to two data source.

First make sure the indicator is linked to the correct data source. The links are set up under the **Link** tab. Make sure link 1 is set to MSFT_D1's Close and link 2 is set to MSFT_W1's Close.


Add Indicator (Formula 2)

Formula 2

Formula 2 returns the evaluation of the expression defined in plot1. Formula 2 requires two series.

Link	Type	Source	Field
1	Data	MSFT_D1	Close
2	Data	MSFT_W1	Close

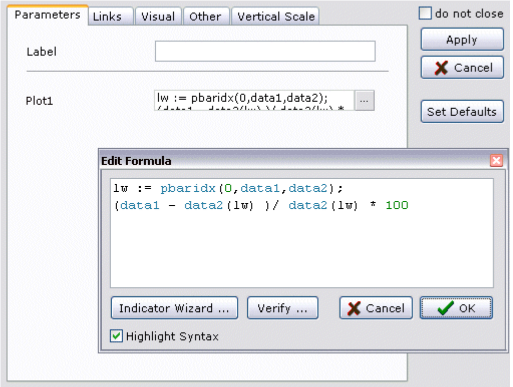
□ do not close

☑ full dialog

Press the **Parameter** tab and enter the following formula into the **plot 1** parameter (you can press the ... button to help you edit) :

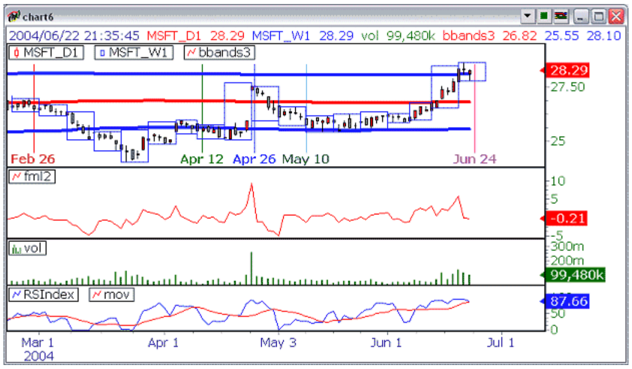
```
lw := pbaridx(0,data1,data2);
(data1 - data2(lw) )/ data2(lw) * 100
```

The `lw` variable resolves to the index of the last week bar. Because you link `data1` and `data2` to MSFT D1 and MSFT D2 close respectively, the above formula translates to what we originally intended.



```
lw := pbaridx(0,data1,data2);
(data1 - data2(lw) )/ data2(lw) * 100
```

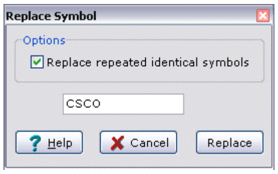
Press the **Apply** button. The daily percentage gain against weekly close is plot as **fml2** in the chart.



Changing Symbols Quickly

It is possible to change symbol by editing the data series, but you need you edit twice for the chart in this tutorial. Instead, you can use the **F9** shortcut to replace the symbol in a chart. **F9** has the option to replace all identical looking symbols.

Press **F9**, and type `CSCO` into the dialog, and press the **Replace** button.



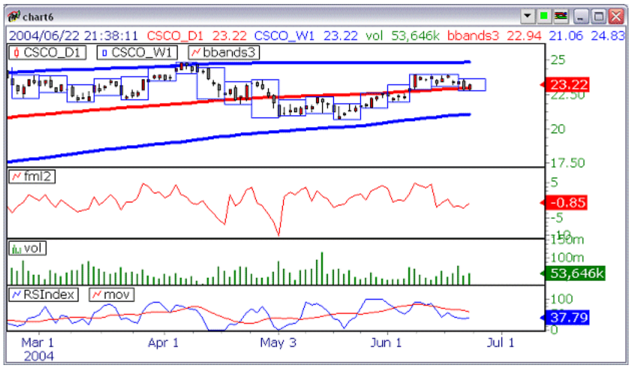
Replace Symbol

Options

☑ Replace repeated identical symbols

CSCO

The chart would look like:



Creating a Template

After working on the chart for so long, you want to re-use the chart set up later. The way to do this is to save the chart as a template.

From the main window choose: **Window>Save As Template**.

You will be prompt to enter a name for the template.

When you want to open a template, choose from the main window **Window>Open Template**. When you open a chart template, you will be prompt for the symbol.

If you save the template using the special name **default,** the template will be used to open all new charts.

Tutorial: Creating a Great Quote Window

NeoTicker®'s quote window is a powerful tool. Rule-based coloring, indicator and formula evaluation, automatic ranking, row filtering are all available to help you locate information quickly.

In this tutorial, we will show you how to create a quote window that perform real-time scanning.

Prerequisite

No prerequisite.

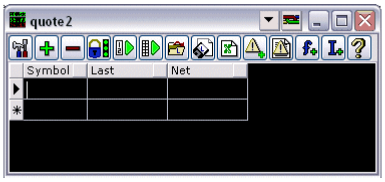
Goal

After you finish this tutorial, you will be able to:

- Create a quote window
- Apply rule-based coloring to the quote window
- Perform row filtering
- Perform automatic ranking

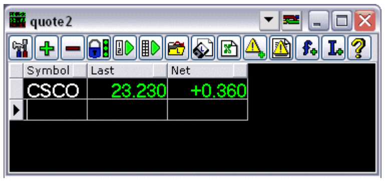
Creating a Quote Window and Importing a Symbol List

First, create a quote window by pressing the quote window tool button in the main window.



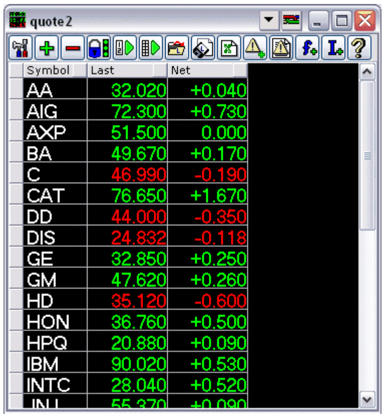
Symbol	Last	Net

By default, the quote window displays the symbol, the last traded price and the net change from previous closing price. Type CSCO in the symbol column, press ENTER and the quote window gives you the last traded price and net change for CSCO (your quote will differ from what is shown here):



Symbol	Last	Net
CSCO	23.230	+0.360

Instead of entering the symbols manually, we will import the symbol from a symbol list file. Press the import symbol list button . You will be prompt for importing a symbol list, choose one of the Dow Jones index. Resize the quote window to show a few more lines. This symbol list will replace what is inside the quote window.



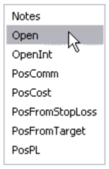
Symbol	Last	Net
AA	32.020	+0.040
AIG	72.300	+0.730
AXP	51.500	0.000
BA	49.670	+0.170
C	46.990	-0.190
CAT	76.650	+1.670
DD	44.000	-0.350
DIS	24.832	-0.118
GE	32.850	+0.250
GM	47.620	+0.260
HD	35.120	-0.600
HON	36.760	+0.500
HPQ	20.880	+0.090
IBM	90.020	+0.530
INTC	28.040	+0.520
INU	55.370	+0.090

It may take a few seconds before the quote window is populated. The exact time is data vendor dependent.

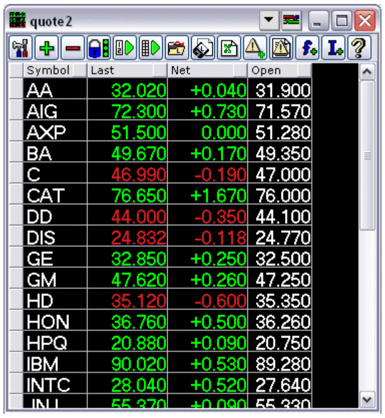
Important: This example uses 30 symbols (Done Jones 30). If this exceeds your data feed subscription limit, some of the quotes may never show up.

Adding Fields

You can use the pop up menu to add additional fields to the quote window. Simply right click on the the quote window to open the pop up window, and choose **Insert Column>Open**. Note that the column will be inserted to the place you click.

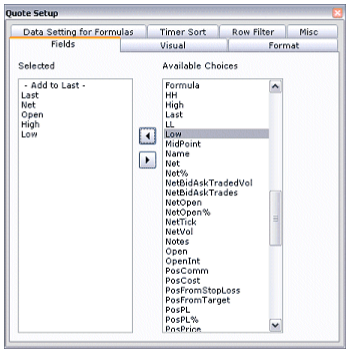


The quote window will look like:


quote2

Symbol	Last	Net	Open
AA	32.020	+0.040	31.900
AIG	72.300	+0.730	71.570
AXP	51.500	0.000	51.280
BA	49.670	+0.170	49.350
C	46.990	-0.190	47.000
CAT	76.650	+1.670	76.000
DD	44.000	-0.350	44.100
DIS	24.832	-0.118	24.770
GE	32.850	+0.250	32.500
GM	47.620	+0.260	47.250
HD	35.120	-0.600	35.350
HON	36.760	+0.500	36.260
HPQ	20.880	+0.090	20.750
IBM	90.020	+0.530	89.280
INTC	28.040	+0.520	27.640
INJ	55.370	+0.090	55.330

We want to add a few more fields. Instead of using the pop up menu, we will use the set up dialog for quote window. Press the set up button . When the quote window set up dialog shows up, you can add more fields under the **Fields** tab. Select **High** in the right hand side, and press the ◀ button. Similarly, select **Low** in the right hand side and press the button ◀.


Quote Setup

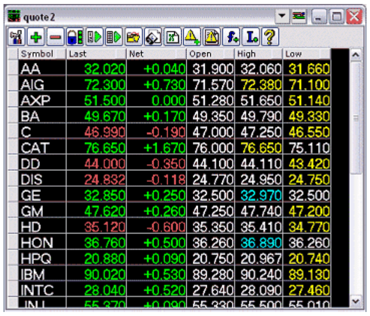
Selected

- Add to Last -
Last
Net
Open
High
Low

Available Choices

Formula
HH
High
Last
LL
Low
MidPoint
Name
Net
Net%
NetBidAskTradedVol
NetBidAskTrades
NetOpen
NetOpen%
NetTick
NetVol
Notes
Open
OpenInt
PosComm
PosCost
PosFromStopLoss
PosFromTarget
PosPL
PosPL%
PosPrice

Resize the quote window to show all fields:


quote2

Symbol	Last	Net	Open	High	Low
AA	32.020	+0.040	31.900	32.060	31.660
AIG	72.300	+0.730	71.570	72.380	71.100
AXP	51.500	0.000	51.280	51.650	51.140
BA	49.670	+0.170	49.350	49.790	49.330
C	46.990	-0.190	47.000	47.250	46.550
CAT	76.650	+1.670	76.000	76.650	75.110
DD	44.000	-0.350	44.100	44.110	43.420
DIS	24.832	-0.118	24.770	24.950	24.750
GE	32.850	+0.250	32.500	32.970	32.500
GM	47.620	+0.260	47.250	47.740	47.200
HD	35.120	-0.600	35.350	35.410	34.770
HON	36.760	+0.500	36.260	36.890	36.260
HPQ	20.880	+0.090	20.750	20.967	20.740
IBM	90.020	+0.530	89.280	90.240	89.130
INTC	28.040	+0.520	27.640	28.090	27.460
INJ	55.370	+0.090	55.330	55.500	55.010

Filtering Rows

You can control which rows to display based on a formula criteria.

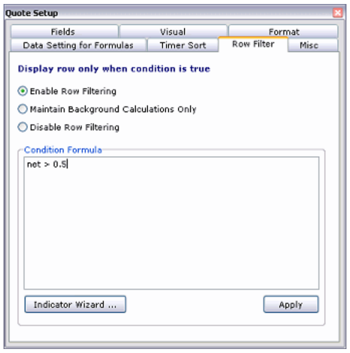
In the quote window set up, press the **Row Filter** tab.

Enter the following formula:

```
Net > 0.5
```

This formula tells the quote window to display rows where the net change is larger than 0.5.

Check the **Enable Row Filter** option and press the **Apply** button:



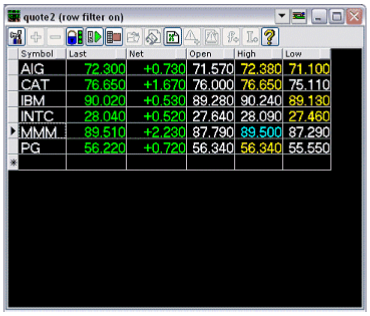
Display row only when condition is true

● Enable Row Filtering
○ Maintain Background Calculations Only
○ Disable Row Filtering

Condition Formula

```
net > 0.5
```

The quote window will look like:



Symbol	Last	Net	Open	High	Low
AIG	72.300	+0.730	71.570	72.380	71.100
CAT	76.650	+1.670	76.000	76.650	75.110
IBM	90.020	+0.530	89.280	90.240	89.130
INTC	28.040	+0.520	27.640	28.090	27.460
MMM	89.510	+2.230	87.790	89.500	87.290
PG	56.220	+0.720	56.340	56.340	55.550

Setting up Rule-based Coloring for a Column

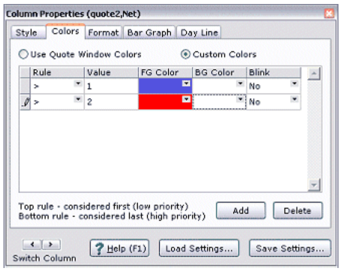
You may notice that the quote window has a default coloring scheme. You can set up your own rule-based coloring.

Double click on any cell under the **Net** column.

The Column Property dialog will be opened for changing the property for the **Net** column. Click on the **Color** tab if the dialog is not already under it.

Change the **Rule** to **>**, **Value** to **1**, **FG Color** to blue, **BG Color** to white.

Press the **Add** button. This will add a new rule. In the new rule, change the **Rule** to **>**, **Value** to **2**, **FG Color** to red, **BG Color** to white.



Column Properties (quote2,Net)

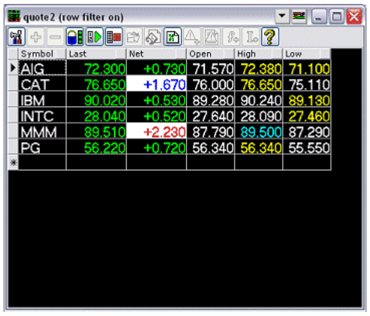
○ Use Quote Window Colors ● Custom Colors

Rule	Value	FG Color	BG Color	Blink
>	1	🟦		No
>	2	🟥		No

Top rule - considered first (low priority)
Bottom rule - considered last (high priority)

By default **Use Quote Window Colors** is selected. **Custom Colors** will be selected automatically after you make changes in the rules. You can turn off the custom rules any time without deleting them by switching back to the **Use Quote Window Colors** option.

Here is what the quote window looks like:



quote2 (row filter on)

Symbol	Last	Net	Open	High	Low
AIG	72.300	+0.730	71.570	72.380	71.100
CAT	76.650	+1.670	76.000	76.650	75.110
IBM	90.020	+0.530	89.280	90.240	89.130
INTC	28.040	+0.520	27.640	28.090	27.460
MMM	89.510	+2.230	87.790	89.500	87.290
PG	56.220	+0.720	56.340	56.340	55.550

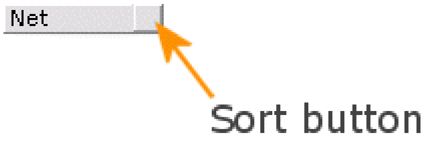
When color rules are applied, the first rule is considered first. Then second rule, and so on, i.e. lower rule has higher priority. For example, even 2.23 is larger than 1, the second rule has higher priority and the color will be red.

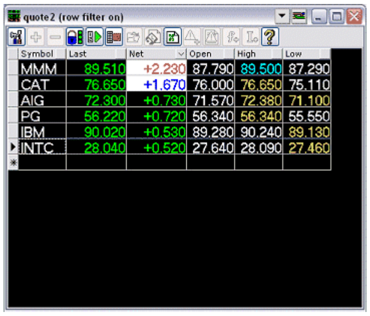
Automatically Ranking

Quote windows let you automatically sort a specific column by using the timer sort feature. When the timer sort feature is on, the specified column is sorted automatically. Combined with row filtering and formulas, you can perform majority of scanning operations using quote window in real time.

NeoTicker® also provides a Pattern Scanner for large scanning jobs and Scan Workshop for chart-based scanning.

Start timer sort first by choosing from the pop up menu **Start Timer Sort** or press the timer sort button. Once sorting started, click on the sort button on the column header to decide which column to sort. Press the sort button again changes from ascending to descending, and vice verse.





Symbol	Last	Net	Open	High	Low
MMM	89.510	+2.230	87.790	89.500	87.290
CAT	76.650	+1.670	76.000	76.650	75.110
AIG	72.300	+0.730	71.570	72.380	71.100
PG	56.220	+0.720	56.340	56.340	55.550
IBM	90.020	+0.530	89.280	90.240	89.130
INTC	28.040	+0.520	27.640	28.090	27.460

When timer sort is on, most of the function of the quote window is disabled to avoid data conflicts. Stop timer sort by pressing the stop timer sort button before editing the quote window again.

Timer sort has a default interval of 10 seconds. That means a specific column is sorted every 10 seconds. To change the default interval, open the set up window of the quote window by choosing **Setup** from the pop up window. Under the **Timer Sort** tab, you can specify the interval between each sort. Press the **Apply** button or **Apply and Start** button to apply the interval.

Considerations for Using Quote Window for Scanning

There are some considerations when you use quote window for real time scanning.

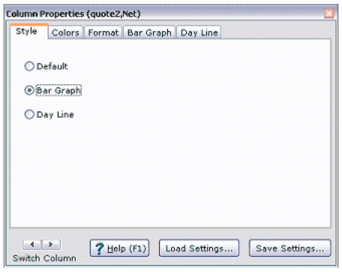
- Symbol Limit. You data vendor may have a symbol limit imposed on you. Quote window cannot calculate properly when the number of symbols exceeds the data vendor imposed limit. If you need to scan more than the imposed limit, use the pattern scanner instead. The pattern scanner does not "hold" on to symbols like the quote window, effectively giving you an arbitrary large symbol list to scan.
- System Performance. Typically for a Pentium 4 system with 512M RAM, you should restrict yourself to below several hundred symbols for acceptable performance. If you need to scan more, you should use the pattern scanner, which is less CPU and RAM demanding.
- Timer Sort Interval. You can set up timer sort interval in the quote window set up window. To improve system performance, you can set a larger timer sort interval than 10 seconds.

Bar Graph

Bar Graph in quote window can greatly improve its quality.

First, turn off auto ranking and row filtering by pressing the ▣ and ▣ buttons. Your quote window should display quotes for all symbols now.

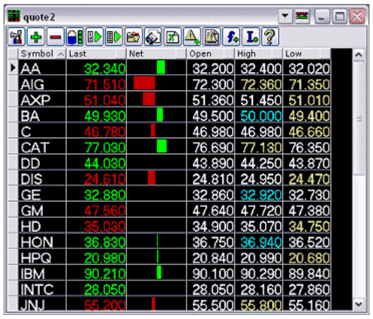
Double click under the **Net** column to open Column Properties dialog. Press the **Style** tab. Select **Bar Graph**.



Column Properties (quote2,Net)

○ Default
● Bar Graph
○ Day Line

Your quote window will now display the **Net** column as a Bar Graph.



Symbol	Last	Net	Open	High	Low
AA	32.340	🟩	32.200	32.400	32.020
AIG	71.510	🟥	72.300	72.360	71.350
AXP	51.040	🟥	51.360	51.450	51.010
BA	49.930	🟩	49.500	50.000	49.400
C	46.780	🟥	46.980	46.980	46.660
CAT	77.030	🟩	76.690	77.130	76.350
DD	44.030		43.890	44.250	43.870
DIS	24.610	🟥	24.810	24.950	24.470
GE	32.880		32.860	32.920	32.730
GM	47.560		47.640	47.720	47.380
HD	35.030		34.900	35.070	34.750
HON	36.830	🟩	36.750	36.940	36.520
HPQ	20.980	🟩	20.840	20.990	20.680
IBM	90.210	🟩	90.100	90.290	89.840
INTC	28.050		28.050	28.160	27.860
JNJ	55.200	🟥	55.500	55.800	55.160

Tutorial: Drawing Tools in Time Chart

This tutorial will walk you through many features of drawing tools in time chart, using trend channel as the example. Trend channel is interesting because it has three handles: an extended handle in addition to the square and diamond handles.

Prerequisite

You know how to create a chart and use the tool bar to select drawing tools.

Goal

After you finish this tutorial, you will have a good grasp of drawing tools in time chart. You will be able to edit channel ratios, labels and colors as well as learning how to save a default drawing tool.

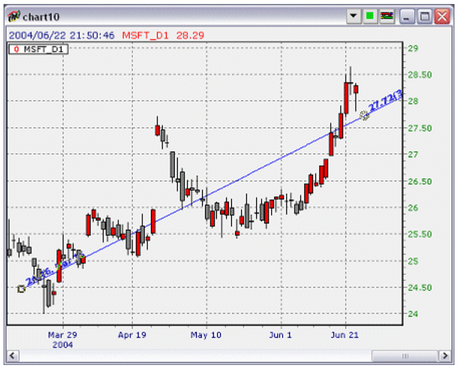
Creating Trend Channels

In this tutorial, we do not need Dynamic Grid or User Panel, turn those off.

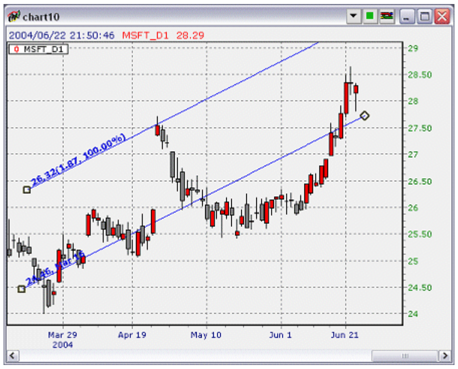
To begin this tutorial, create a new chart and add a data series.

To draw trend channels, press the trend channel tool button on the tool bar. In the chart,

- Press the mouse button on a point where you want the trend channels to begin. Do not release mouse button
- Drag to a point where you want the trend channels to end. Release mouse button. After you have released the mouse button, a trend line will be drawn. This trend line forms a base for the trend channels.



As you release the mouse button, the tool also creates a line parallel to the trend line. Move the mouse to another position on the chart, then press the left mouse button to anchor the parallel line.

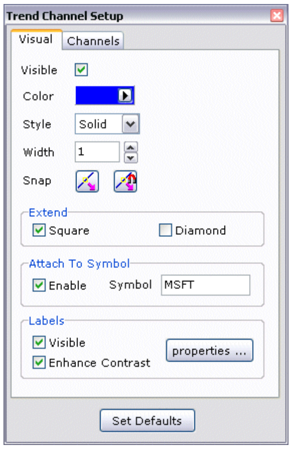


The trend line and the parallel line are the reference for trend channels.

Editing Trend Channels

Double click on the trend channel. This opens an editor to edit the properties of the trend channels.

Under the **Visual** tab, **Extend** box, click on the **Square** check box.


Trend Channel Setup

Visual | Channels

Visible ☑

Color

Style: Solid

Width: 1

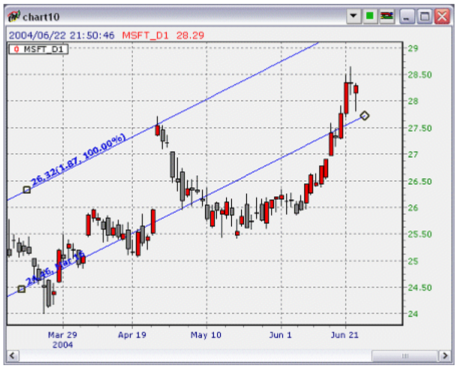
Snap

Extend
☑ Square ☐ Diamond

Attach To Symbol
☑ Enable Symbol: MSFT

Labels
☑ Visible
☑ Enhance Contrast

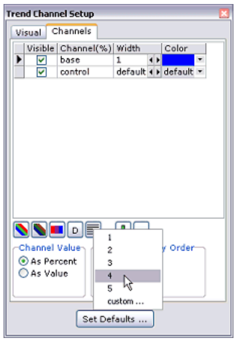
Trend channels have two ends, the square ends and the diamond ends. Clicking the **Square** check box will extend the square ends of the trend channels. After you click, the chart should look like:



Notice that the lines at the square end are extended.

Adding More Channels

Press the **Channels** tab in the trend channel window. This tab lets you add, edit or remove channels. Click on the make partition button ☰ and choose 4.



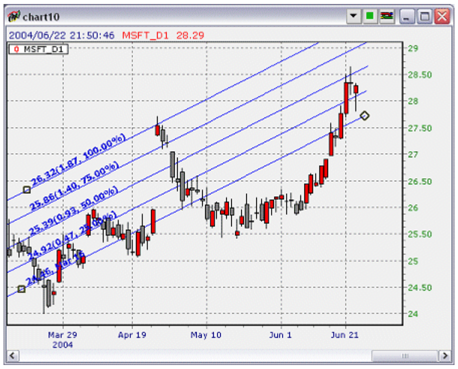
Trend Channel Setup

Visual | Channels

	Visible	Channel(%)	Width	Color
▶	☑	base	1	🟦
	☑	control	default	default

Channel Value
● As Percent
○ As Value

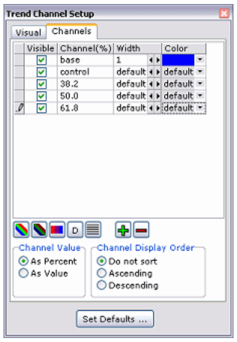
This will partition the trend channels into 4 parts by adding three trend lines to the trend channel. The chart should look like:



Editing Channels

In the Trend Channel editor:

- Click on the **25%** value to select it, type `38.2,` press Enter key
- Click on the **75%** value to select it, type `61.8,` press Enter key



Trend Channel Setup

Visual | Channels

Visible	Channel(%)	Width	Color
☑	base	1	🟦
☑	control	default	default
☑	38.2	default	default
☑	50.0	default	default
☑	61.8	default	default

Channel Value
● As Percent
○ As Value

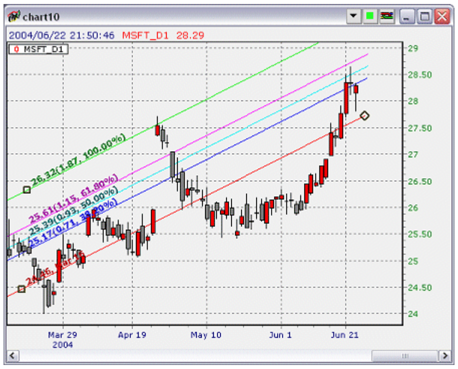
Channel Display Order
● Do not sort
○ Ascending
○ Descending

Set Defaults ...

This will replace the trend channel ratio with Fibonacci ratio.

Now press the rainbow button . This will assign a different color to each trend channel.

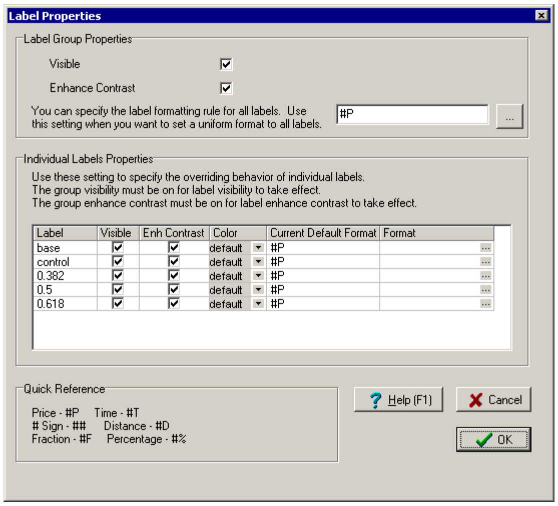
The chart should look like:



Notice the labels on the trend channels are displaying a lot of information: price, distance between channels and channel ratio. We can customize the labels so only the price is displayed.

Click on the **Visual** tab on the editor and press the **properties** button under **labels**.

The label properties dialog is opened. Simply enter #P in the field under **Label Group Properties**, and press the **OK** button.



Label Properties

Label Group Properties

Visible ☑

Enhance Contrast ☑

You can specify the label formatting rule for all labels. Use this setting when you want to set a uniform format to all labels. #P

Individual Labels Properties

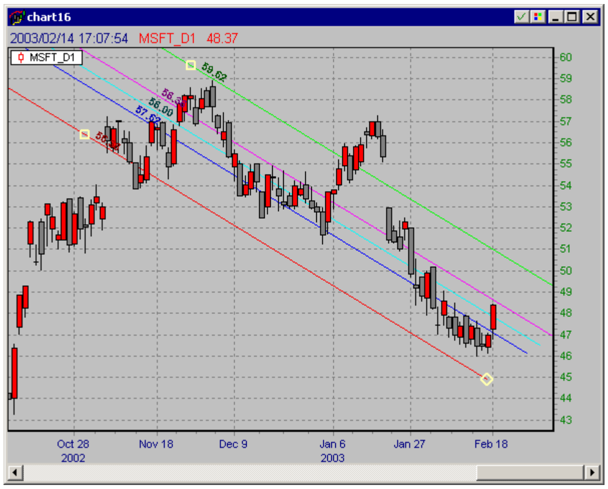
Use these setting to specify the overriding behavior of individual labels.
The group visibility must be on for label visibility to take effect.
The group enhance contrast must be on for label enhance contrast to take effect.

Label	Visible	Enh Contrast	Color	Current Default Format	Format
base	☑	☑	default	#P	...
control	☑	☑	default	#P	...
0.382	☑	☑	default	#P	...
0.5	☑	☑	default	#P	...
0.618	☑	☑	default	#P	...

Quick Reference

Price - #P Time - #T
Sign - ## Distance - #D
Fraction - #F Percentage - #%

The chart should look like:



Saving Drawing Tool to Template

To re-use the settings of a drawing tool, you can save the drawing tool into a template. There are five templates for each type of drawing tool.

Drawing tool templates are set on the tool bar. By default, when you click on a drawing tool, template 1 is chosen. We will save the trend channel we created to template 3.

On the trend channel editor, press the **Set Defaults** button, then choose **Template 3** in the pop up menu.

If you create another trend channel and if you choose template 3 by press the button , the trend channel will be drawn using the style you have saved.

Drawing Tools Keep on Drawing

After you drew something with a drawing tool, you have to select the drawing tool on the tool bar to draw again.

If you prefer to keep on drawing until manually turning off the drawing. You can configure the tool bar for this behavior.

Choose **Manager>Tool Bar** from the main window.

Turn on the option **Keep the drawing tool so you can use it again** option. Press the **OK** button.

After you turn on this option, you can keep on drawing using the same drawing tool until you press the pointer cursor ↖ or crosshair cursor button ✛ on the tool bar. You can also use the TAB key on the keyboard to toggle away from the drawing tool.

Tutorial: Quote Window Formula 1

One of the most powerful feature of NeoTicker® is the ability to calculate formula in quote window. The formula is evaluated for each symbol and can contain indicator evaluation.

Quote window formula is surprising easy to use. These is no special set up required. Simply add a formula column to a quote window and enter the formula.

In this tutorial, we will go through the steps to calculate the difference between last price and its moving average in ratio. The result is a good candidate for scanning for a stock that is greatly above or below its moving average.

We will also briefly touch on quote window formatting. So not only you will know how to calculate what you want, you can display the result nicely too.

Prerequisite

You should know how to create a quote window and add a symbol list to it.

Goal

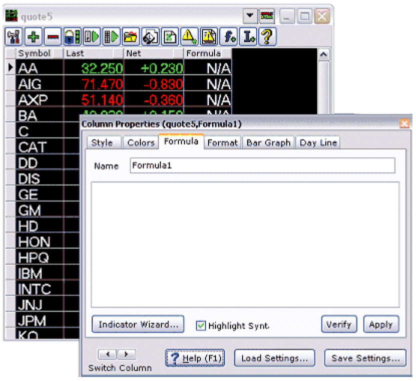
After you finish this tutorial, you will be able to:

- Create a quote window formula
- Use the indicator wizard to help you enter indicator in formula
- Format a column

Creating a Formula Column

Create a quote window and import the Dow Jones 30 symbol list.

Press the formula column button 𝑓+ to create a formula column. After you press the button, a formula column will be created in the quote window. At the same time, the column properties dialog for the formula column is opened, and the **Formula** tab is selected for you.



Symbol	Last	Net	Formula
AA	32.250	+0.230	N/A
AIG	71.470	-0.830	N/A
AXP	51.140	-0.360	N/A
BA			
C			
CAT			
DD			
DIS			
GE			
GM			
HD			
HON			
HPQ			
IBM			
INTC			
JNJ			
JPM			
KO			

Name Formula1

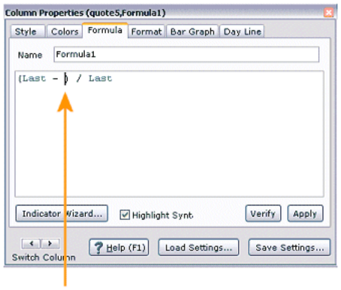
You can name your formula with a unique name that you can later refer to it from within another formula column.

Type `My_Average` to the **Name** field.

Type the following formula into the formula area (don't press the **Apply** button yet!):

```
(Last - ) / Last
```

This is not a complete formula yet. We will enter a moving average indicator using the indicator wizard. Position the cursor between the minus sign and the close bracket.

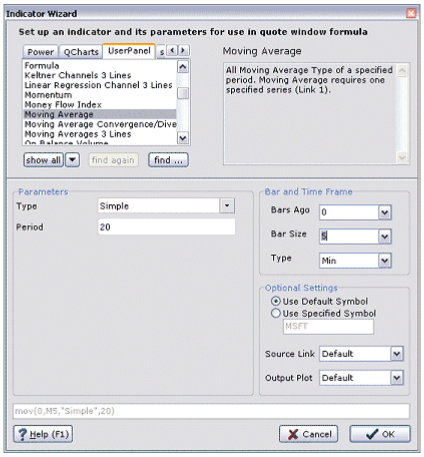


Name Formula1

```
(Last - ) / Last
```

Position cursor here

Then press the **Indicator Wizard** button to open the Indicator Wizard.



Indicator Wizard

Set up an indicator and its parameters for use in quote window formula

Formula
Keltner Channels 3 Lines
Linear Regression Channel 3 Lines
Momentum
Money Flow Index
Moving Average
Moving Average Convergence/Dive
Moving Averages 3 Lines
On Balance Volume

Moving Average

All Moving Average Type of a specified period. Moving Average requires one specified series (Link 1).

Parameters
Type: Simple
Period: 20

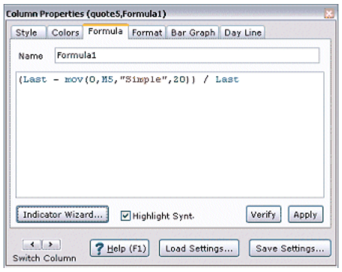
Bar and Time Frame
Bars Ago: 0
Bar Size: 5
Type: Min

Optional Settings
● Use Default Symbol
○ Use Specified Symbol
MSFT
Source Link: Default
Output Plot: Default

```
mov(0,M5,"Simple",20)
```

Indicator Wizard helps you specifies the parameter for an indicator used in formula. This tool simplifies you task a lot because you do not have to memorize which indicator uses what parameters.

Locate the moving average indicator and fill in a **Period** of 20 and a **Bar Size** of 5. Press the **OK** button. Indicator Wizard will specify a 20 period 5-minute moving average indicator for you in the column properties dialog:



Column Properties (quote5,Formula1)

Name: Formula1

```
(Last - mov(0,M5,"Simple",20)) / Last
```

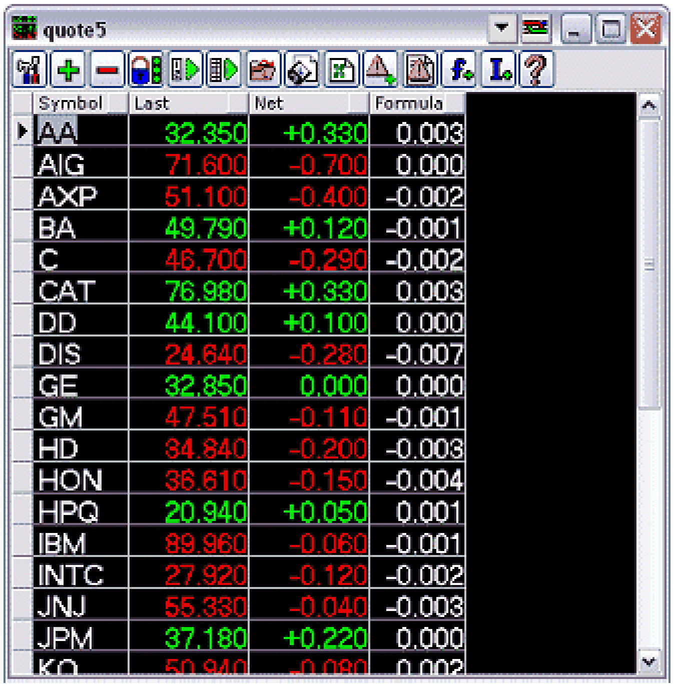
☑ Highlight Synt.

Now press the **Apply** button on the column properties dialog.

The first time you use indicator in a formula can be slow. Depending on the data vendor and indicator, it will take several seconds to several minutes to initially filled the quote window. Afterward, data is cached in NeoTicker® and speed will be greatly improved. In terms of data load, each formula with indicator is equivalent to a chart. For this tutorial using Dow Jones 30, the amount of data requested is similar to 30 charts.

Indicator calculation in quote window formula works on data in the RAM Cache. You can monitor the data status in the RAM Cache. Refer to ***Monitor Data Status in RAM Cache*** (on page 468).

Tutorial: Turbo Charging Data Loading Speed (on page 205) describes techniques on performance tuning, including techniques of using RAM Cache.



Symbol	Last	Net	Formula
AA	32.350	+0.330	0.003
AIG	71.600	-0.700	0.000
AXP	51.100	-0.400	-0.002
BA	49.790	+0.120	-0.001
C	46.700	-0.290	-0.002
CAT	76.980	+0.330	0.003
DD	44.100	+0.100	0.000
DIS	24.640	-0.280	-0.007
GE	32.850	0.000	0.000
GM	47.510	-0.110	-0.001
HD	34.840	-0.200	-0.003
HON	36.610	-0.150	-0.004
HPQ	20.940	+0.050	0.001
IBM	89.960	-0.060	-0.001
INTC	27.920	-0.120	-0.002
JNJ	55.330	-0.040	-0.003
JPM	37.180	+0.220	0.000
KO	50.940	-0.080	0.002

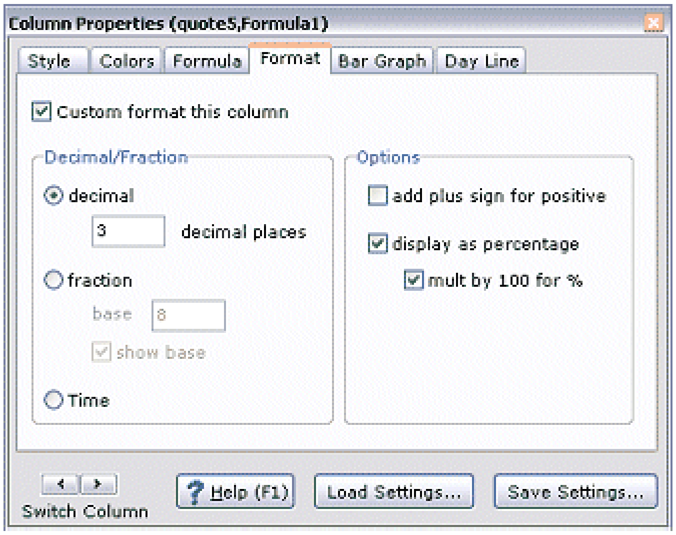
You have just defined a custom formula calculating the difference between the last price of a symbol and its 20 period 5-min bar simple moving average in ratio.

You can sort by this column. You can apply column coloring to it. You can even reuse the result in another formula column with the name `My_Average`.

You can also define an alert based on this column just like referencing any other columns within the alert.

Formatting the Column

The values under the My_Average column can be formatted as percentage. A final nice touch for this tutorial is to change the formatting of the column to percentage. Double click on any cell under the My_Average column to open the column property dialog. Press the **Format** tab.



Column Properties (quote5,Formula1)

☑ Custom format this column

Decimal/Fraction

● decimal

3 decimal places

○ fraction

base 8

☑ show base

○ Time

Options

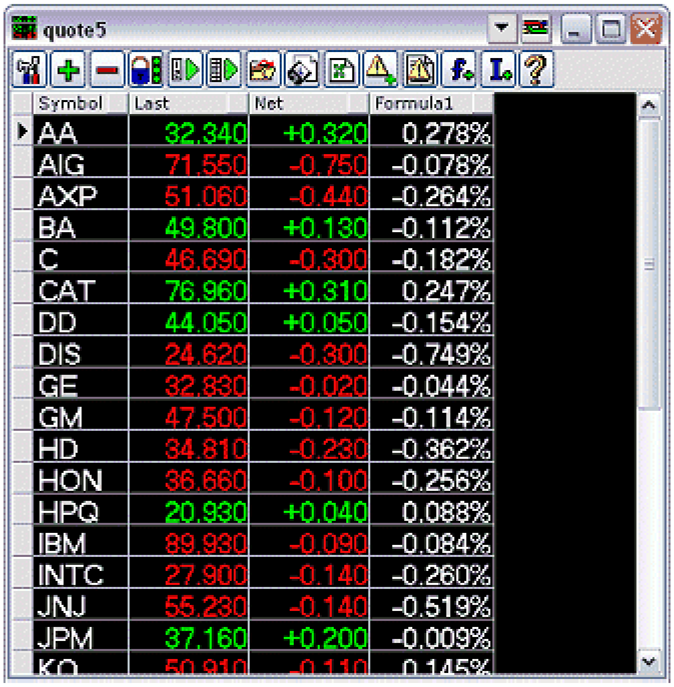
□ add plus sign for positive

☑ display as percentage

☑ mult by 100 for %

Check **Custom format this column**. Check the **displayed as percentage** option. Make sure the **mult by 100 for %** option is checked.

The My_Average column will then be formatted in percentage. You can resize the column by dragging its edge to give more space for display.



Symbol	Last	Net	Formula1
AA	32.340	+0.320	0.278%
AIG	71.550	-0.750	-0.078%
AXP	51.060	-0.440	-0.264%
BA	49.800	+0.130	-0.112%
C	46.690	-0.300	-0.182%
CAT	76.960	+0.310	0.247%
DD	44.050	+0.050	-0.154%
DIS	24.620	-0.300	-0.749%
GE	32.830	-0.020	-0.044%
GM	47.500	-0.120	-0.114%
HD	34.810	-0.230	-0.362%
HON	36.660	-0.100	-0.256%
HPQ	20.930	+0.040	0.088%
IBM	89.930	-0.090	-0.084%
INTC	27.900	-0.140	-0.260%
JNJ	55.230	-0.140	-0.519%
JPM	37.160	+0.200	-0.009%
KO	50.910	-0.110	0.145%

Tutorial: Quote Window Formula 2

In this tutorial, we will combine quote window formula with other quote window features such as rule-based coloring and alert.

Prerequisite

You should know how to create a quote window and add a symbol list to it.

Goal

After you finish this tutorial, you will be able to:

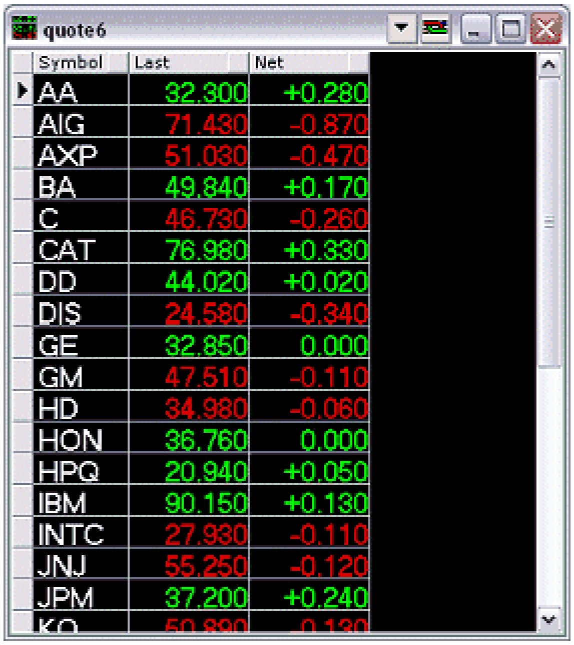
- Combine the use of quote window formula with other quote window features

Adding Formula Column

Open a new quote window. Right click on the quote window to open pop up menu and choose **Hide Tool Buttons**. This will hide the tool buttons from the quote window and give you more space to view quotes. Without the tool buttons, you will rely mostly on pop up menu to work on quote window.

Right click on the quote window and choose **Import Symbol List**. Select one of the Dow Jones 30 symbol list.

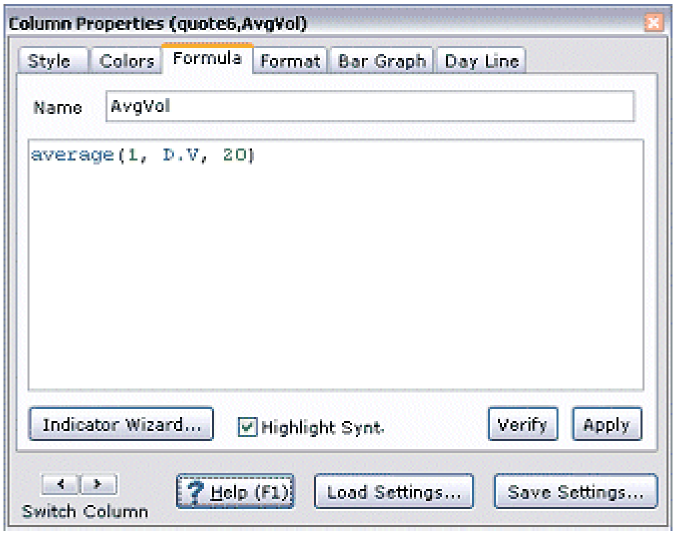
At this stage, your quote window will look like:



Symbol	Last	Net
AA	32.300	+0.280
AIG	71.430	-0.870
AXP	51.030	-0.470
BA	49.840	+0.170
C	46.730	-0.260
CAT	76.980	+0.330
DD	44.020	+0.020
DIS	24.580	-0.340
GE	32.850	0.000
GM	47.510	-0.110
HD	34.980	-0.060
HON	36.760	0.000
HPQ	20.940	+0.050
IBM	90.150	+0.130
INTC	27.930	-0.110
JNJ	55.250	-0.120
JPM	37.200	+0.240
KO	50.890	-0.130

Right click on the quote window and select **Insert Column>Formula**. You will see a new column with the name **Formula1** added to the quote window. Double click on the **Formula1** column, the column properties window will open at the **Formula** tab. Change the formula name to `AvgVol`. In the formula area type in the formula:

```
average(1, D.V, 20)
```

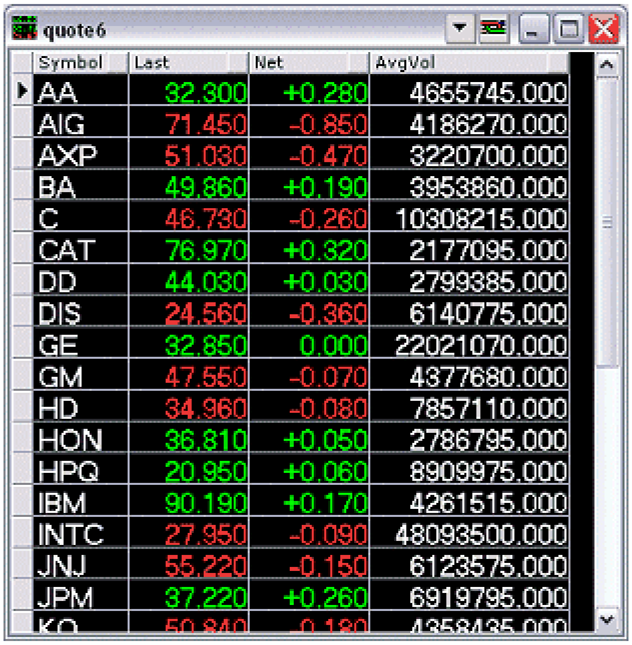


Name AvgVol

```
average(1, D.V, 20)
```

☑ Highlight Synt.

Press the **Apply** button. This will give you the average volume of the pass 20 days. Resize the quote window and the AvgVol column to display the values. At this stage, your quote window will look like:



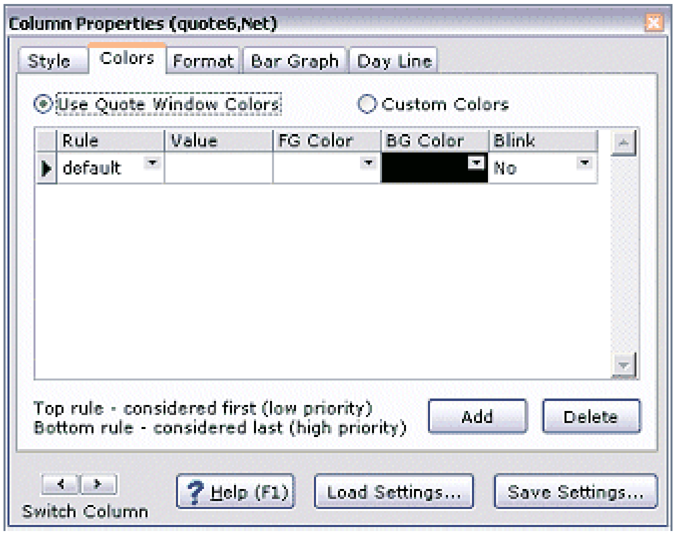
Symbol	Last	Net	AvgVol
AA	32.300	+0.280	4655745.000
AIG	71.450	-0.850	4186270.000
AXP	51.030	-0.470	3220700.000
BA	49.860	+0.190	3953860.000
C	46.730	-0.260	10308215.000
CAT	76.970	+0.320	2177095.000
DD	44.030	+0.030	2799385.000
DIS	24.560	-0.360	6140775.000
GE	32.850	0.000	22021070.000
GM	47.550	-0.070	4377680.000
HD	34.960	-0.080	7857110.000
HON	36.810	+0.050	2786795.000
HPQ	20.950	+0.060	8909975.000
IBM	90.190	+0.170	4261515.000
INTC	27.950	-0.090	48093500.000
JNJ	55.220	-0.150	6123575.000
JPM	37.220	+0.260	6919795.000
KO	50.840	-0.180	4358435.000

Adding Color Properties

You can add different visual effects to any columns based on criteria you define. This will help you easily spot any movements you want to track with any indicators or price values.

In this tutorial we will set the **Net** column background color to white and text color to blue when **Net** is greater than 0.5.

To set column color properties, double click under the **Net** column to open Column Properties dialog. Press the **Colors** tab.



Column Properties (quote6,Net)

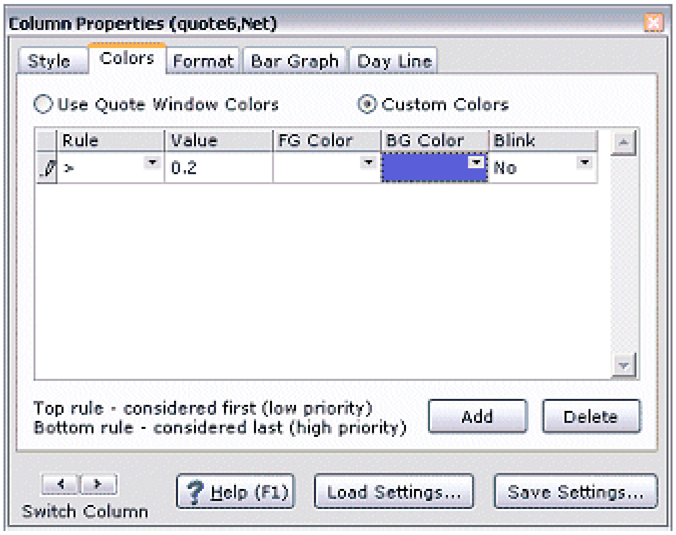
● Use Quote Window Colors ○ Custom Colors

Rule	Value	FG Color	BG Color	Blink
default			■	No

Top rule - considered first (low priority)
Bottom rule - considered last (high priority)

Click on the **Custom Colors** radio button. This will force quote window to use the color setting you define instead of the default.

Now change the **Rule** field to **>**, type `0.2` into the **Value** field and change the **BG Color** to blue.

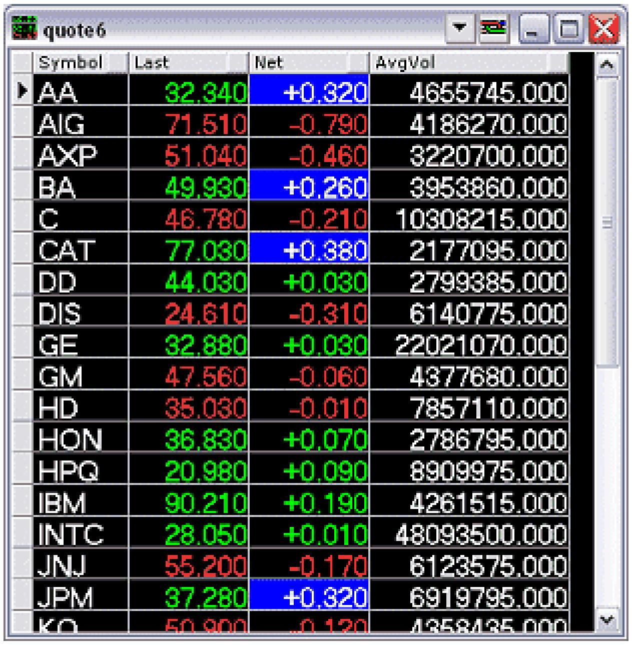

Column Properties (quote6,Net)

○ Use Quote Window Colors ● Custom Colors

Rule	Value	FG Color	BG Color	Blink
>	0.2		🟦	No

Top rule - considered first (low priority)
Bottom rule - considered last (high priority)

You will immediately see the results taking effect in the column.

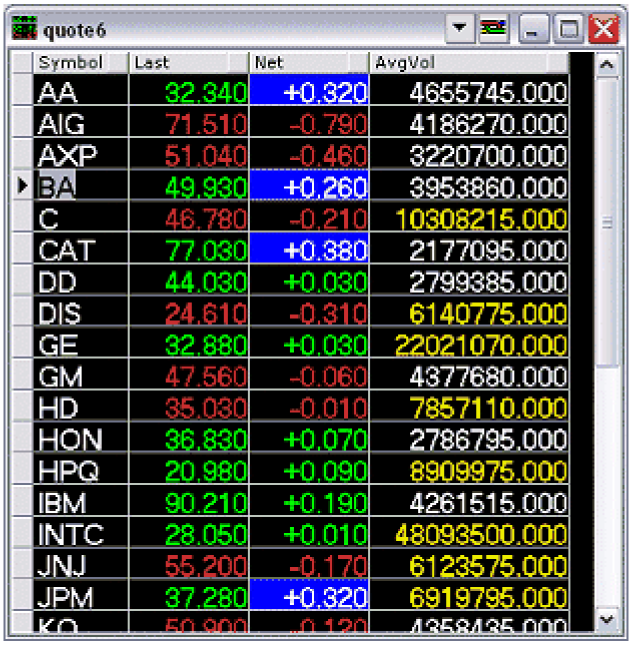

quote6

Symbol	Last	Net	AvgVol
AA	32.340	+0.320	4655745.000
AIG	71.510	-0.790	4186270.000
AXP	51.040	-0.460	3220700.000
BA	49.930	+0.260	3953860.000
C	46.780	-0.210	10308215.000
CAT	77.030	+0.380	2177095.000
DD	44.030	+0.030	2799385.000
DIS	24.610	-0.310	6140775.000
GE	32.880	+0.030	22021070.000
GM	47.560	-0.060	4377680.000
HD	35.030	-0.010	7857110.000
HON	36.830	+0.070	2786795.000
HPQ	20.980	+0.090	8909975.000
IBM	90.210	+0.190	4261515.000
INTC	28.050	+0.010	48093500.000
JNJ	55.200	-0.170	6123575.000
JPM	37.280	+0.320	6919795.000
KO	50.900	-0.120	4358435.000

Now you know how to set column color effects, you can make the **AvgVol** column brink in yellow when it exceed 6,000,000. Double click on the **AvgVol** column. In the column properties window, press the **Colors** tab, and set the parameters as follows:

- choose **Custom Colors**
- change **Rule** to **>**
- **Value** to `6000000`
- **FG Color** to yellow
- **BG Color** to black
- **Blink** to **Yes**

The quote window will look like:



Symbol	Last	Net	AvgVol
AA	32.340	+0.320	4655745.000
AIG	71.510	-0.790	4186270.000
AXP	51.040	-0.460	3220700.000
BA	49.930	+0.260	3953860.000
C	46.780	-0.210	10308215.000
CAT	77.030	+0.380	2177095.000
DD	44.030	+0.030	2799385.000
DIS	24.610	-0.310	6140775.000
GE	32.880	+0.030	22021070.000
GM	47.560	-0.060	4377680.000
HD	35.030	-0.010	7857110.000
HON	36.830	+0.070	2786795.000
HPQ	20.980	+0.090	8909975.000
IBM	90.210	+0.190	4261515.000
INTC	28.050	+0.010	48093500.000
JNJ	55.200	-0.170	6123575.000
JPM	37.280	+0.320	6919795.000
KO	50.900	-0.120	4358435.000

Adding Custom Formula

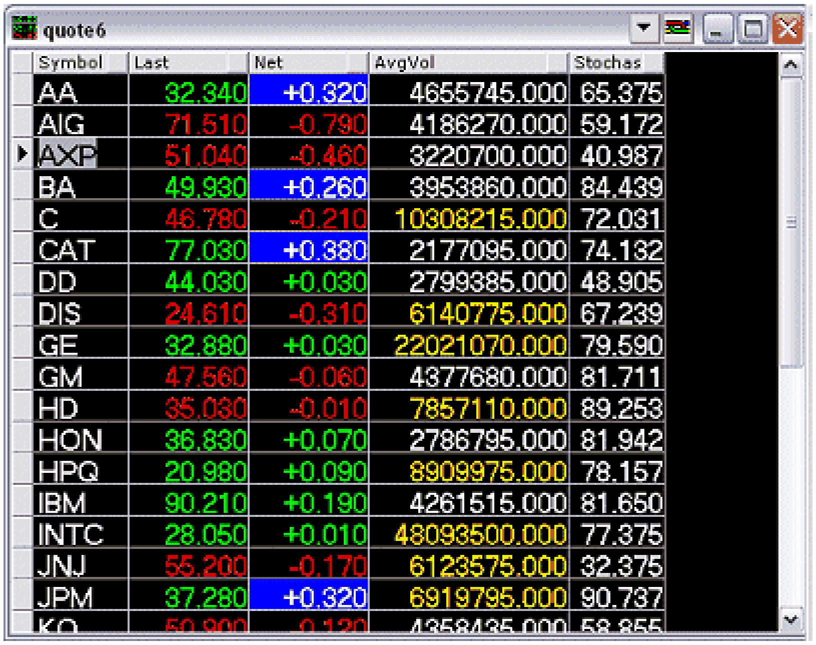
Besides simply calling an indicator from within a formula column, you can also do calculation and evaluation in formula columns. In this tutorial we are going to add an evaluation formula column to evaluate if the stochastic value is greater than 80 or less than 20.

Insert a new formula column into the quote window. Double click on the new column to add formula. First we will have to add a new column to calculate the stochastic indicator, type in:

```
fastd(0, M5, 5, 3)
```

This formula tells quote window to give you the current value based on 5-minute bar data with parameters period equals 5 and smoothing equals 3.

Enter `stochastic` into the **Name** field for this column. Press the **Apply** button to see the result.



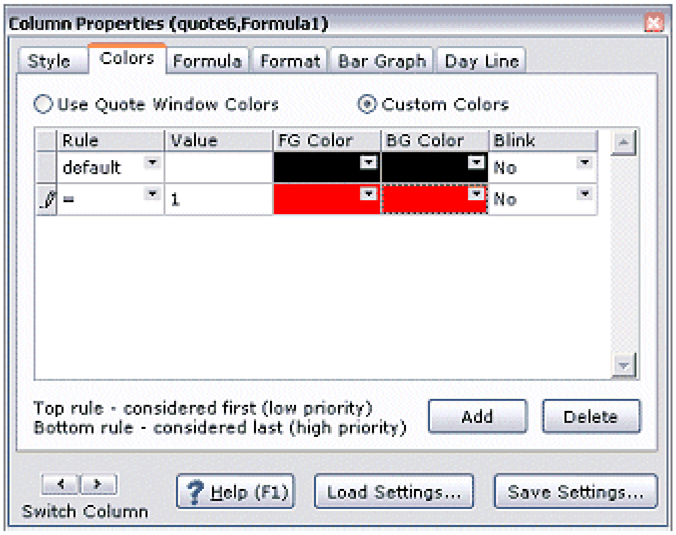
Symbol	Last	Net	AvgVol	Stochas
AA	32.340	+0.320	4655745.000	65.375
AIG	71.510	-0.790	4186270.000	59.172
AXP	51.040	-0.460	3220700.000	40.987
BA	49.930	+0.260	3953860.000	84.439
C	46.780	-0.210	10308215.000	72.031
CAT	77.030	+0.380	2177095.000	74.132
DD	44.030	+0.030	2799385.000	48.905
DIS	24.610	-0.310	6140775.000	67.239
GE	32.880	+0.030	22021070.000	79.590
GM	47.560	-0.060	4377680.000	81.711
HD	35.030	-0.010	7857110.000	89.253
HON	36.830	+0.070	2786795.000	81.942
HPQ	20.980	+0.090	8909975.000	78.157
IBM	90.210	+0.190	4261515.000	81.650
INTC	28.050	+0.010	48093500.000	77.375
JNJ	55.200	-0.170	6123575.000	32.375
JPM	37.280	+0.320	6919795.000	90.737
KO	50.900	-0.120	4358435.000	58.855

Next we are going to add the evaluation column. Insert a new formula column; double click on the new column to open Column Properties dialog to edit formula window, type in the line:

```
if((stochastic > 80) or (stochastic < 20), 1, 0)
```

This formula will return the value 1 if the stochastic column is greater than 80 or less than 20.

In Column Properties dialog, press the **Colors** tab. Change the default rules to: **Rule** - default, **FG Color** - black, **BG Color** - black. Add a new rule, set the new rule to: **Rule** - =, **Value** - 1, **FG Color** - red, **BG Color** - red.



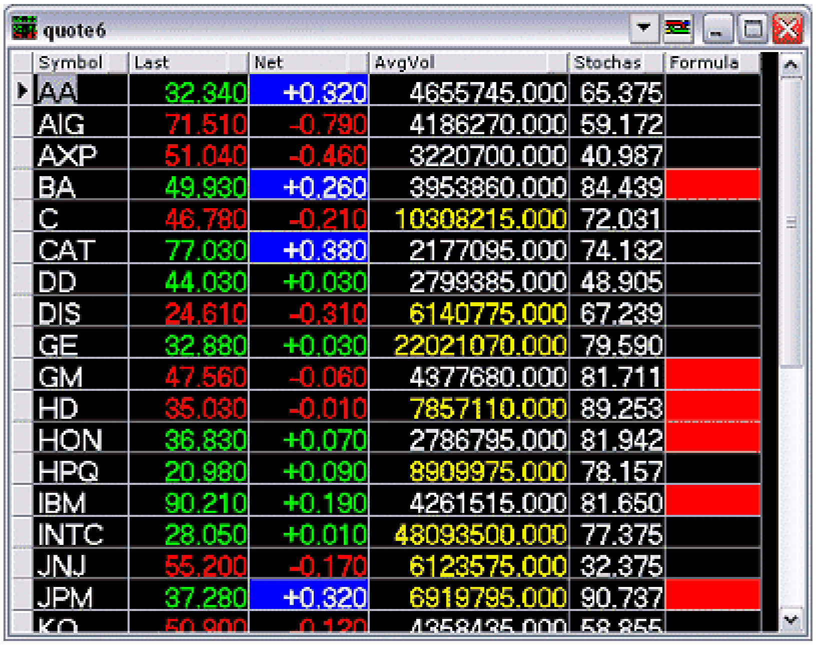
Column Properties (quote6,Formula1)

○ Use Quote Window Colors ● Custom Colors

Rule	Value	FG Color	BG Color	Blink
default		⬛	⬛	No
=	1	🟥	🟥	No

Top rule - considered first (low priority)
Bottom rule - considered last (high priority)

This will create red markers that highlight symbols with stochastic value greater than 80 or less than 20.



quote6

Symbol	Last	Net	AvgVol	Stochas	Formula
AA	32.340	+0.320	4655745.000	65.375	
AIG	71.510	-0.790	4186270.000	59.172	
AXP	51.040	-0.460	3220700.000	40.987	
BA	49.930	+0.260	3953860.000	84.439	🟥
C	46.780	-0.210	10308215.000	72.031	
CAT	77.030	+0.380	2177095.000	74.132	
DD	44.030	+0.030	2799385.000	48.905	
DIS	24.610	-0.310	6140775.000	67.239	
GE	32.880	+0.030	22021070.000	79.590	
GM	47.560	-0.060	4377680.000	81.711	🟥
HD	35.030	-0.010	7857110.000	89.253	🟥
HON	36.830	+0.070	2786795.000	81.942	🟥
HPQ	20.980	+0.090	8909975.000	78.157	
IBM	90.210	+0.190	4261515.000	81.650	🟥
INTC	28.050	+0.010	48093500.000	77.375	
JNJ	55.200	-0.170	6123575.000	32.375	
JPM	37.280	+0.320	6919795.000	90.737	🟥
KO	50.900	-0.120	4358435.000	58.855	

Adding Alert

There is a centralized alert service that helps you keep track of changing conditions.

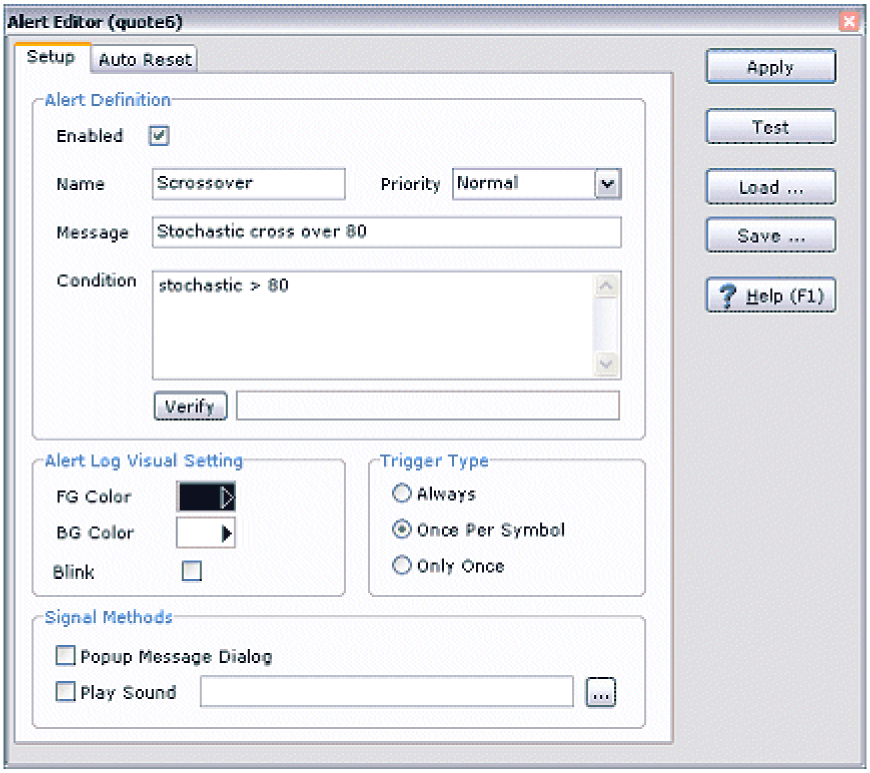
In the quote window we will setup an alert to notify us when stochastic of a particular symbol crossover 80. To setup an alert, right-click on the quote window, select **Add Alert** from the pop up menu.

Check the **Enable** option.

In the **Name** field enter `ScrossOver`, this will be the name you see when the alert pop up.

In the **Message** field type in `Stochastic cross over 80`.

In the **Condition** field type in `stochastic > 80`.



Alert Editor (quote6)

Alert Definition

Enabled ☑

Name: Scrossover

Priority: Normal

Message: Stochastic cross over 80

Condition: stochastic > 80

Alert Log Visual Setting

FG Color

BG Color

Blink □

Trigger Type

○ Always

● Once Per Symbol

○ Only Once

Signal Methods

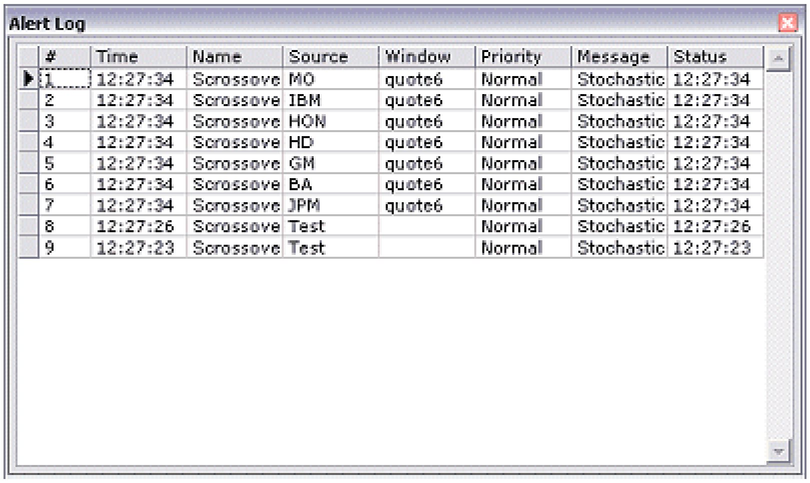
□ Popup Message Dialog

□ Play Sound

Press the **Verify** button to see if you have the correct syntax in the **Condition** field.

Press the **Apply** button to confirm the setting and start the alert.

To see the resulting alerts you can open the Alert Log Window by choosing **Program>Alert Log** in the main window, you will see the Alert Log show up.



#	Time	Name	Source	Window	Priority	Message	Status
1	12:27:34	Scrossove	MO	quote6	Normal	Stochastic	12:27:34
2	12:27:34	Scrossove	IBM	quote6	Normal	Stochastic	12:27:34
3	12:27:34	Scrossove	HON	quote6	Normal	Stochastic	12:27:34
4	12:27:34	Scrossove	HD	quote6	Normal	Stochastic	12:27:34
5	12:27:34	Scrossove	GM	quote6	Normal	Stochastic	12:27:34
6	12:27:34	Scrossove	BA	quote6	Normal	Stochastic	12:27:34
7	12:27:34	Scrossove	JPM	quote6	Normal	Stochastic	12:27:34
8	12:27:26	Scrossove	Test		Normal	Stochastic	12:27:26
9	12:27:23	Scrossove	Test		Normal	Stochastic	12:27:23

Now you can see the results of the alert you have just created. You can press the **Test** button in the alert editor to test the alert.

Tutorial: Creating a Symbol List

NeoTicker® lets you utilize symbol list in many places. For example, you can import symbol list in a quote window instead of manually typing the symbols in.

NeoTicker® already comes with several build-in symbol lists. This tutorial will show you how to create one.

Prerequisite

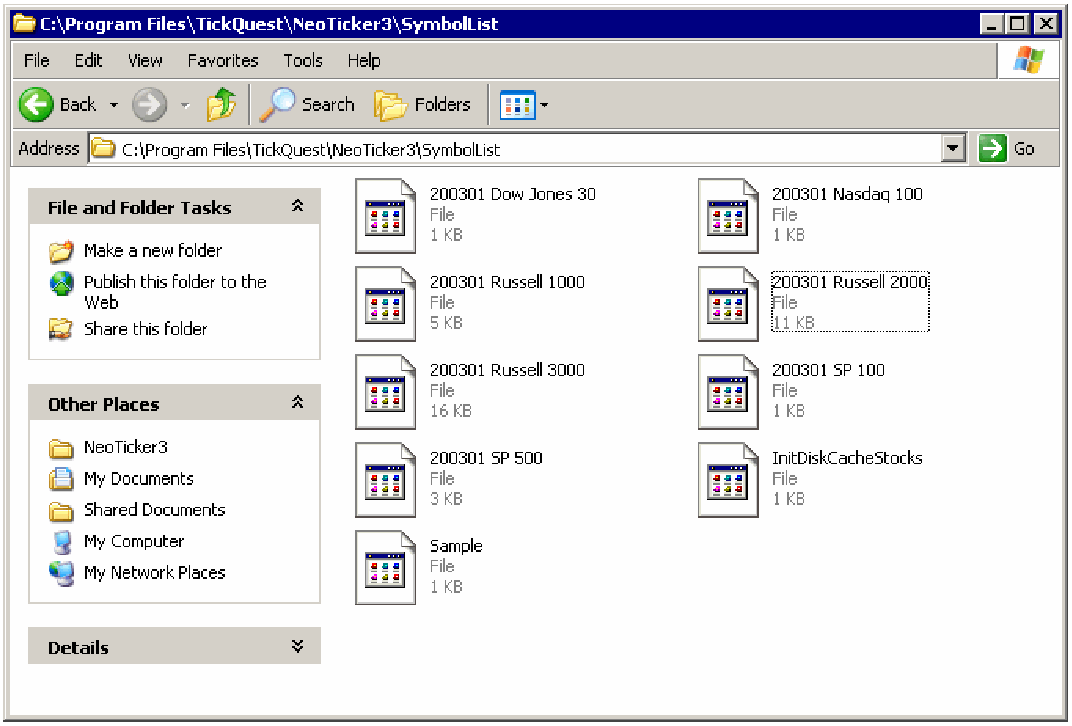
Familiar with working with text files in Windows.

Goal

After you finish this tutorial, you will be able to create symbol list for use in NeoTicker®.

Where the Symbol Lists are Stored

NeoTicker® automatically recognizes symbol list stored in the `SymbolList` directory in your NeoTicker® installation, e.g. `C:\Program Files\TickQuest\NeoTicker3\SymbolList`.



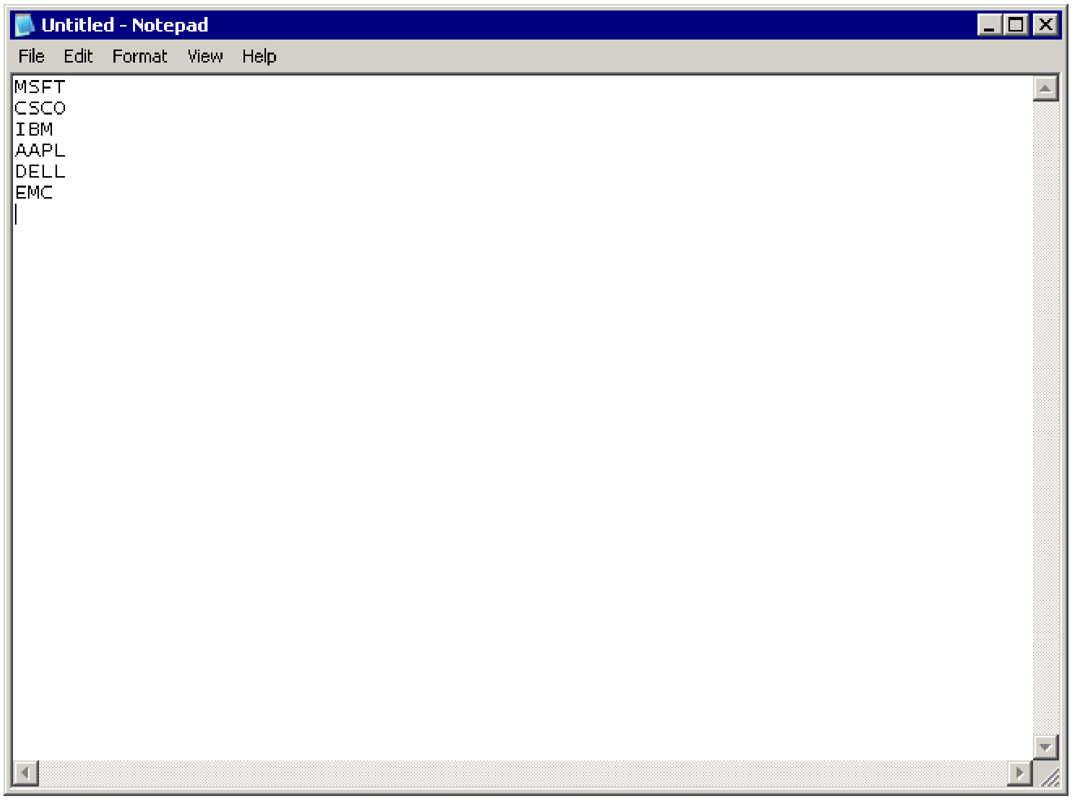
There is no suffix in the file names of the symbol lists that come with NeoTicker®. However, the symbol list files are simply text files, with a single symbol occupying a line. So you can edit the symbol list with Windows Notepad utility.

Creating a Symbol List with Notepad

We will create a symbol list with Notepad.

Open Notepad by choosing from Windows **Start** button, **All Programs>Accessories>Notepad**.

The Notepad application will be opened. Enter a few symbols into Notepad, with each symbol occupying a line. Notepad will look like:



```
MSFT
CSCO
IBM
AAPL
DELL
EMC
```

Choose **File>Save** from Notepad. Save the file to NeoTicker®'s symbol list directory.

This will create a symbol list for use within NeoTicker®.

Converting a Symbol List with UNIX origin for Use in Windows

Many servers on the Internet runs some variants of the UNIX operating system. If you downloaded a symbol list from the web, there is a good chance the file has an UNIX origin.

Trouble is, while a symbol list with an UNIX origin will open fine within Notepad, they are not Windows text file and cannot be used in NeoTicker® as symbol list.

If you encounter problem reading a symbol list you downloaded from the Internet, try the following steps to convert the file:

1 Open the file with Windows' WordPad utility.

2 From WordPad, choose **File>Save As**.

3 Choose a file name and for **Save as type**, choose **Text Document - MS-DOS Format**.

4 Press the **Save** button.

Tutorial: Working with Level 2 Windows

This help page will show you how to work with Level 2 windows. Level 2 windows are used to display market makers bid and ask information in the Nasdaq.

This tutorial will guide you through the basic workflow of a Level 2 window.

Prerequisite

None.

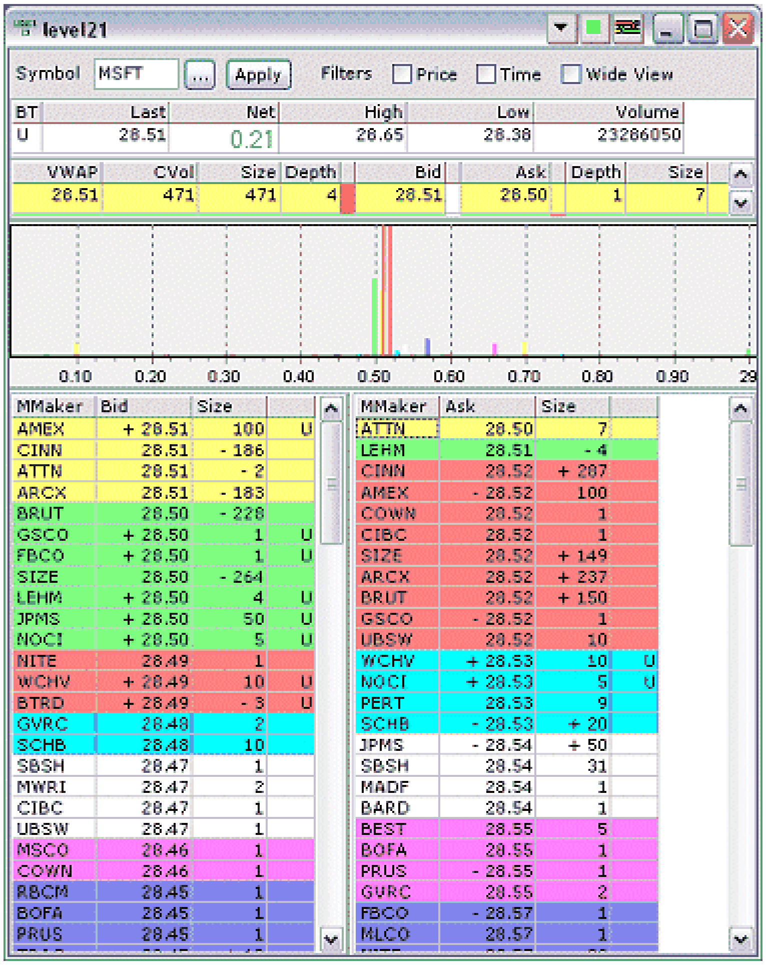
Goal

After you finish this tutorial you will know how to use the Level 2 window effectively.

Creating Level 2 Window

Create a Level 2 window by pressing the Level 2 window button LEVEL II on the main window.

After the Level 2 window is created, enter a Nasdaq symbol (e.g. MSFT) and press the **Apply** button. The symbol will be tracked in the newly created Level 2 window.



Symbol MSFT [...] [Apply] Filters □ Price □ Time □ Wide View

BT	Last	Net	High	Low	Volume
U	28.51	0.21	28.65	28.38	23286050

VWAP	CVol	Size	Depth	Bid	Ask	Depth	Size
28.51	471	471	4	28.51	28.50	1	7

MMaker	Bid	Size	
AMEX	+ 28.51	100	U
CINN	28.51	- 186	
ATTN	28.51	- 2	
ARCX	28.51	- 183	
BRUT	28.50	- 228	
GSCO	+ 28.50	1	U
FBCO	+ 28.50	1	U
SIZE	28.50	- 264	
LEHM	+ 28.50	4	U
JPMS	+ 28.50	50	U
NOCI	+ 28.50	5	U
NITE	28.49	1	
WCHV	+ 28.49	10	U
BTRD	+ 28.49	- 3	U
GVRC	28.48	2	
SCHB	28.48	10	
SBSH	28.47	1	
MWRI	28.47	2	
CIBC	28.47	1	
UBSW	28.47	1	
MSCO	28.46	1	
COWN	28.46	1	
RBCM	28.45	1	
BOFA	28.45	1	
PRUS	28.45	1	

MMaker	Ask	Size	
ATTN	28.50	7	
LEHM	28.51	- 4	
CINN	28.52	+ 207	
AMEX	- 28.52	100	
COWN	28.52	1	
CIBC	28.52	1	
SIZE	28.52	+ 149	
ARCX	28.52	+ 237	
BRUT	28.52	+ 150	
GSCO	- 28.52	1	
UBSW	28.52	10	
WCHV	+ 28.53	10	U
NOCI	+ 28.53	5	U
PERT	28.53	9	
SCHB	- 28.53	+ 20	
JPMS	- 28.54	+ 50	
SBSH	28.54	31	
MADF	28.54	1	
BARD	28.54	1	
BEST	28.55	5	
BOFA	28.55	1	
PRUS	- 28.55	1	
GVRC	28.55	2	
FBCO	- 28.57	1	
MLCO	28.57	1	

You can adjust the size of the Level 2 window. The vertical spacing between panels inside the Level 2 window can also be adjusted.

Filtering out Stale Records

There are stale records that you may not want to look at. These records are broadcast by the exchange so they are visible in the Level 2 window. You can choose to filter these records out by price or time.

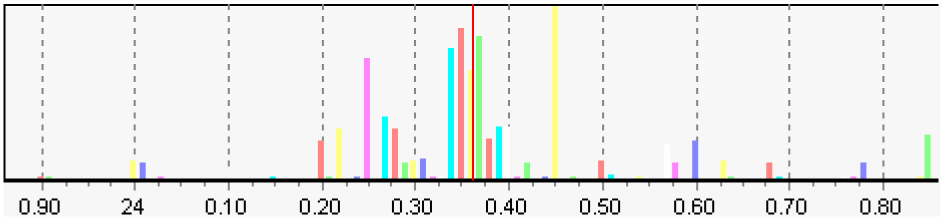
Check the **Filter Price** option. After this option is checked, records contain bid/ask prices that are too far away from the current price are filtered out.

Check the **Filter Time** option. After this option is checked, records that are not updated for a long time are filtered out.

You can configure the filtering criteria, see ***Adjusting Level 2 Time and Price Filters*** (see "Adjusting Time and Price Filters" on page 617).

Reading and Adjusting the Depth Chart

The depth chart displays visually the market makers and their bid/ask sizes at difference prices.



The vertical red line in the depth chart is the last traded price of the stock.

Each vertical bar in the chart represents the combined bid/ask size at a certain price. The color of the bar corresponds to the entries in the depth table.

You can switch between a wide and narrow view of the depth chart by toggling the **Wide View** option in the upper right corner of the Level 2 window.

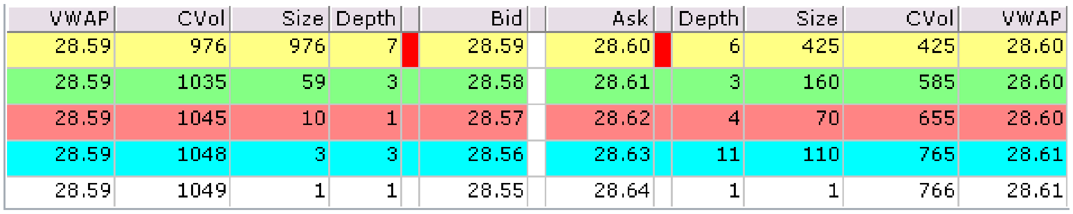
In wide view, the horizontal axis covers a wide price range and more market makers are shown.

In narrow view, the horizontal axis covers a narrow price range and fewer bars are shown, but with more details.

Depth chart can be hidden by choosing **Panels>Depth Chart** from the pop up menu.

To adjust the settings for the depth chart, refer to ***Adjusting Depth Chart*** (on page 618).

Reading the Depth Table



VWAP	CVol	Size	Depth		Bid		Ask		Depth	Size	CVol	VWAP
28.59	976	976	7	🟥	28.59		28.60	🟥	6	425	425	28.60
28.59	1035	59	3		28.58		28.61		3	160	585	28.60
28.59	1045	10	1		28.57		28.62		4	70	655	28.60
28.59	1048	3	3		28.56		28.63		11	110	765	28.61
28.59	1049	1	1		28.55		28.64		1	1	766	28.61

The depth table shows the aggregate of bid and ask records. For example, if there are four market makers bidding at 24.42, they will be shown as a single entry on the bid side of the depth table, with a depth of 4.

The bid and ask tables are ranked by their best offers. A red mark is shown for the currently

Depth table in another words are the summary of Level 2 information across different marker makers.

Here are what the fields mean:

VWAP - Volume Weighted Average Price. This is average of size multiplied by bid/ask price for all entries since the best offer.

CVol - Cummulated Volume. This is the cummulation of size since the best offer.

Size - The sum of bid/ask size for the same bid/ask price across different market makers.

Depth - Number of market makers at a specific bid/ask price.

Bid/Ask - The bid/ask price.

You can hide the fields by choosing **Depth Columns** from the pop up menu.

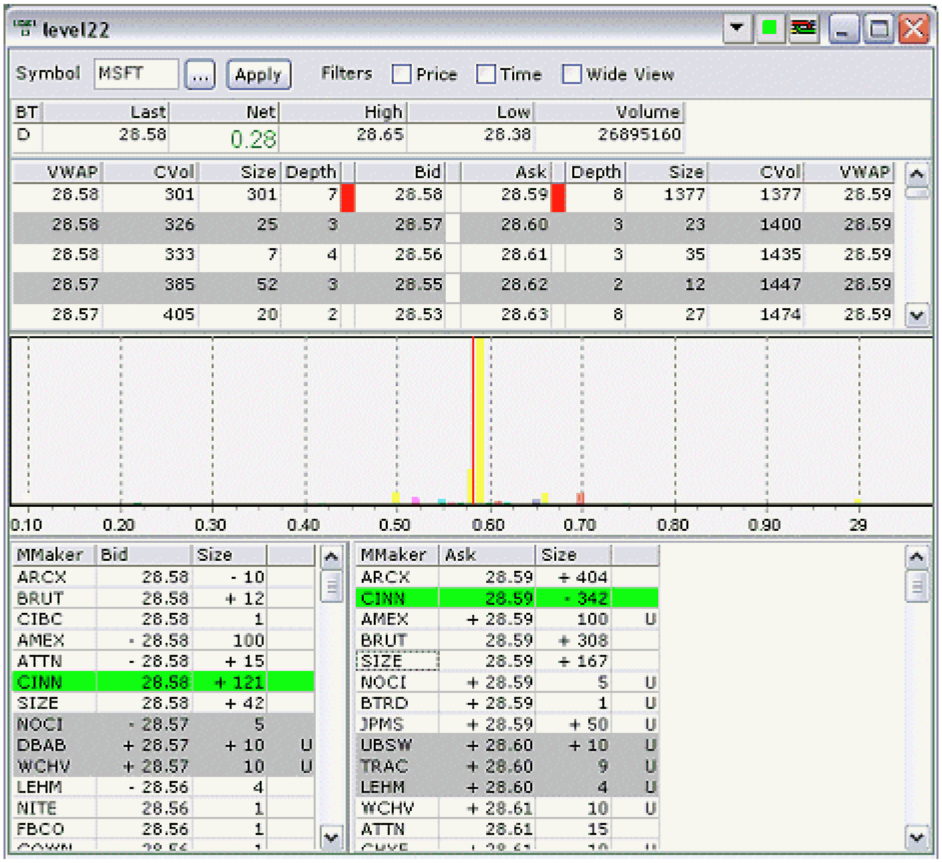
Tracking a Market Maker

You can track a market maker by marking their bid/ask with a special color.

Level 2 window is quite colorful, so color marking can be difficult to read. So the first thing we do is to make the Level 2 window plainer.

Right click to open the pop up menu, and choose **Color Style>2 Colors**. This reduces the color use in the Level 2 window to white and gray color except in the depth chart.

Then click on the name of the market maker you want to track under **MMaker** in the bid ask list. The market maker will be shown in a special color (light green by default).



Symbol MSFT

Filters □ Price □ Time □ Wide View

BT	Last	Net	High	Low	Volume
D	28.58	0.28	28.65	28.38	26895160

VWAP	CVol	Size	Depth	Bid	Ask	Depth	Size	CVol	VWAP
28.58	301	301	7	28.58	28.59	8	1377	1377	28.59
28.58	326	25	3	28.57	28.60	3	23	1400	28.59
28.58	333	7	4	28.56	28.61	3	35	1435	28.59
28.57	385	52	3	28.55	28.62	2	12	1447	28.59
28.57	405	20	2	28.53	28.63	8	27	1474	28.59

MMaker	Bid	Size	
ARCX	28.58	- 10	
BRUT	28.58	+ 12	
CIBC	28.58	1	
AMEX	- 28.58	100	
ATTN	- 28.58	+ 15	
CINN	28.58	+ 121	
SIZE	28.58	+ 42	
NOCI	- 28.57	5	
DBAB	+ 28.57	+ 10	U
WCHV	+ 28.57	10	U
LEHM	- 28.56	4	
NITE	28.56	1	
FBCO	28.56	1	
COWN	28.56	1	

MMaker	Ask	Size	
ARCX	28.59	+ 404	
CINN	28.59	- 342	
AMEX	+ 28.59	100	U
BRUT	28.59	+ 308	
SIZE	28.59	+ 167	
NOCI	+ 28.59	5	U
BTRD	+ 28.59	1	U
JPMS	+ 28.59	+ 50	U
UBSW	+ 28.60	+ 10	U
TRAC	+ 28.60	9	U
LEHM	+ 28.60	4	U
WCHV	+ 28.61	10	U
ATTN	28.61	15	
CHXE	+ 28.61	10	U

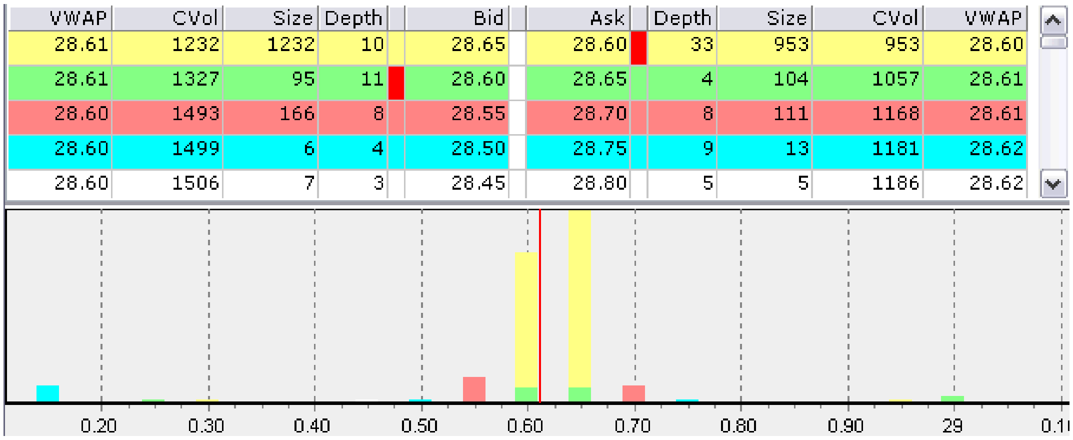
To un-track, press on the name again.

Summarizing Depth at Specific Increment

You can display the depth table and depth chart at a specific increment rather than at exact price. This is useful when the stock you are tracking has a lot of bid and ask records.

Choose **Setup** from the pop up menu to open Level 2 set up dialog. Press the **Depth** tab. Choose **Partition By Fixed Increment** and enter 0.05 for the increment value. Press the **Apply** button.

The resulting depth table and depth chart will have bid and ask increment at $0.05, and would look like:



VWAP	CVol	Size	Depth		Bid		Ask		Depth	Size	CVol	VWAP
28.61	1232	1232	10		28.65		28.60	🟥	33	953	953	28.60
28.61	1327	95	11	🟥	28.60		28.65		4	104	1057	28.61
28.60	1493	166	8		28.55		28.70		8	111	1168	28.61
28.60	1499	6	4		28.50		28.75		9	13	1181	28.62
28.60	1506	7	3		28.45		28.80		5	5	1186	28.62

Making a Fast Level 2 Window

If you have a fast computer, you can increase the update frequency of the Level 2 window.

To increase speed, right click on the Level 2 window to open the pop up menu and choose **Setup.**

Click on the **Speed** tab in the set up window.

A slider is available to let you set the update speed.

Tutorial: Windows Chaining

Window chaining allows you to send a single symbol to multiple function windows with a single command.

Window chaining is useful for users who want to perform multiple types of analysis on a single symbol and like to switch to analyze another symbol. For example, you can have a time and sales window and several time charts opened for MSFT, and use window chaining to switch the symbol of all these windows to INTC.

Prerequisite

You know how to create charts and quote windows.

Goal

After you finish this tutorial, you will be able to:

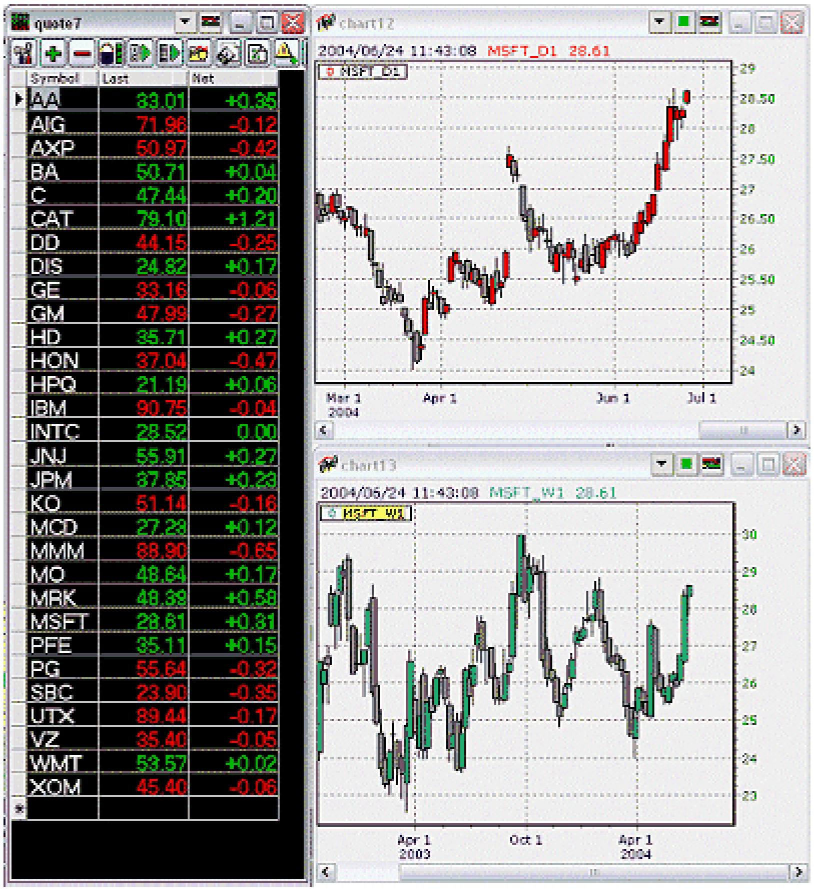
- Send symbols from chain
- Work with multiple chains

Setting up the Scenario

Create two new charts, add data series to each chart. You can use any symbol and any time frame.

Create a quote window, and load Dow Jones 30 into the quote window.

For example, you set up may look like this:

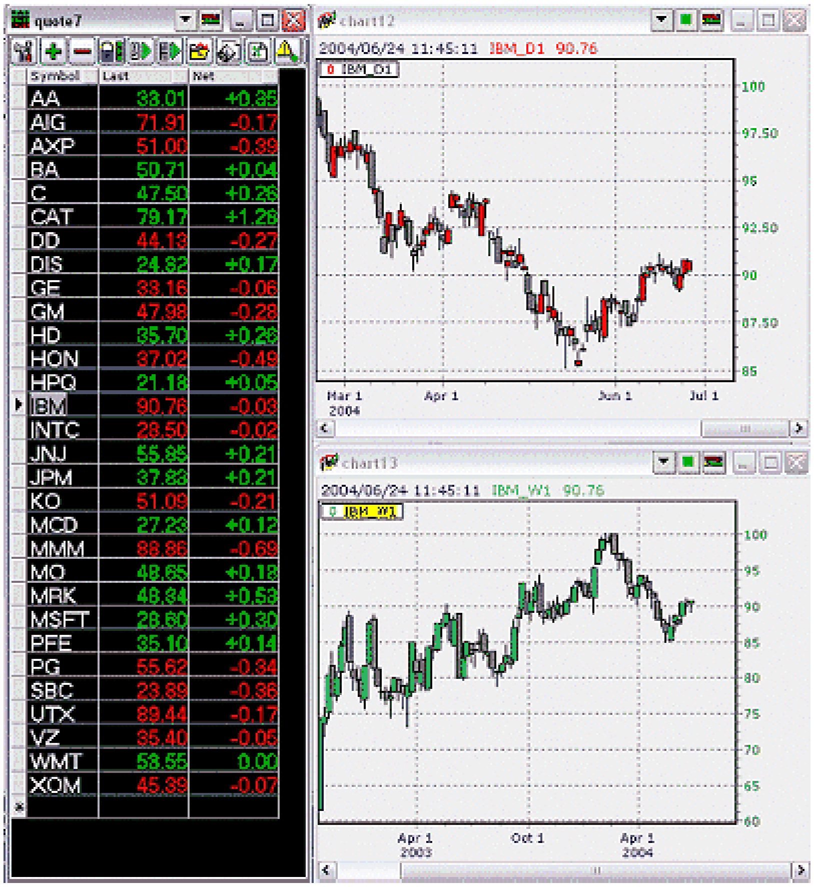


Symbol	Last	Net
AA	33.01	+0.35
AIG	71.96	-0.12
AXP	50.97	-0.42
BA	50.71	+0.04
C	47.44	+0.20
CAT	79.10	+1.21
DD	44.15	-0.25
DIS	24.82	+0.17
GE	33.16	-0.06
GM	47.99	-0.27
HD	35.71	+0.27
HON	37.04	-0.47
HPQ	21.19	+0.06
IBM	90.75	-0.04
INTC	28.52	0.00
JNJ	55.91	+0.27
JPM	37.85	+0.23
KO	51.14	-0.16
MCD	27.28	+0.12
MMM	88.90	-0.65
MO	48.64	+0.17
MRK	48.39	+0.58
MSFT	28.61	+0.81
PFE	35.11	+0.15
PG	55.64	-0.32
SBC	23.90	-0.35
UTX	89.44	-0.17
VZ	35.40	-0.05
WMT	53.57	+0.02
XOM	45.40	-0.06

Sending a Symbol to a Chain

Double click on a symbol, say IBM, on the quote window. This instructs the quote window to send the symbol IBM to the chain. The charts will replace their current symbols with IBM.

The set up after you double click on IBM will look like this:



quote7

Symbol	Last	Net
AA	33.01	+0.35
AIG	71.91	-0.17
AXP	51.00	-0.39
BA	50.71	+0.04
C	47.50	+0.26
CAT	79.17	+1.26
DD	44.13	-0.27
DIS	24.82	+0.17
GE	33.16	-0.06
GM	47.98	-0.28
HD	35.70	+0.26
HON	37.02	-0.49
HPQ	21.18	+0.05
IBM	90.76	-0.03
INTC	28.50	-0.02
JNJ	55.85	+0.21
JPM	37.83	+0.21
KO	51.09	-0.21
MCD	27.23	+0.12
MMM	88.86	-0.69
MO	48.65	+0.18
MRK	48.34	+0.53
MSFT	28.60	+0.30
PFE	35.10	+0.14
PG	55.62	-0.34
SBC	23.89	-0.36
UTX	89.44	-0.17
VZ	35.40	-0.05
WMT	53.55	0.00
XOM	45.39	-0.07

chart12

2004/06/24 11:45:11 IBM_D1 90.76

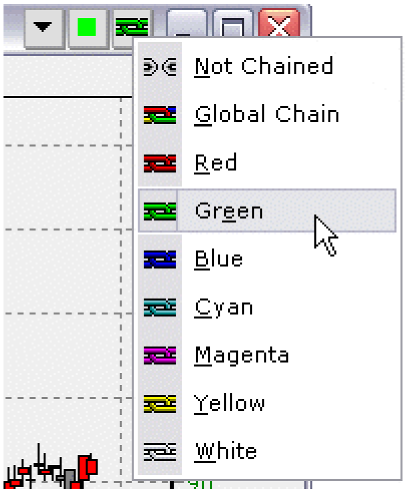
chart13

2004/06/24 11:45:11 IBM_W1 90.76

Setting up Different Types of Chains

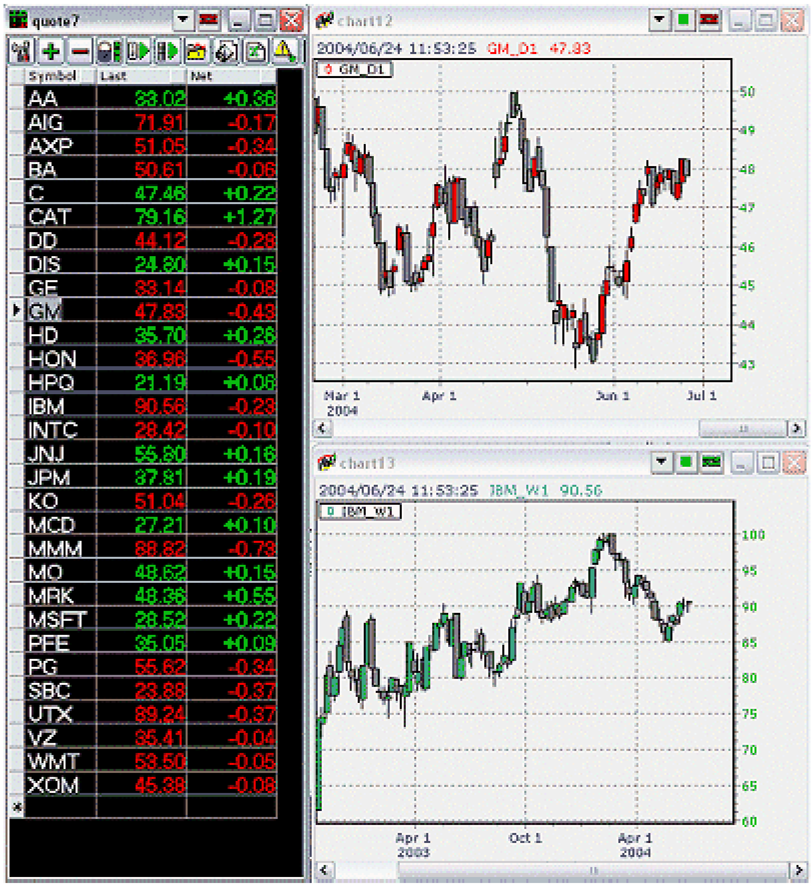
NeoTicker® provides a set of color chains and two chaining states (no chain and global chain). We will go through what they do.

Press the chain button on one of the chart's caption. The chain button looks like this or this , depending on the size of your window caption. A menu will be open:



Choose the red chain for one of the chart. Choose the green chain for the other chart. Choose the red chain for the quote window. Notice that the chain buttons change their appearance, reflecting the current chain of the window.

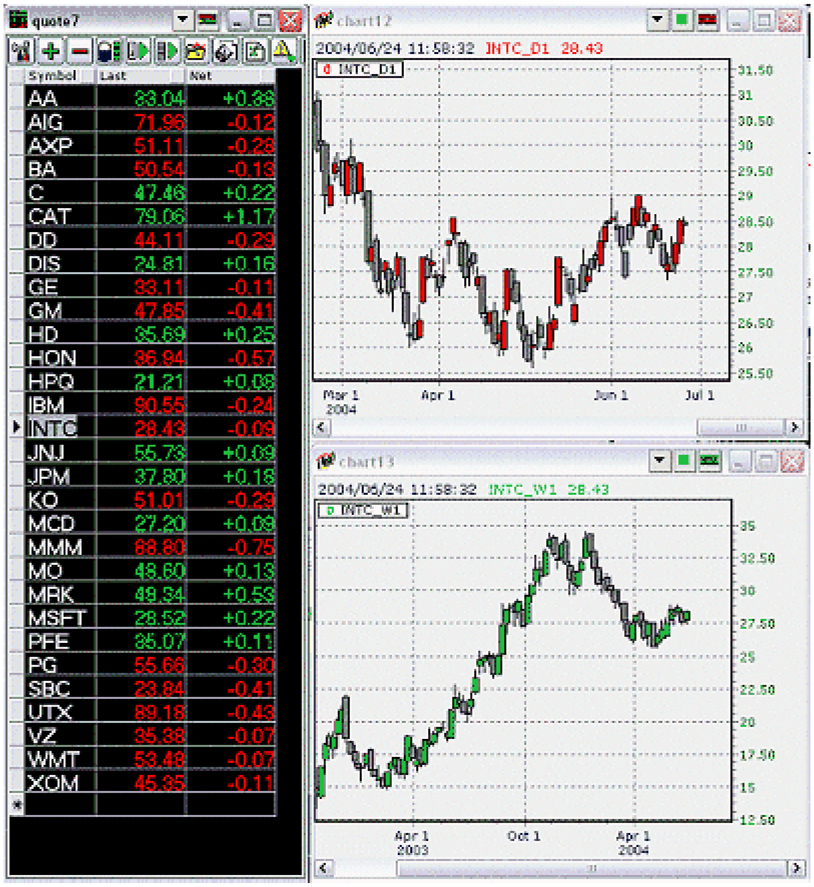
Now in quote window, double click on one of the symbol, say GM. Notice that only the chart with the red chain changes to displaying GM. The chart with the green chain is not changed. This is because the quote window is set to the red chain.



Symbol	Last	Net
AA	33.02	+0.36
AIG	71.91	-0.17
AXP	51.05	-0.34
BA	50.61	-0.06
C	47.46	+0.22
CAT	79.16	+1.27
DD	44.12	-0.28
DIS	24.80	+0.15
GE	33.14	-0.08
GM	47.83	-0.43
HD	35.70	+0.26
HON	36.96	-0.55
HPQ	21.19	+0.06
IBM	90.56	-0.23
INTC	28.42	-0.10
JNJ	55.80	+0.16
JPM	37.81	+0.19
KO	51.04	-0.26
MCD	27.21	+0.10
MMM	88.82	-0.73
MO	48.62	+0.15
MRK	48.36	+0.55
MSFT	28.52	+0.22
PFE	35.05	+0.09
PG	55.62	-0.34
SBC	23.88	-0.37
UTX	89.24	-0.37
VZ	35.41	-0.04
WMT	53.50	-0.05
XOM	45.38	-0.08

Similarly, if you change the chain of the quote window to green, when the double click on a symbol, only the chart with the green chain will change.

For the next step, change the chain of the quote window to the global chain. In quote window, double click on a symbol, say INTC. Notice that both the chart with red chain and the chart with green chain will replace their symbol with INTC.



2004/06/24 11:58:32 INTC_D1 28.43

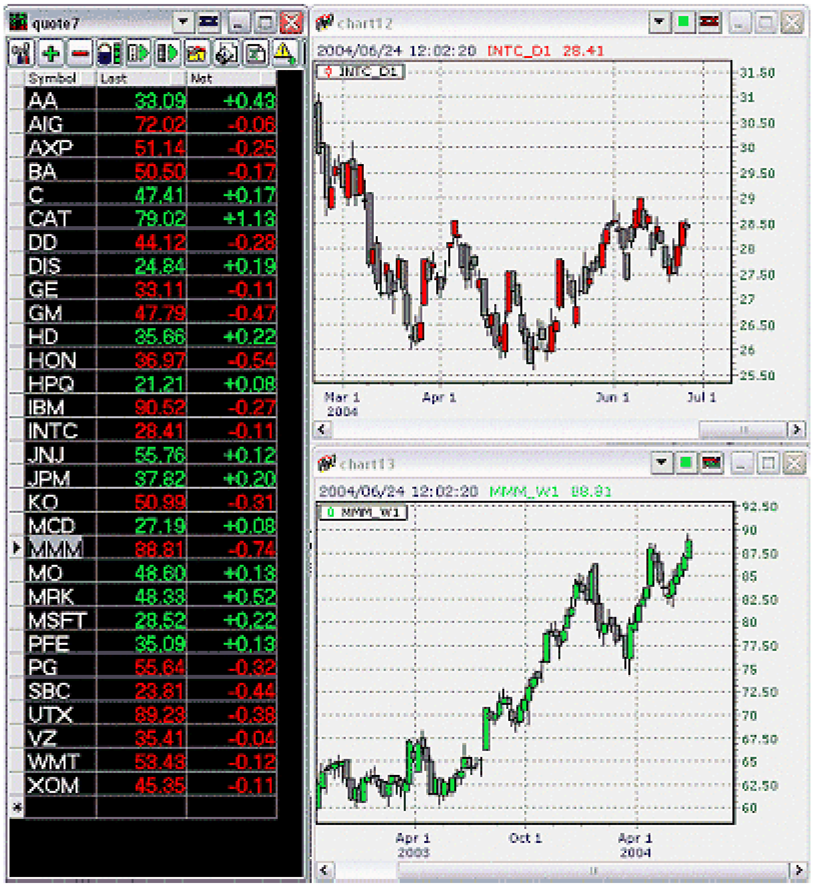
2004/06/24 11:58:32 INTC_W1 28.43

Symbol	Last	Net
AA	33.04	+0.38
AIG	71.96	-0.12
AXP	51.11	-0.28
BA	50.54	-0.13
C	47.46	+0.22
CAT	79.06	+1.17
DD	44.11	-0.29
DIS	24.81	+0.16
GE	33.11	-0.11
GM	47.85	-0.41
HD	35.69	+0.25
HON	36.94	-0.57
HPQ	21.21	+0.08
IBM	90.55	-0.24
INTC	28.43	-0.09
JNJ	55.73	+0.09
JPM	37.80	+0.18
KO	51.01	-0.29
MCD	27.20	+0.09
MMM	88.80	-0.75
MO	48.60	+0.13
MRK	49.34	+0.53
MSFT	28.52	+0.22
PFE	35.07	+0.11
PG	55.66	-0.30
SBC	23.84	-0.41
UTX	89.18	-0.43
VZ	35.38	-0.07
WMT	53.48	-0.07
XOM	45.35	-0.11

The global chain is for global matching. If a source is a global chain, all chains will receive the symbol you sent.

Let's see what happens when you set the receiving end to a global chain. Change the chain of the quote window to blue and the chain of one of the chart to the global chain.

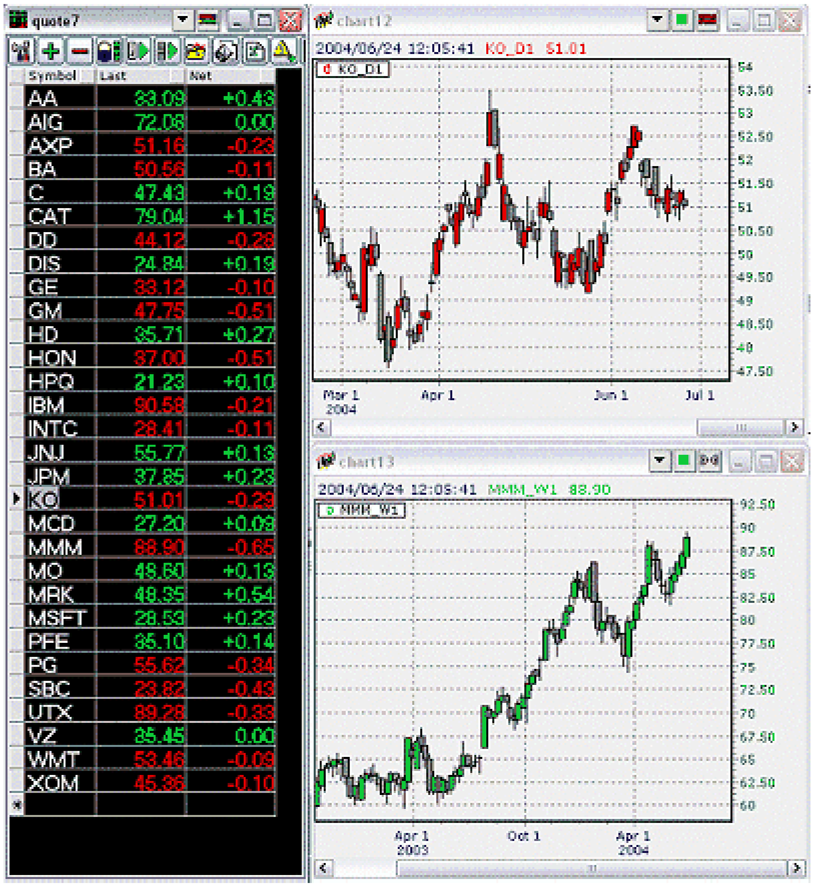
Double click on a symbol, say MMM. Notice that the the chart with the global chain will have the symbol changed to MMM. When the chain on the receiving side will receive symbol from any chain, including the blue chain of the quote window.



Symbol	Last	Net
AA	33.09	+0.43
AIG	72.02	-0.06
AXP	51.14	-0.25
BA	50.50	-0.17
C	47.41	+0.17
CAT	79.02	+1.13
DD	44.12	-0.28
DIS	24.84	+0.19
GE	33.11	-0.11
GM	47.79	-0.47
HD	35.66	+0.22
HON	36.97	-0.54
HPQ	21.21	+0.08
IBM	90.52	-0.27
INTC	28.41	-0.11
JNJ	55.76	+0.12
JPM	37.82	+0.20
KO	50.99	-0.31
MCD	27.19	+0.08
MMM	88.81	-0.74
MO	48.60	+0.13
MRK	48.33	+0.52
MSFT	28.52	+0.22
PFE	35.09	+0.13
PG	55.64	-0.32
SBC	23.81	-0.44
UTX	89.23	-0.38
VZ	35.41	-0.04
WMT	53.43	-0.12
XOM	45.35	-0.11

Now change the chain of the quote window to the global chain, one chart to red chain, and another chart to not chained. In quote window, double click on a symbol, say KO.

The set up should look like:



Symbol	Last	Net
AA	33.09	+0.43
AIG	72.08	0.00
AXP	51.16	-0.23
BA	50.56	-0.11
C	47.43	+0.19
CAT	79.04	+1.15
DD	44.12	-0.28
DIS	24.84	+0.19
GE	33.12	-0.10
GM	47.75	-0.51
HD	35.71	+0.27
HON	37.00	-0.51
HPQ	21.23	+0.10
IBM	90.58	-0.21
INTC	28.41	-0.11
JNJ	55.77	+0.13
JPM	37.85	+0.23
KO	51.01	-0.29
MCD	27.20	+0.09
MMM	88.90	-0.65
MO	48.60	+0.13
MRK	48.35	+0.54
MSFT	28.53	+0.23
PFE	35.10	+0.14
PG	55.62	-0.34
SBC	23.82	-0.43
UTX	89.28	-0.33
VZ	35.45	0.00
WMT	53.46	-0.09
XOM	45.36	-0.10

Notice that only the chart of red chain is changed to KO. The chart that is not chained will not change.

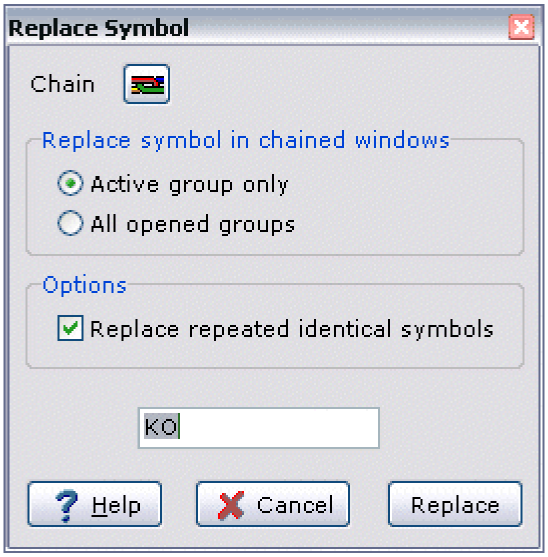
When a window is set to not chained. It will not receive symbol change from any chain, even the global chain.

If the sender is set to not chained. It will not send symbol down any chain, including the global chain.

Using the Chain Dialog to Send Symbols

You can send symbols down a chain directly without using a quote window.

Press F8 on the keyboard. The replace symbol dialog will be opened.



Replace symbol in chained windows

● Active group only

○ All opened groups

Options

☑ Replace repeated identical symbols

KO

You can select chain and enter the symbol, then press the **Replace** button to send to chain.

Tutorial: Trading System Testing with Backtest EZ

NeoTicker® supports a rich set of trading system related features that will satisfy the most demanding system traders. Yet much of the power is available through Backtest EZ.

Backtest EZ is signal driven trading system. There is no need for you to do any programming. You will focus on developing the trading signals rather than the specific of programming logic.

Prerequisite

You know how to create charts.

To follow the last step of this tutorial, some knowledge about pattern scanner will be helpful.

Goal

After you finish this tutorial, you will be able to:

- Create a trading system with Backtest EZ.

Background

What are Trading Systems

Trading systems are mechanical trading models. Because trading systems are mechanical, it does not fall into emotional pitfall of human traders.

What NeoTicker® Provides

NeoTicker® provides a rich set of trading system features to help you back test trading system. Back testing is applying the mechanical trading system on historical data to see the performance of the trading system.

NeoTicker® also tracks trading system when it runs on real-time live data. This way, you can follow the orders from the mechanical modal for live trading.

How NeoTicker® Implements Trading Systems

NeoTicker® implements trading systems as a type of specialized indicators.

Trading systems have all the convenient of indicators, but also has the ability to access the trading system specific features such as issuing orders, analyzing performance, etc.

Because trading systems are specialized indicators, you can use them in quote window, pattern scanner etc to analyze the same trading system on different symbols.

What does Backtest EZ Do

Backtest EZ is an indicator that takes care of most of the common tasks in a trading system. All you need to do is to provide entry signals and exit conditions. Using Backtest EZ, you will focus on developing the trading signals rather than the specific of programming logic.

This is in contrast with many applications that require you to write a program even for a very simple trading systems.

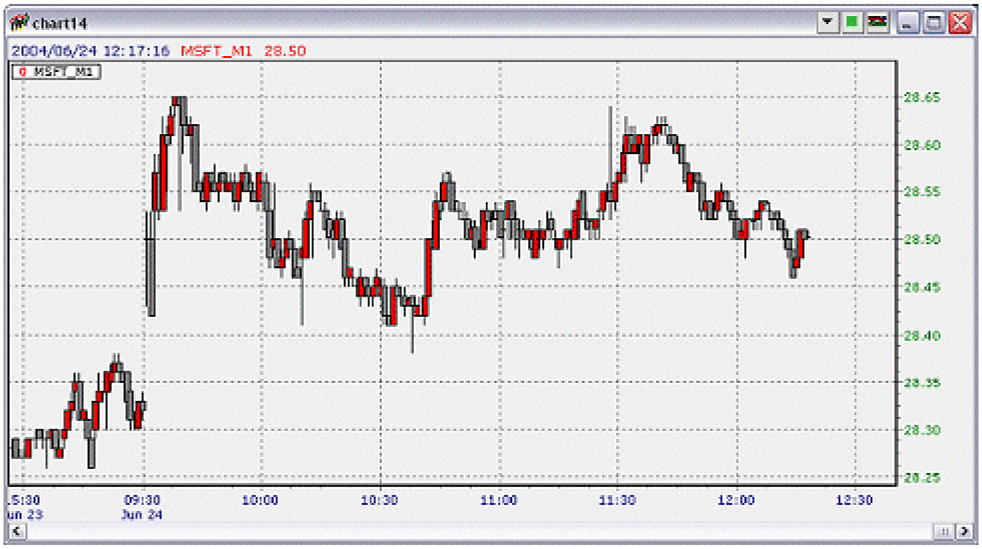
Applying Backtest EZ

Create a chart of 1-minute MSFT. Load 25 days of data into the chart. You can use **Days of Data to Load** from pop up menu to do this.

There are about 9750 bars (25 days * 390 bars/day) in this chart. This is quite a lot of data for normal chart viewing, but for system testing, you should test a long period to gain confidence of your system's performance. The chart below shows only a part of the data.

You may also want to turn off Dynamic Grid and User Panel in the chart to maximize the viewing area.

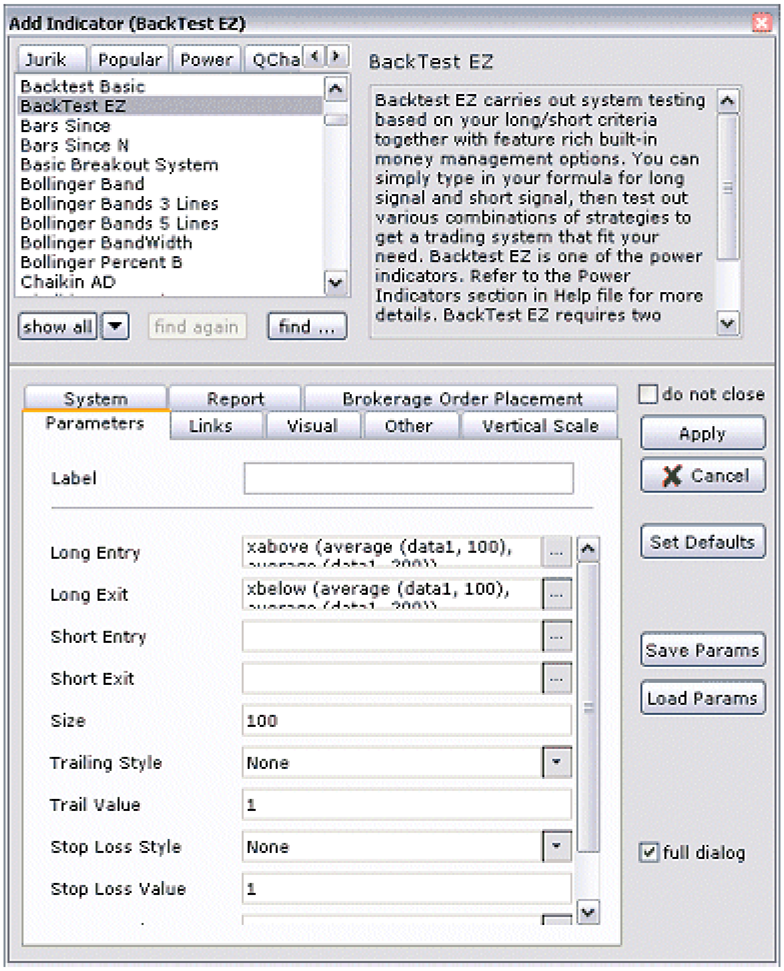
If you are interested in EOD trading systems, you can use a 500 day daily chart of MSFT to follow this tutorial.



Add the indicator Backtest EZ. Here are the settings when you add it.

First we will set how the system trades. We are creating a long side only, moving average crossover system. Under the **Parameters** tab:

- **Long Entry** is set to `xabove (average (data1, 100), average (data1, 200))`
- **Long Exit** is set to `xbelow (average (data1, 100), average (data1, 200))`
- No **Short Entry** and **Short Exit**
- **Size** is 100
- **Trailing Style**, **Stop Loss Style** and **Target Style** all set to None.

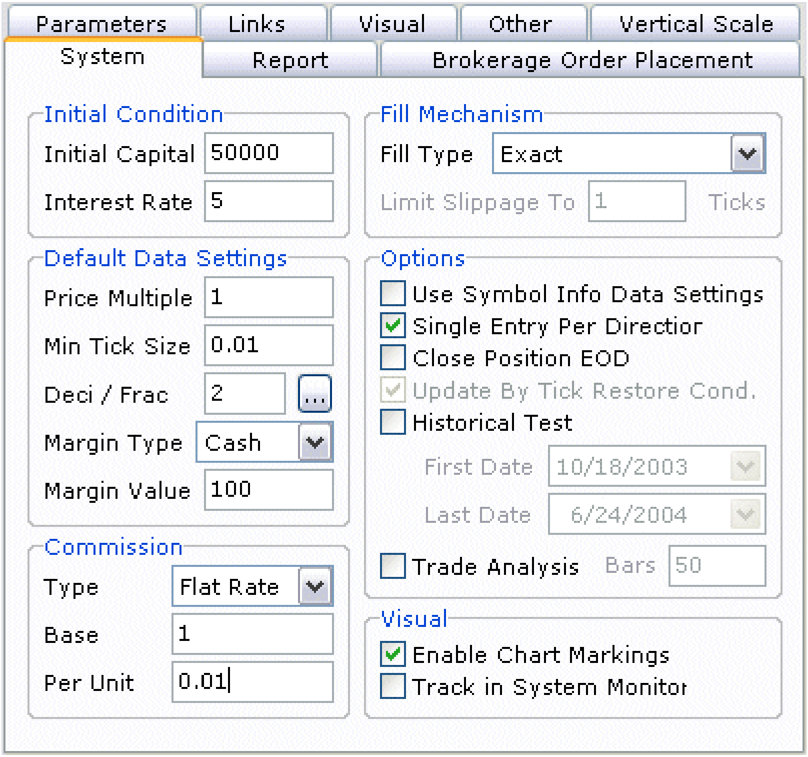


Backtest EZ carries out system testing based on your long/short criteria together with feature rich built-in money management options. You can simply type in your formula for long signal and short signal, then test out various combinations of strategies to get a trading system that fit your need. Backtest EZ is one of the power indicators. Refer to the Power Indicators section in Help file for more details. BackTest EZ requires two

Label	
Long Entry	xabove (average (data1, 100),
Long Exit	xbelow (average (data1, 100),
Short Entry	
Short Exit	
Size	100
Trailing Style	None
Trail Value	1
Stop Loss Style	None
Stop Loss Value	1

Then we tell Backtest EZ more about the specific instrument we are trading. Under the **System** tab:

- In **Commission**, set **Base** price to 1, **Per Unit** price to 0.01.



Initial Condition
Initial Capital 50000
Interest Rate 5

Default Data Settings
Price Multiple 1
Min Tick Size 0.01
Deci / Frac 2 ...
Margin Type Cash
Margin Value 100

Commission
Type Flat Rate
Base 1
Per Unit 0.01

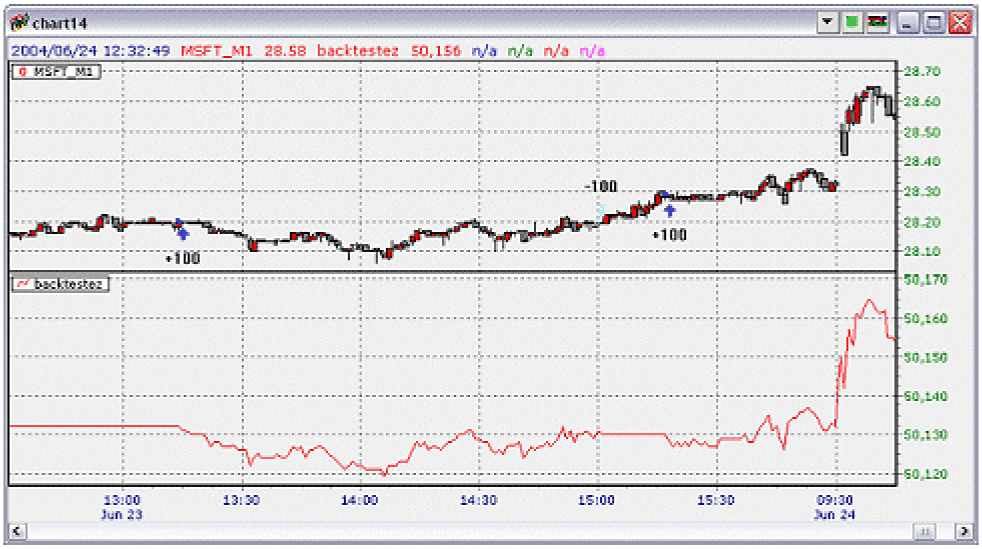
Fill Mechanism
Fill Type Exact
Limit Slippage To 1 Ticks

Options
□ Use Symbol Info Data Settings
☑ Single Entry Per Direction
□ Close Position EOD
☑ Update By Tick Restore Cond.
□ Historical Test
First Date 10/18/2003
Last Date 6/24/2004
□ Trade Analysis Bars 50

Visual
☑ Enable Chart Markings
□ Track in System Monitor

Backtest EZ starts with $50000 capital. You can change this value under the **System** tab.

After you press the **Apply** button, Backtest EZ will be added to the chart. You may need to scroll the chart to see the trades.



Backtest EZ will use the rules you specify to trade on 1-Min MSFT. Backtest EZ's first plot shows the equity curve of the trading system. Equity curve is how much money is in the trading system, which is the total of the cash plus the price of the equity.

In the example chart above, the blue upward arrows mark the entry orders, with a the number of shares (100) also marked. The light blue downward arrows mark the exit orders. Note that there is a little horizontal marking at the tail of the exit orders. This indicates that the order is for covering a long order, rather than a short entry.

Your chart will look slightly different, depending on the data.

This is a long only simple moving average cross over system that will buy when the 100-period simple moving average cross above the 200-period simple moving average. For the simplicity of this tutorial, money management rules such as stop loss are not included.

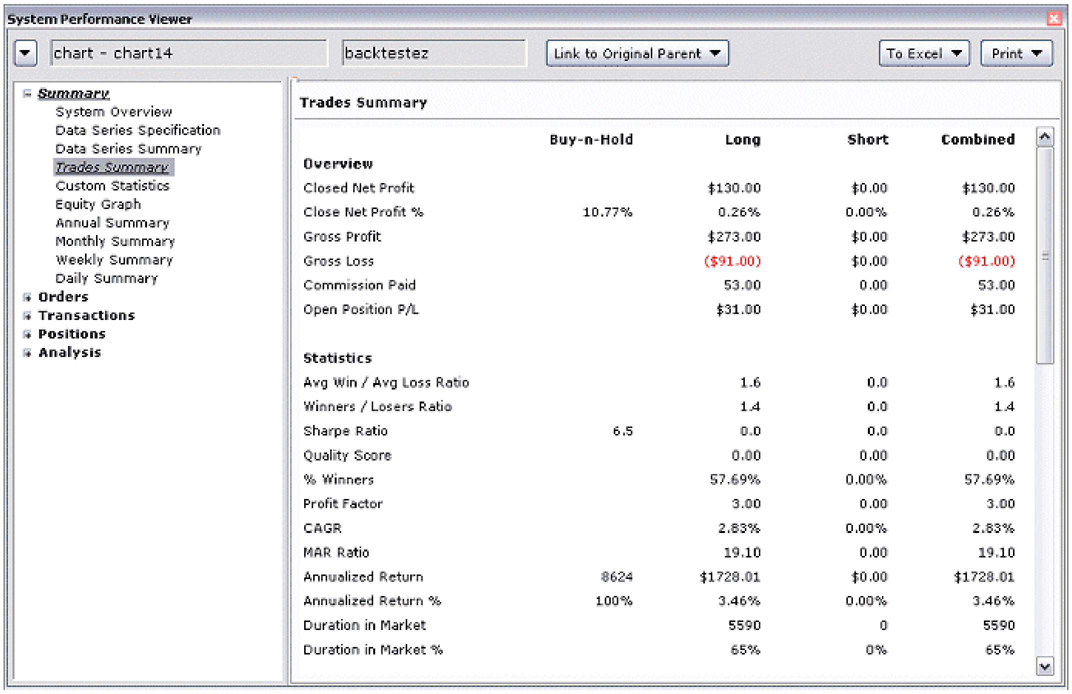
Using Performance Viewer to Analyze Trading System

The trading system created in the previous section is not doing so well. It starts with $50000 cash and after 25 days of trading, it stays more or less at the same level.

Performance Viewer is a tool that can help you analyze trading systems and make improvements.

Click on Backtest EZ's legend to select it, right click to open the pop up menu, and choose **Trading System>Open Performance Viewer**.

Performance Viewer provides a wide variety of options to help you analyze the trades. For example, to get a trade summary, you can click on **Trade Summary** under **Summary**.

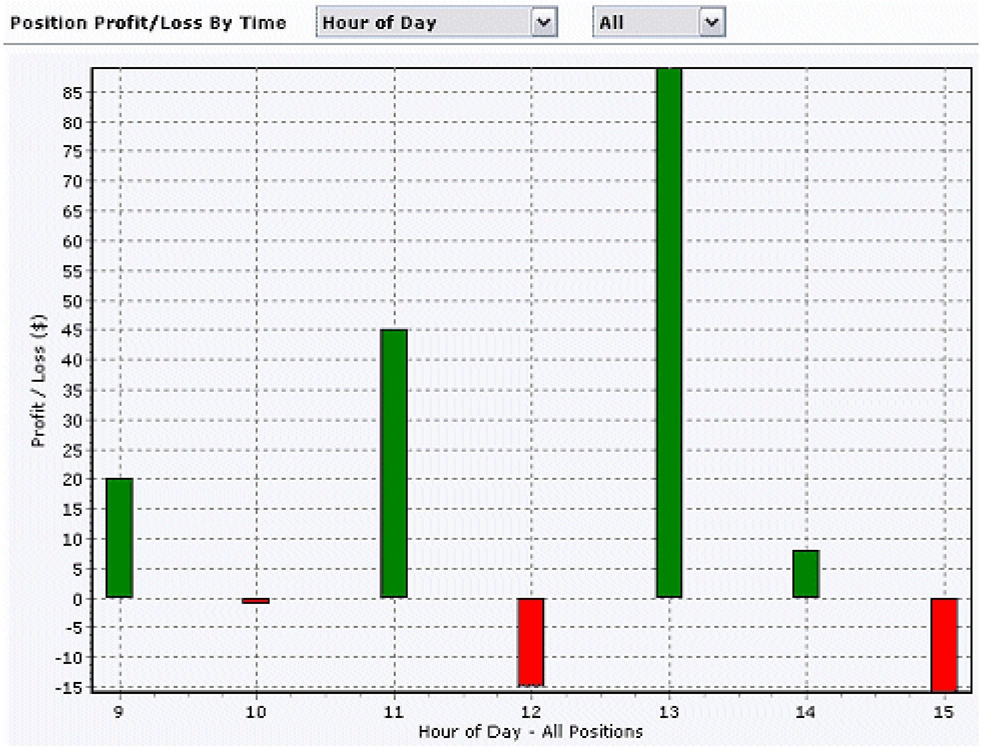


Trades Summary

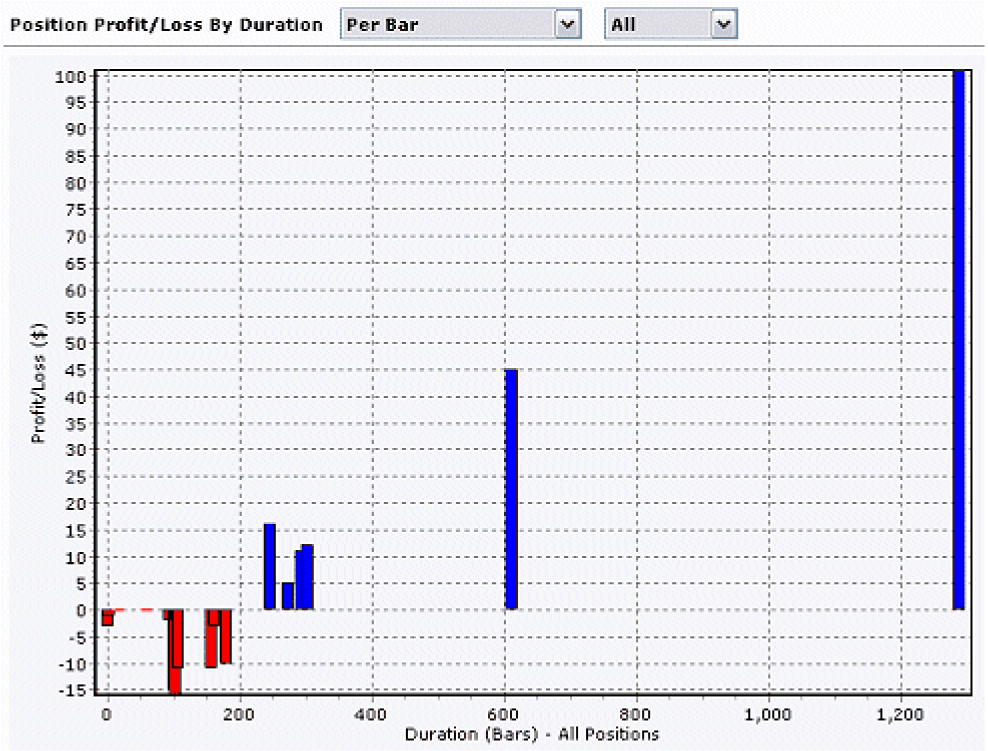
	Buy-n-Hold	Long	Short	Combined
Overview				
Closed Net Profit		$130.00	$0.00	$130.00
Close Net Profit %	10.77%	0.26%	0.00%	0.26%
Gross Profit		$273.00	$0.00	$273.00
Gross Loss		($91.00)	$0.00	($91.00)
Commission Paid		53.00	0.00	53.00
Open Position P/L		$31.00	$0.00	$31.00
Statistics				
Avg Win / Avg Loss Ratio		1.6	0.0	1.6
Winners / Losers Ratio		1.4	0.0	1.4
Sharpe Ratio	6.5	0.0	0.0	0.0
Quality Score		0.00	0.00	0.00
% Winners		57.69%	0.00%	57.69%
Profit Factor		3.00	0.00	3.00
CAGR		2.83%	0.00%	2.83%
MAR Ratio		19.10	0.00	19.10
Annualized Return	8624	$1728.01	$0.00	$1728.01
Annualized Return %	100%	3.46%	0.00%	3.46%
Duration in Market		5590	0	5590
Duration in Market %		65%	0%	65%

Scroll down the report page. There are **Winners** and **Losers** sections in the report. Under the **Winners** and **Losers** sections, you can see the statistics for winning and losing averages and the number of winning and losing trades.

Under **Positions** are position statistics. For example, click on **Position Profit/Loss By Time** under **Position**. This is a distribution of profit/lost by the trading hour. From the distribution, we may form a theory that this trading system tend make money between 11am to 1pm. You can attempt to fine tune the trading system based on this information.



Click on **Position Profit/Loss By Duration**.



On this page, we see that the majority of the profit is made by holding on to a position. You can form a theory that by holding on to positions, the system will improve.

Using various statistics, you can incrementally improve the performance of a particular trading system.

Applying Backtest EZ to a Group of Stocks

We can easily apply Backtest EZ we just created to a group of stocks to see how each stock performs under this trading system.

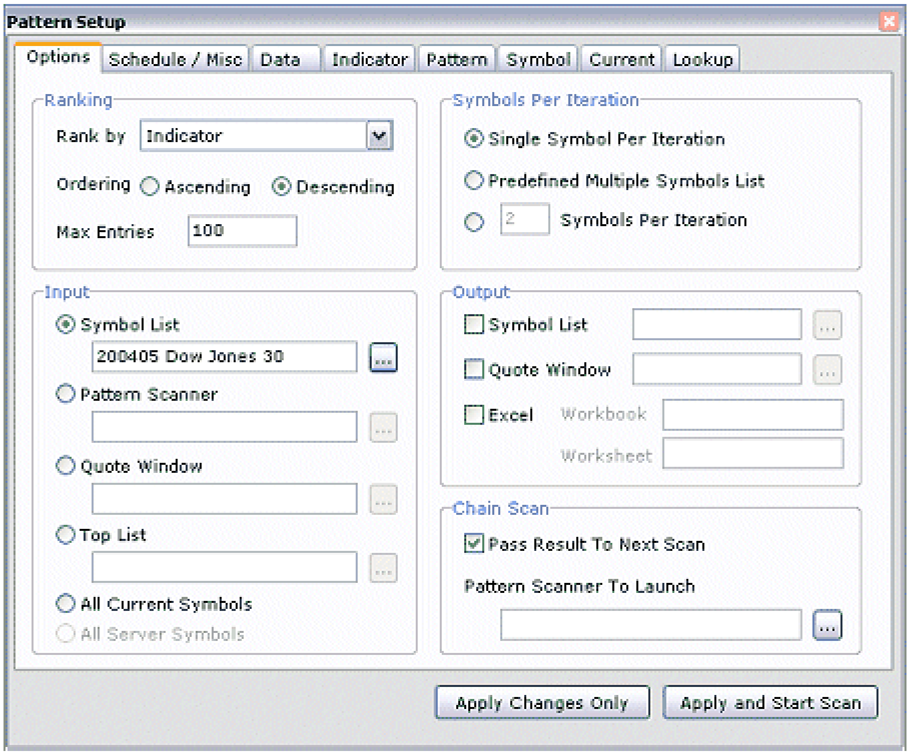
We will use pattern scanner for this job, and use the chart to help setup pattern scanner. To readers who are not familiar with pattern scanner, pattern scanner is a highly versatile scanner that can handle heavy duty scanning job.

Let's save the chart first. Right click on the chart to open pop up menu and choose **Save As**. Choose a file name you can remember, for example, `forscan`.

Now create a pattern scanner by clicking on the pattern scanner button on the main window. Right click on the pattern scanner and choose **Setup** to open pattern scanner's setup window. Here is how to set up:

First we set the symbols to scan and how to rank the scanning result. Under the **Options** tab:

- Set **Input** to **Symbol List** and select one of the Dow Jones 30 symbol list. We will be applying Backtest EZ to stocks in the Dow Jones 30.
- Set **Rank by** to Indicator. This will make the scanning rank by the equity curves of running Backtest EZ on multiple stocks.



Pattern Setup

Options | Schedule / Misc | Data | Indicator | Pattern | Symbol | Current | Lookup

Ranking
Rank by: Indicator
Ordering: ○ Ascending ● Descending
Max Entries: 100

Symbols Per Iteration
● Single Symbol Per Iteration
○ Predefined Multiple Symbols List
○ 2 Symbols Per Iteration

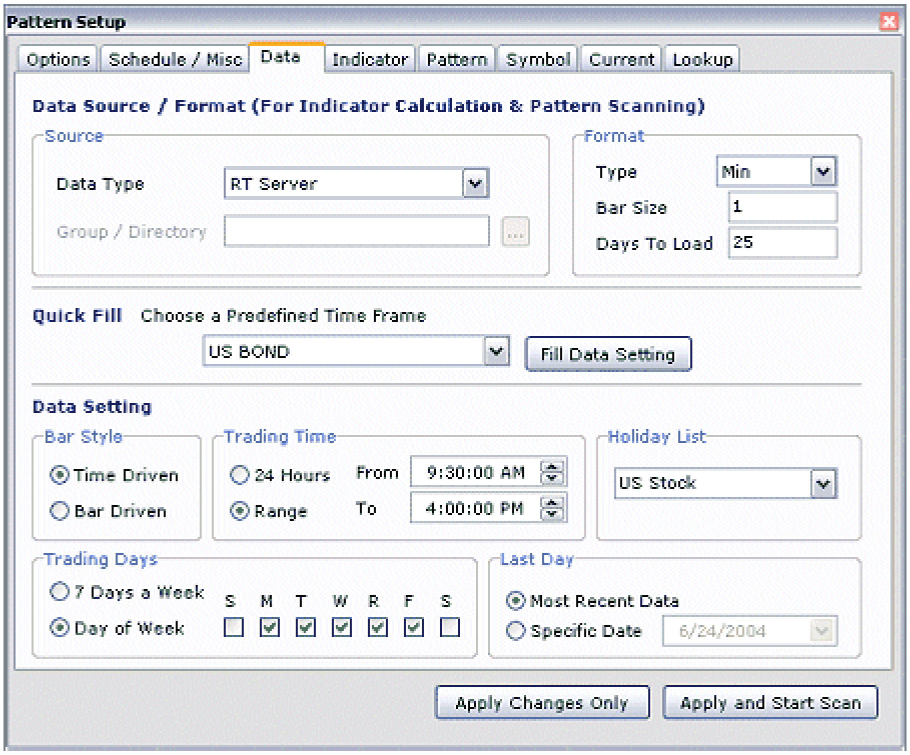
Input
● Symbol List: 200405 Dow Jones 30
○ Pattern Scanner
○ Quote Window
○ Top List
○ All Current Symbols
○ All Server Symbols

Output
□ Symbol List
□ Quote Window
□ Excel Workbook
Worksheet

Chain Scan
☑ Pass Result To Next Scan
Pattern Scanner To Launch

Then we set the time frame for scanning. We want the time frame to be the same as the chart we created. Under the **Data** tab:

- Set **Type** to Min and **Bar Size** to 1. This will make Backtest EZ works on 1-Min bar.
- Set **Days to Load** to 25. This is the duration of the testing.



Data Source / Format (For Indicator Calculation & Pattern Scanning)

Source

Data Type: RT Server

Group / Directory:

Format

Type: Min

Bar Size: 1

Days To Load: 25

Quick Fill Choose a Predefined Time Frame: US BOND

Data Setting

Bar Style: ● Time Driven ○ Bar Driven

Trading Time: ○ 24 Hours ● Range From 9:30:00 AM To 4:00:00 PM

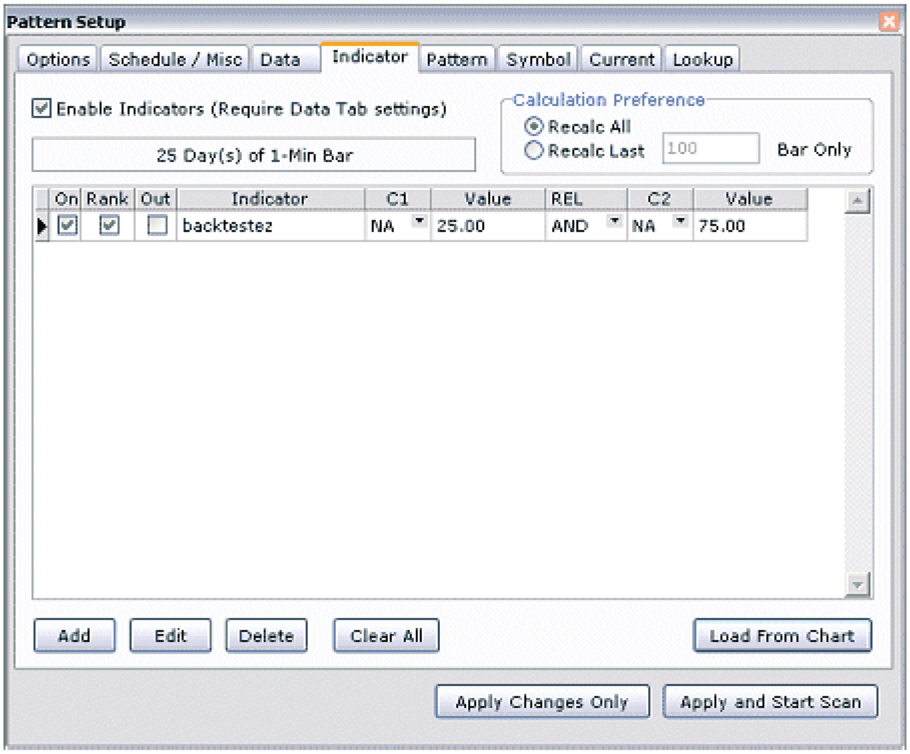
Holiday List: US Stock

Trading Days: ○ 7 Days a Week ● Day of Week S □ M ☑ T ☑ W ☑ R ☑ F ☑ S □

Last Day: ● Most Recent Data ○ Specific Date 6/24/2004

Then we tell pattern scanner to use the chart to set up scanning. Under the **Indicator** tab:

- Check **Enable Indicators**.This tells pattern scanner we are scanning an indicator (Backtest EZ).
- Press the **Load from Chart** button, and choose the file `forscan`. This will import the Backtest EZ setup from the chart you just saved.
- Make sure the **Rank** option is checked besides Backtest EZ.



☑ Enable Indicators (Require Data Tab settings)

25 Day(s) of 1-Min Bar

Calculation Preference

● Recalc All

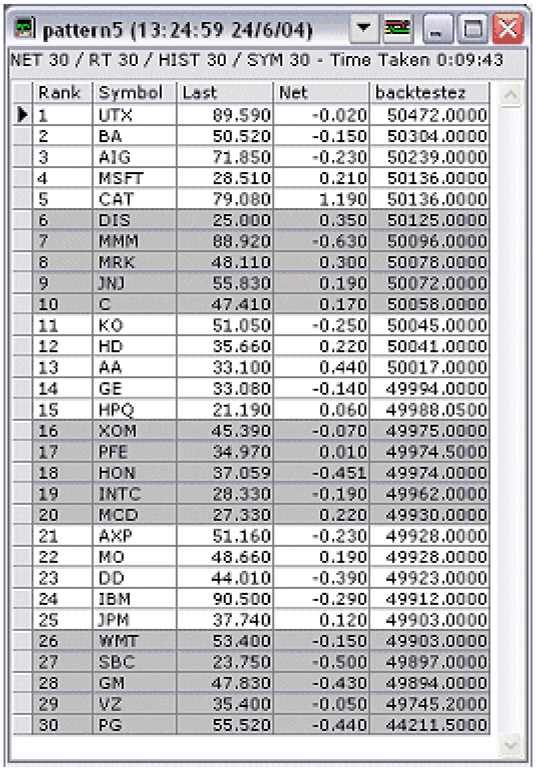
○ Recalc Last 100 Bar Only

On	Rank	Out	Indicator	C1	Value	REL	C2	Value
☑	☑	□	backtestez	NA	25.00	AND	NA	75.00

Press the **Apply and Start Scan** button.

Pattern scanner will apply Backtest EZ on all Dow Jones 30 stocks. This will take a while because pattern scanner is working on 25 days of minute data for 30 stocks. There are close to 300000 bars to work on.

Once done, pattern scanner will return the result of Backtest EZ on 30 stocks, ranked by the equity curve.



NET 30 / RT 30 / HIST 30 / SYM 30 - Time Taken 0:09:43

Rank	Symbol	Last	Net	backtestez
1	UTX	89.590	-0.020	50472.0000
2	BA	50.520	-0.150	50304.0000
3	AIG	71.850	-0.230	50239.0000
4	MSFT	28.510	0.210	50136.0000
5	CAT	79.080	1.190	50136.0000
6	DIS	25.000	0.350	50125.0000
7	MMM	88.920	-0.630	50096.0000
8	MRK	48.110	0.300	50078.0000
9	JNJ	55.830	0.190	50072.0000
10	C	47.410	0.170	50058.0000
11	KO	51.050	-0.250	50045.0000
12	HD	35.660	0.220	50041.0000
13	AA	33.100	0.440	50017.0000
14	GE	33.080	-0.140	49994.0000
15	HPQ	21.190	0.060	49988.0500
16	XOM	45.390	-0.070	49975.0000
17	PFE	34.970	0.010	49974.5000
18	HON	37.059	-0.451	49974.0000
19	INTC	28.330	-0.190	49962.0000
20	MCD	27.330	0.220	49930.0000
21	AXP	51.160	-0.230	49928.0000
22	MO	48.660	0.190	49928.0000
23	DD	44.010	-0.390	49923.0000
24	IBM	90.500	-0.290	49912.0000
25	JPM	37.740	0.120	49903.0000
26	WMT	53.400	-0.150	49903.0000
27	SBC	23.750	-0.500	49897.0000
28	GM	47.830	-0.430	49894.0000
29	VZ	35.400	-0.050	49745.2000
30	PG	55.520	-0.440	44211.5000

In this particular example, this trading system works best when trading UTX Backtest EZ. You can double click on a symbol to view its Backtest EZ performance in the chart.

Tutorial: Monitoring Large Number of Symbols with Symbol List Manager

Symbol List Manager allows you to distribute symbols from a symbol list to multiple function windows.

Symbol List Manager is useful for users who keep large number of identical windows opened to analyze different symbols. Because the Symbol List Manager can read a symbol list from a file, it is also useful for reviewing scanning results.

Prerequisite

You know how to create charts

Goal

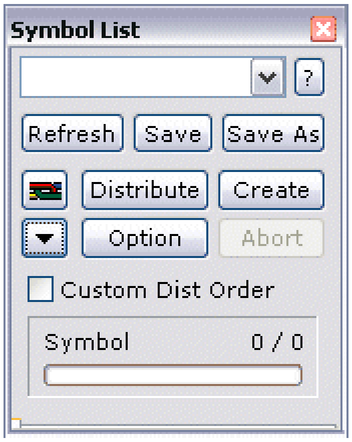
After you finish this tutorial, you will be able to use the symbol list manager.

Setting up the Scenario

Create a chart and add a data series to it. You can also add some indicators to the chart to make it look interesting.

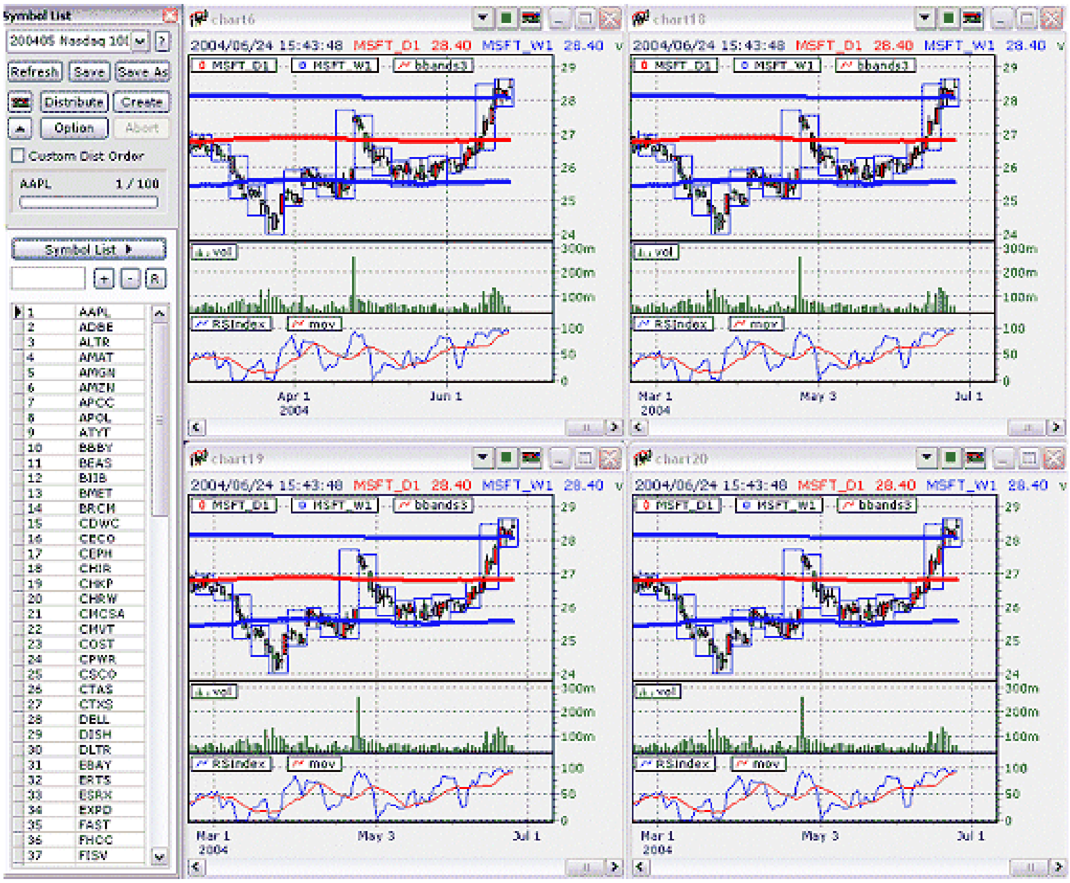
Open the chart's pop up menu by right clicking on the chart. Choose **Copy to New Window**. A new chart with identical setting will be created. Create a few more charts this way.

Open the symbol list manager by choosing **Manager>Symbol List** from the main window.



Choose one of the symbol list from the drop down (e.g. one of the Nasdaq 100 list) and press the ▼ button on the symbol list manager to reveal the symbols. You can resize the symbol list manager to see more symbols.

The set up will look like:



1	AAPL
2	ADBE
3	ALTR
4	AMAT
5	AMGN
6	AMZN
7	APCC
8	APOL
9	ATYT
10	BBBY
11	BEAS
12	BIIB
13	BMET
14	BRCM
15	CDWC
16	CECO
17	CEPH
18	CHIR
19	CHKP
20	CHRW
21	CMCSA
22	CMVT
23	COST
24	CPWR
25	CSCO
26	CTAS
27	CTXS
28	DELL
29	DISH
30	DLTR
31	EBAY
32	ERTS
33	ESRX
34	EXPD
35	FAST
36	FHCC
37	FISV

Distributing Symbols

Press the **Distribute** button in the Symbol List Manager. The symbols from the list will be distributed to the charts you created.

Press the **Distribute** button a few more times. The Symbol List Manager will go through the symbol list and send the symbols to the windows.

By default, the last symbol distributed is repeated in the next distribution to give you context. You can turn off this behavior. Refer to ***Turning off Last Symbol Repeat*** (on page 933).

Symbol distribution performance can be greatly enhanced if the symbols are already in RAM Cache. Refer to ***Tutorial: Turbo Charging Data Loading Speed*** (on page 205).

Tutorial: Working with Groups and Function Windows

Groups and function windows are the basic objects in NeoTicker®. Charts, quotes and time sales are function windows and a group is a collection of function windows. A NeoTicker® session is the collection of your current set of groups.

This tutorial teaches you how to work with groups and function windows. Knowing how to work with them is essential to manage large number of windows in NeoTicker® effectively.

In some demo version of NeoTicker®, there is a limit on the number of function windows you can open. You can open fewer window than this tutorial suggested and still follow the tutorial successfully.

Prerequisite

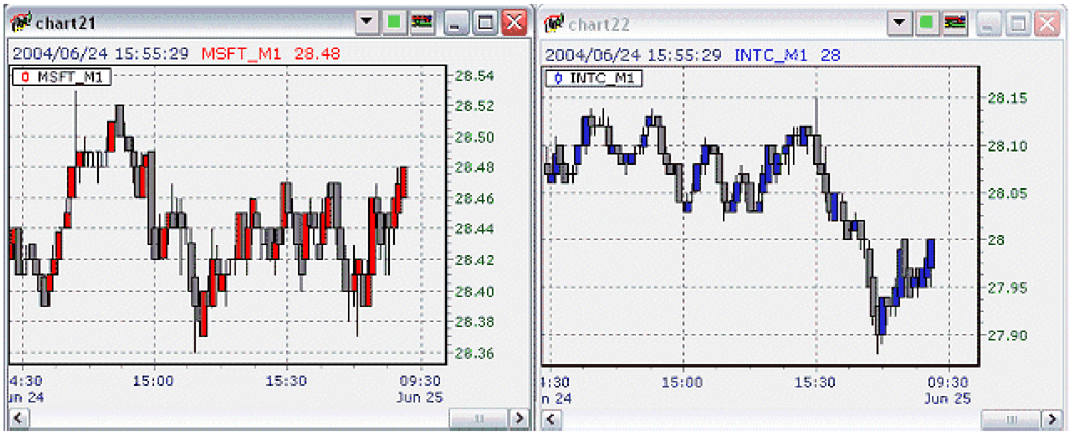
You know how to create charts and quote windows.

Goal

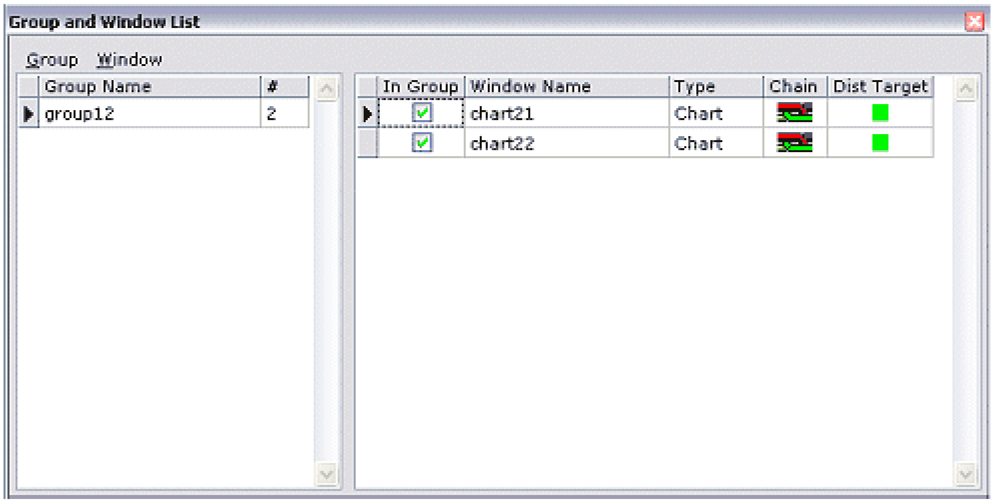
After you finish this tutorial, you will able to organize windows into groups and manage them effectively.

Creating Charts

Create two charts. Add data series to the charts. It does not matter how they look. For example, the charts may look like:



When you create a time chart, a group is automatically created for you. So the second chart is added to that group. To see the group, choose **Group>Group and Window List** from the main window.



	Group Name	#
▶	group12	2

	In Group	Window Name	Type	Chain	Dist Target
▶	☑	chart21	Chart		🟩
	☑	chart22	Chart		🟩

The left list is the group list. It displays all the groups that are currently opened. Currently only one group is open (your group name may be different). The right triangle ▶ marks the active group. Since you only have one group, this is the active group.

The right list is the window list. It displays all the windows that are currently opened. The check under the **In Group** column indicates whether the window belongs to the active group. The right triangle ▶ in the window list marks the active window.

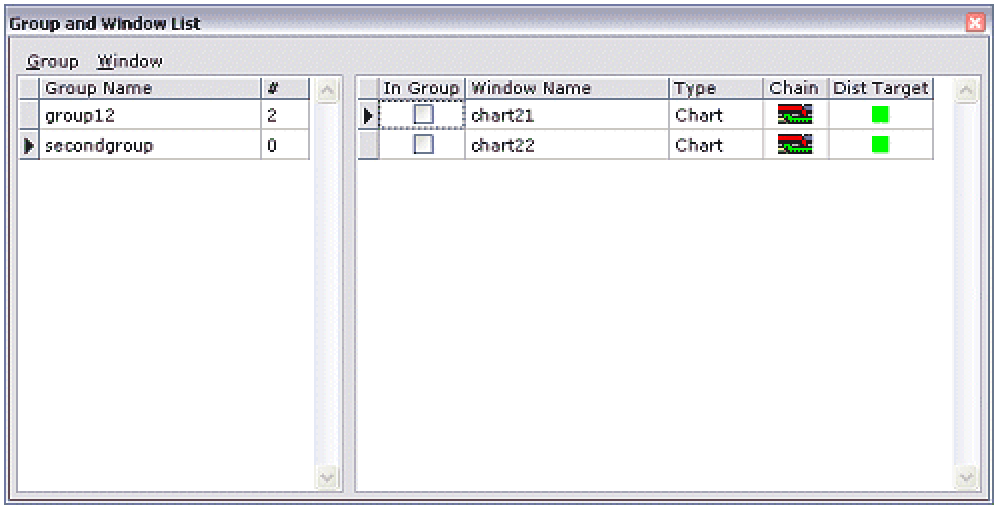
Creating Another Group

Create another group by choosing **Group>New Group**. You can choose the menu either from the main window or from the group and window list window.

You will be prompt for a name for new group. Name the group `secondgroup` and press the **Save** button.

Once `secondgroup` is created, NeoTicker® will switch to `secondgroup` as the active group. The charts you created will be hidden because they are in the first group, but not in `secondgroup`.

In the group and window list, you can observe that `secondgroup` is now the active group and the checks for the charts under **In Group** are not checked.



	Group Name	#
	group12	2
▶	secondgroup	0

	In Group	Window Name	Type	Chain	Dist Target
▶	□	chart21	Chart	▨	🟩
	□	chart22	Chart	▨	🟩

In the group and window list, you can click anywhere in a group row to switch to that group. When you click on the first group, the charts will reappear.

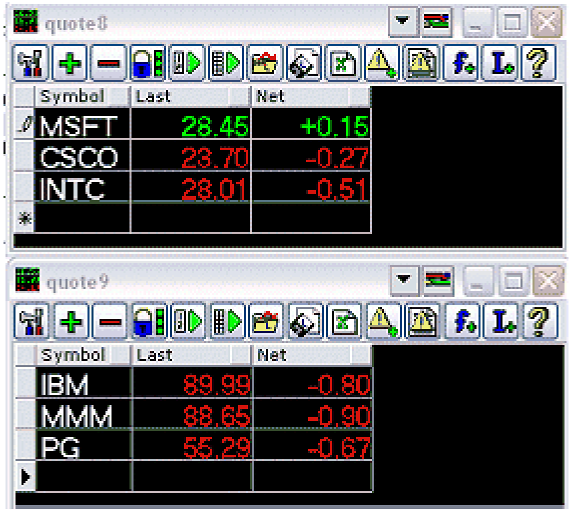
You can also use the PAGE UP and PAGE DOWN keys on your keyboard to flip through the groups.

Adding Windows to the new Group

Make sure you have selected `secondgroup` as the active group. Right now there is no window in `secondgroup`.

Add two quote windows to the new quote and enter some quotes to the quote windows.

The combination of group and window list and the quote window will look something like:



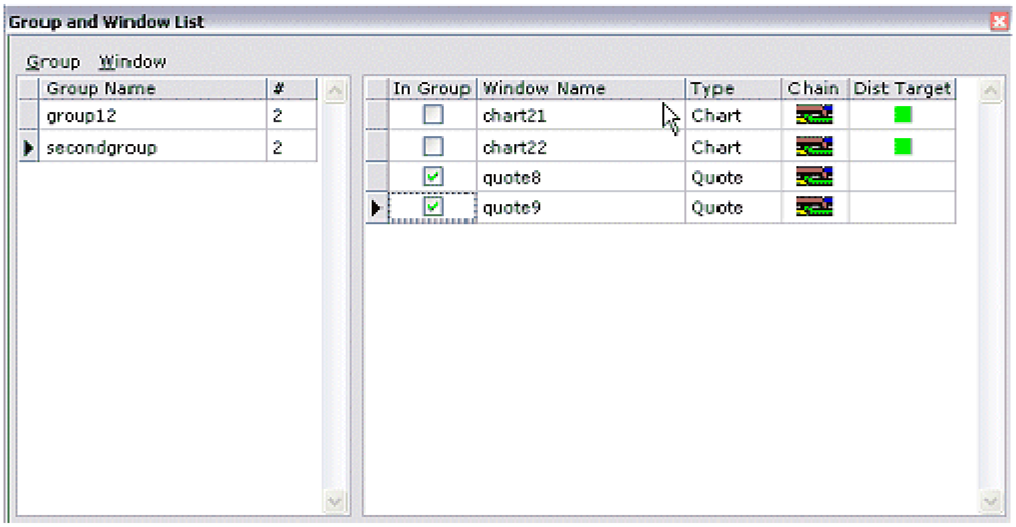
quote8

Symbol	Last	Net
MSFT	28.45	+0.15
CSCO	23.70	-0.27
INTC	28.01	-0.51

quote9

Symbol	Last	Net
IBM	89.99	-0.80
MMM	88.65	-0.90
PG	55.29	-0.67

Note that the quote windows are created under `secondgroup` and the are checked under **In Group** when secondgroup is the active group.



Group and Window List

Group Name	#
group12	2
secondgroup	2

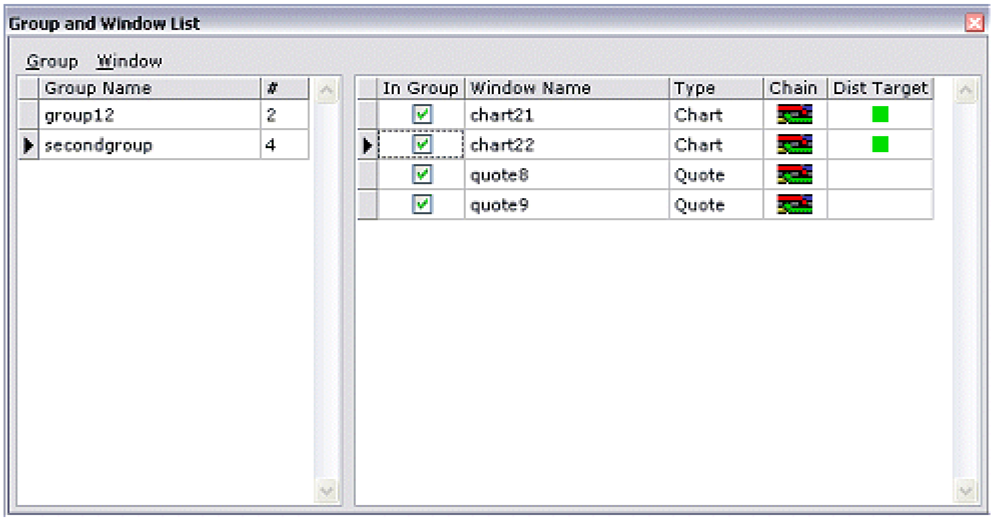
In Group	Window Name	Type	Chain	Dist Target
□	chart21	Chart		🟩
□	chart22	Chart		🟩
☑	quote8	Quote		
☑	quote9	Quote		

Use the PAGE UP and PAGE DOWN keys to flip through the groups and observe the behavior. When the first group is active, the charts will show up. When the second group is active, the quote windows will show up.

Function Window Can Belong to Multiple Groups

A function window can belong to multiple groups.

When the first group is active, the quote windows' checks under **In Group** is off. Check the boxes for the quote window so they belong to both the first and second group.



Group Name	#
group12	2
secondgroup	4

In Group	Window Name	Type	Chain	Dist Target
☑	chart21	Chart	▨	🟩
☑	chart22	Chart	▨	🟩
☑	quote8	Quote	▨	
☑	quote9	Quote	▨	

Now when the first group is active, both charts and quote windows will show up. When the second group is active, only the quote window will show up. Press PAGE UP and PAGE DOWN a few times to see how group switching works.

Tutorial: Constructing a Scan using Pattern Scanner

Pattern scanner scans symbols that match the user specified criteria. Criteria such as real-time price, indicator values and visual patterns can be used. Pattern scanner can scan on timer or on demand.

Pattern scanner is suitable for scanning jobs that calls for:

- Large number of symbols (30000+ symbols with a satellite feed)
- Chaining (one scanner triggers another scanner)
- Spread scanning (multiple symbol scanning)
- Visual pattern scanning

Because pattern scanner is such a general tool, it takes some effort to set up. For simpler scanning jobs, you can consider using quote window as an alternative. Refer to ***Tutorial: Creating a Great Quote Window*** (on page 125) for more details.

If you use an Internet data feed, please also read the section ***Scanning with Internet Data Feed*** (on page 834).

Prerequisite

None.

Goal

After you finish this tutorial, you will be able to use the pattern scanner to scan for net percentage change, and export the scan result.

Setting up the Scenario

Suppose you want to locate the stocks that fit the following criteria:

- Price between $10 to $50
- Has a net price increase of 1% to 5%

Here is the problem: you want to locate the stock among the stocks in the S&P 500, but your data subscription limits you to less than 500 symbols so you cannot use the quote window to find the symbols.

Solution is to use the pattern scanner. In a pattern scanner, a symbol is released after use so it does not use up your symbol limit. There is some performance penality as each symbol has to be requested again from the data vendor when it is scanned.

Create a pattern scanner by pressing the pattern scanner button in the main window.

Setting up Pattern Scanner

You need to setup data and filters make the pattern scanner work.

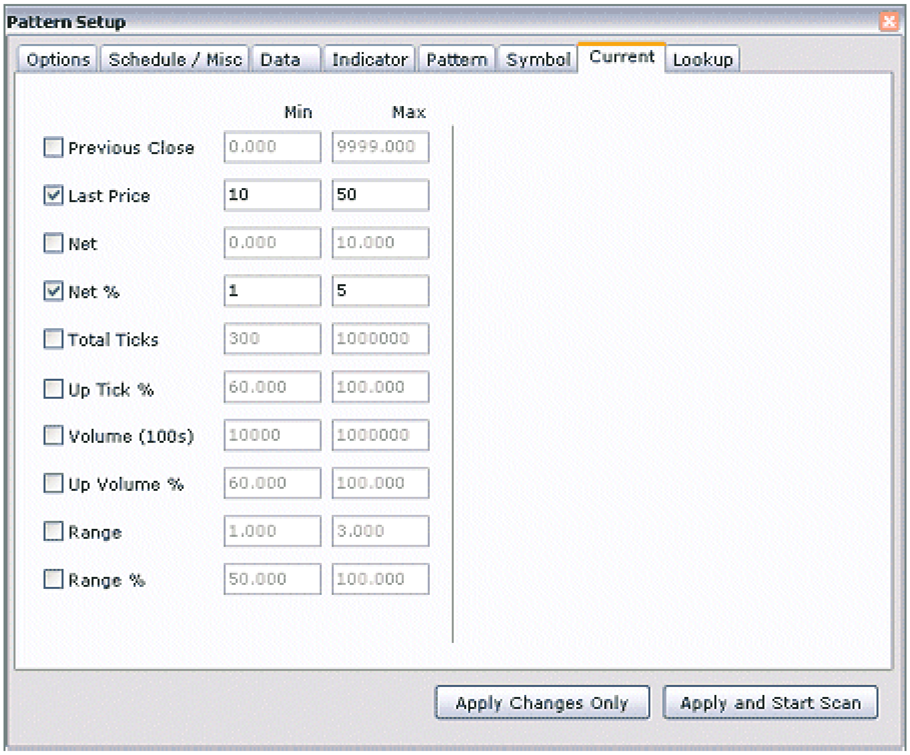
Right click on the pattern scanner to open the pop up menu, and choose **Setup**.

This opens the pattern setup window.

First we will set up the scanning criteria. Press the **Current** tab. Under this tab you can set up filters that are applied to the real-time current values of symbols.

Check **Last Price**, and enter 10 for **Min** and 50 for **Max**. These settings tell the scanner to filter out any symbol that has a price outside of this range.

Check **Net %**, and enter 1 for Min and 5 for Max. These settings tell the scanner to filter out any symbol that has a net % outside of this range.

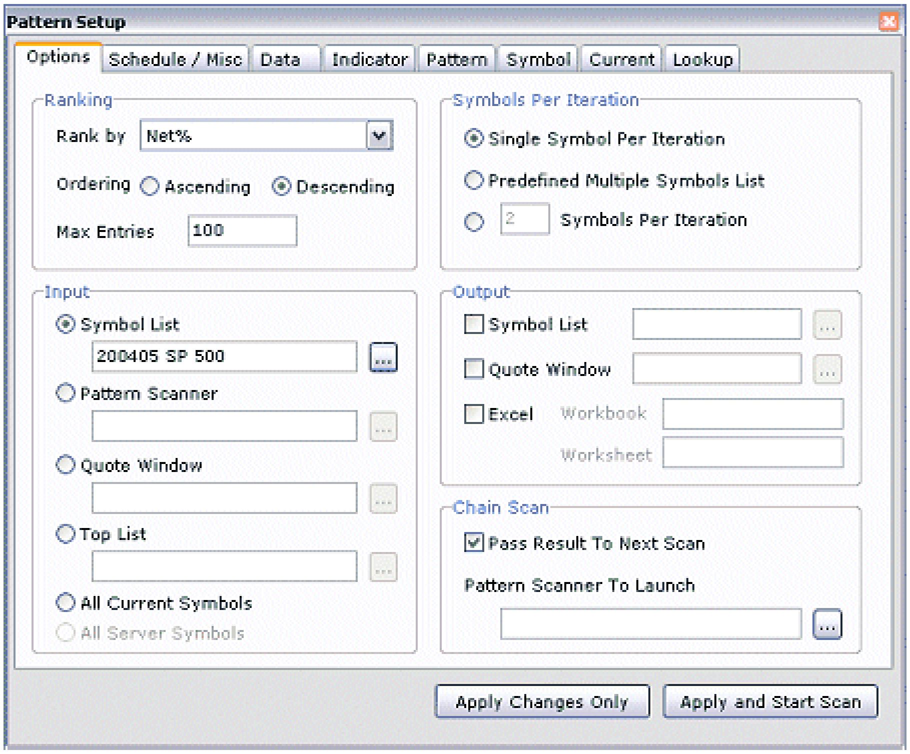


	Min	Max
☐ Previous Close	0.000	9999.000
☑ Last Price	10	50
☐ Net	0.000	10.000
☑ Net %	1	5
☐ Total Ticks	300	1000000
☐ Up Tick %	60.000	100.000
☐ Volume (100s)	10000	1000000
☐ Up Volume %	60.000	100.000
☐ Range	1.000	3.000
☐ Range %	50.000	100.000

The next step is to set up the symbol list and ranking field. Press the **Options** tab.

Change **Rank by** to **Net %**. This tells the pattern scanner to return the result, sorted by the net percentage field.

Check **Symbol List** under **Input**. Press the ... button and choose one of the S&P 500 symbol list. This tells the pattern scanner to scan using the symbols in the S&P 500 symbol list.



Pattern Setup

Options | Schedule / Misc | Data | Indicator | Pattern | Symbol | Current | Lookup

Ranking

Rank by: Net%

Ordering: ○ Ascending ● Descending

Max Entries: 100

Symbols Per Iteration

● Single Symbol Per Iteration

○ Predefined Multiple Symbols List

○ 2 Symbols Per Iteration

Input

● Symbol List: 200405 SP 500 ...

○ Pattern Scanner ...

○ Quote Window ...

○ Top List ...

○ All Current Symbols

○ All Server Symbols

Output

□ Symbol List ...

□ Quote Window ...

□ Excel Workbook

Worksheet

Chain Scan

☑ Pass Result To Next Scan

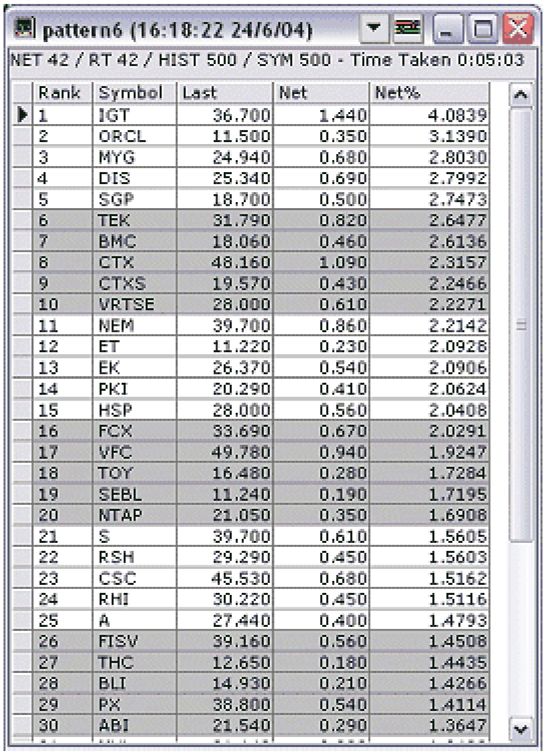
Pattern Scanner To Launch ...

Apply Changes Only | Apply and Start Scan

Running Pattern Scanner

Once the scanning criteria are specified, press the **Apply and Start Scan** button.

Depending on your data feed, this scanning job can take a few seconds to several minutes. Once it is done, the pattern scanner window displays the result of the scan. The result are the symbols that match the criteria you specified in the setup window.



NET 42 / RT 42 / HIST 500 / SYM 500 - Time Taken 0:05:03

Rank	Symbol	Last	Net	Net%
1	IGT	36.700	1.440	4.0839
2	ORCL	11.500	0.350	3.1390
3	MYG	24.940	0.680	2.8030
4	DIS	25.340	0.690	2.7992
5	SGP	18.700	0.500	2.7473
6	TEK	31.790	0.820	2.6477
7	BMC	18.060	0.460	2.6136
8	CTX	48.160	1.090	2.3157
9	CTXS	19.570	0.430	2.2466
10	VRTSE	28.000	0.610	2.2271
11	NEM	39.700	0.860	2.2142
12	ET	11.220	0.230	2.0928
13	EK	26.370	0.540	2.0906
14	PKI	20.290	0.410	2.0624
15	HSP	28.000	0.560	2.0408
16	FCX	33.690	0.670	2.0291
17	VFC	49.780	0.940	1.9247
18	TOY	16.480	0.280	1.7284
19	SEBL	11.240	0.190	1.7195
20	NTAP	21.050	0.350	1.6908
21	S	39.700	0.610	1.5605
22	RSH	29.290	0.450	1.5603
23	CSC	45.530	0.680	1.5162
24	RHI	30.220	0.450	1.5116
25	A	27.440	0.400	1.4793
26	FISV	39.160	0.560	1.4508
27	THC	12.650	0.180	1.4435
28	BLI	14.930	0.210	1.4266
29	PX	38.800	0.540	1.4114
30	ABI	21.540	0.290	1.3647

If you want to stop the scanning job, choose **Stop Scan** from the pop up menu. You can start a scanning job without the set up window by choosing **Start Scan** from the pop up menu.

Exporting Scanning Result

To send the scanning result to another window, from the pop up menu, choose **Send List to** and the window you want to send to.

You can also export the scanning result to Microsoft Excel. Choose **Export Table>Excel** from the pop up menu to export to Microsoft Excel.

Tutorial: Turbo Charging Data Loading Speed

There are many settings that can greatly improve data loading performance. These settings are not on by default because they are usage and data feed specific.

For example, the pre-load symbol list feature let you make the trade off between start up time and data loading speed. The performance boost for a typical minute chart can be a loading time decreasing from several seconds to almost instantaneous.

Prerequisite

None.

Goal

After you finish this tutorial, you will learn techniques that can greatly improve performance.

Pre-Load Symbol List

How does Pre-Load Symbol List Work

The pre-load symbol list is a list of symbols that are loaded into the RAM Cache when NeoTicker® starts, or when NeoTicker® is performing a morning restart. RAM Cache is a place to hold existing symbols to receive real time data. Because initial data requesting is already done for symbols in the RAM Cache, accessing symbols already in RAM Cache is very fast.

Pre-loading a symbol list has a trade off. Pre-loading takes time and if the symbol list is long, pre-loading will take time. This is an issue to consider if you do not run NeoTicker® overnight.

Another issue is memory. As the data is stored in RAM and usage will grow during trading hours, you can use up your computer's memory without realizing it. If you experience stability problem after using pre-load symbol list, you should consider reducing the number of symbols to pre-load.

Specifying Pre-Load Symbol List

To specify the pre-load symbol list:

1 Open the cache manager by choosing **Manager>Cache** from the main window.

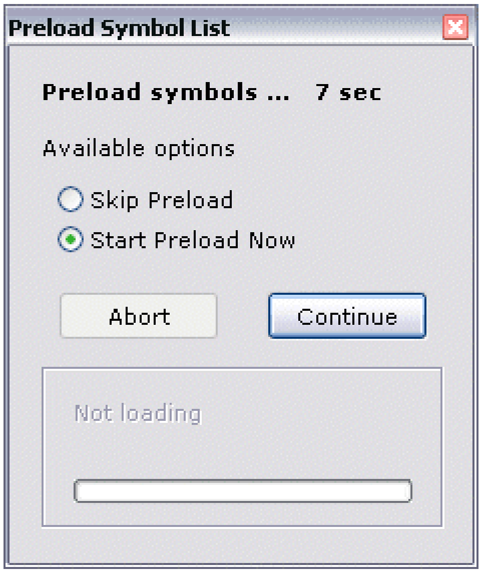
2 Cache Manager is opened. Press the **RAM Cache** tab.

3 Press the **Options** button.

4 The RAM Cache Options window is opened. Choose **Use Symbol List** and press the ... button to choose a symbol list

To start pre-loading symbols in the current session press the **Add Symbols to RAM Cache from Designated Symbol List Now** button. A dialog is opened, make sure **Start Preload Now** is selected and press the **Continue** button. The same dialog will be opened when you restart NeoTicker®.

The symbols in the symbol list will be added to RAM Cache one-by-one. This may take some time. Once this is finished, open a chart and load a stock that belongs to Dow Jones 30. You will notice immediate improvement to loading speed.



Preload Symbol List

Preload symbols ... 7 sec

Available options

○ Skip Preload

● Start Preload Now

Not loading

If your data feed has a symbol limit, you should not specify symbols that you do not need. The symbols in the RAM Cache are counted towards your symbol limit.

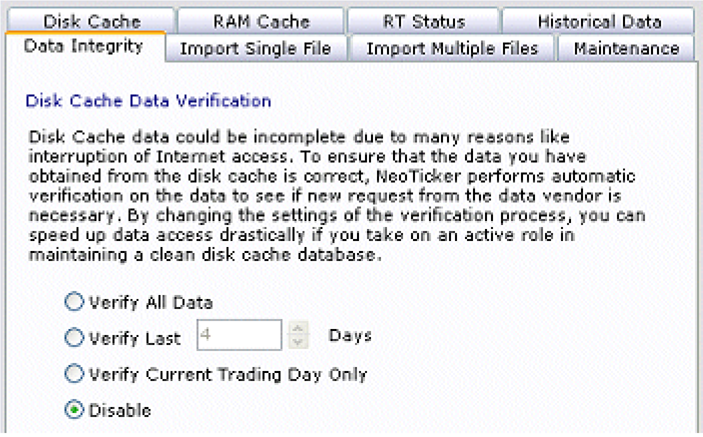
Turning off Cache Verification

You can turn off the cache verification option to improve data loading speed. This trick is applicable to Internet data feed with historical database.

1 Choose **Manager>Cache**.

2 The cache manager will be opened. Click the **Data Integrity** tab.

3 Choose **Disable**. Once cache verification is disabled NeoTicker® will not verify cached data file on disk and data loading performance will be improved. For a full explanation of what this option does, go to ***Data Verification between Disk Cache and Data Feed*** (on page 449).



Disk Cache Data Verification

Disk Cache data could be incomplete due to many reasons like interruption of Internet access. To ensure that the data you have obtained from the disk cache is correct, NeoTicker performs automatic verification on the data to see if new request from the data vendor is necessary. By changing the settings of the verification process, you can speed up data access drastically if you take on an active role in maintaining a clean disk cache database.

○ Verify All Data

○ Verify Last 4 Days

○ Verify Current Trading Day Only

● Disable

Fill up Your Disk Cache at Night

This section is applicable only if you are using an Internet feed with historical data server.

NeoTicker® can chart a lot faster if the data is already in the disk cache because less data is requested from the server. If you track a lot of different symbols, you can download data nightly to the disk cache. The next day, you will have last trading day's data in disk cache.

For more information, refer to ***Retrieving Data from Data Vendor Every Night*** (on page 456).

Auto Insert Symbols in RAM Cache

See ***Automatically Insert Symbols into RAM Cache*** (on page 464).

Tutorial: EOD Data Speed Improvement

This tutorial is for users who use Internet EOD Data. This tutorial is not applicable if you use TC2005, Quotes Plus or a real-time feed for data.

If you work on a large number of stocks using Internet EOD Data, this section is for you. You can improve NeoTicker® data access speed by actively managing EOD data. This tutorial will show you how.

Prerequisite

You should already have NeoTicker® installed and running. You know how to create a chart.

Goal

After you finish this tutorial, you will:

- Understand how NeoTicker® works with Internet EOD Data.
- Utilizing this knowledge to improve data access speed.

Overview

When you chart a stock, say MSFT, you will notice there is a slight delay (typically 1-2 second, slower if your Internet connection is slow) before the chart is completed.

This delay is caused by accessing the Internet to download data. If you chart the same symbol again, this delay can be gone - NeoTicker® already stores the data locally in your hard drive so it is very fast to chart the same symbol again.

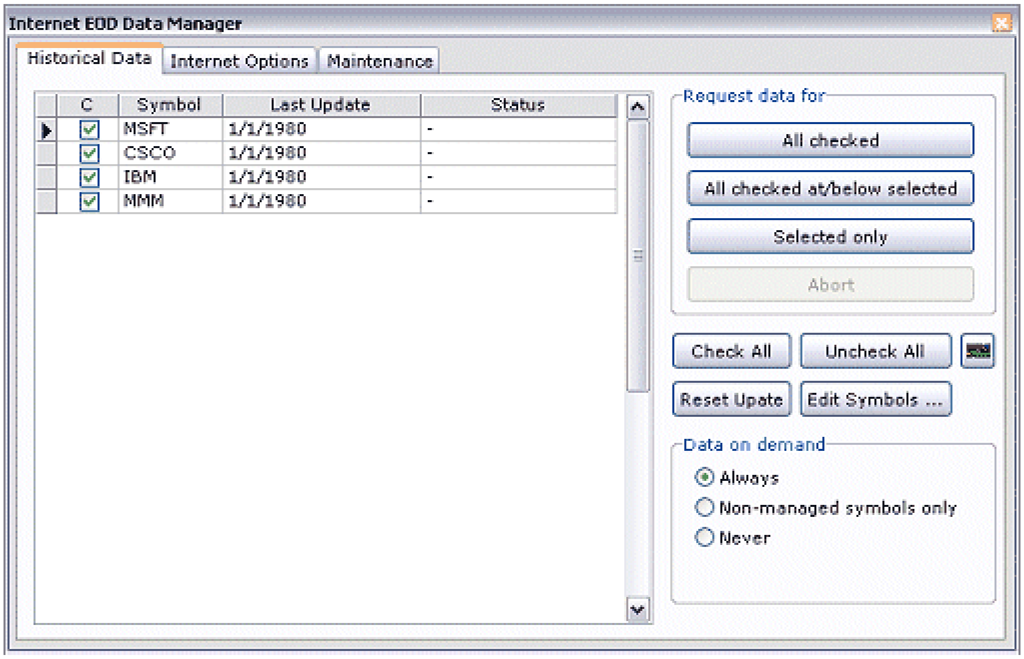
Often enough the delay will still be there. What happened? NeoTicker® decides data stored in your hard drive is not the most updated and tries to download data again from the Internet. Due to the nature of the Internet and availability of data, NeoTicker® can be making the wrong decision - there is no need to download data again.

You can tell NeoTicker® not to download data so you can work faster without waiting for data download.

Internet EOD Data Manager

First, create a chart for DELL. Observe the delay on your system. You are going to manage the data for DELL so you get a speed improvement.

Now open Internet EOD Data Manager by choosing **Manager>Internet EOD Data** from the main window. The Internet EOD Data Manager is opened. Make sure you are under the **Historical Data** tab. Select the **Non-managed symbols only** option under **Data on Demand**. This will tell NeoTicker® not to load data from Internet unless the symbol is not managed by you.

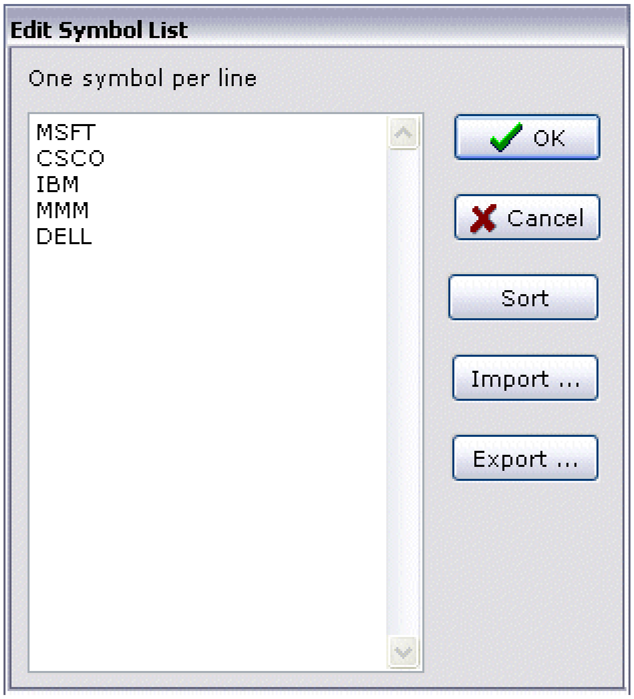


C	Symbol	Last Update	Status
☑	MSFT	1/1/1980	-
☑	CSCO	1/1/1980	-
☑	IBM	1/1/1980	-
☑	MMM	1/1/1980	-

Data on demand

● Always
○ Non-managed symbols only
○ Never

Note that by default, there are 4 symbols already entered for you: MSFT, CSCO, IBM and MMM, but DELL is not one of them. Press the **Edit Symbols** button. You will open an editor to let you enter stock symbols. Enter DELL to the editor (one symbol per line), then press the **OK** button.



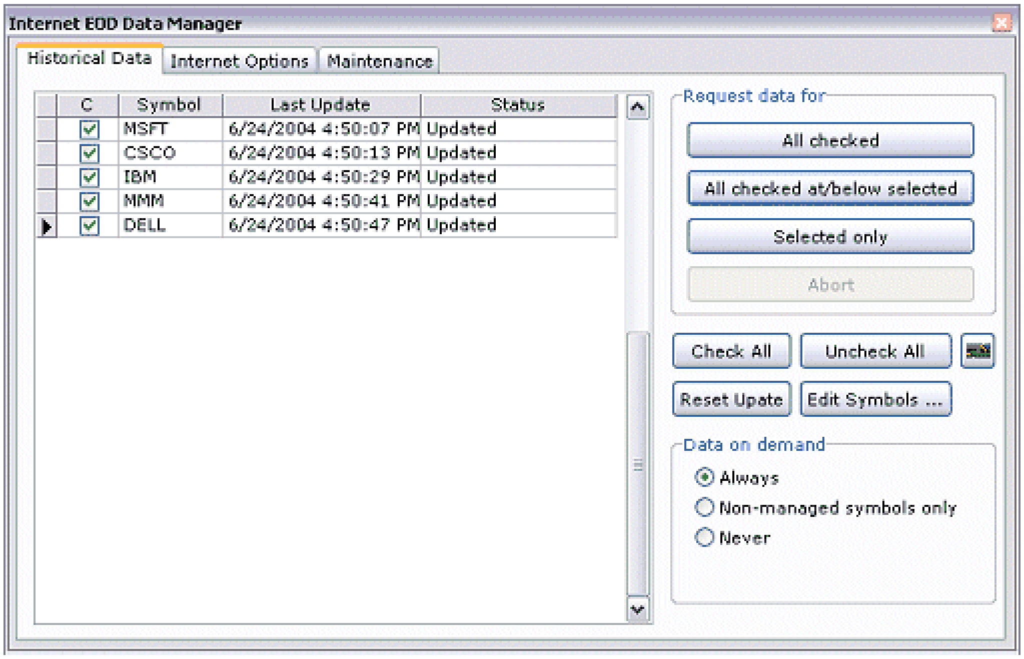
Edit Symbol List

One symbol per line

```
MSFT
CSCO
IBM
MMM
DELL
```

In Internet EOD Data Manager, you will notice the symbols you entered appear in the window. Press the **All Checked** button under **Request data for**. You will notice Internet EOD Data Manager will starting updating. What happen is Internet EOD Data Manager will start downloading data from the Internet.

Tip: requesting data for multiple symbols is fastest after trading hour in North America.



	C	Symbol	Last Update	Status
	☑	MSFT	6/24/2004 4:50:07 PM	Updated
	☑	CSCO	6/24/2004 4:50:13 PM	Updated
	☑	IBM	6/24/2004 4:50:29 PM	Updated
	☑	MMM	6/24/2004 4:50:41 PM	Updated
▶	☑	DELL	6/24/2004 4:50:47 PM	Updated

Data on demand
- ● Always
- ○ Non-managed symbols only
- ○ Never

Open a new chart, chart DELL again. The chart for DELL will come up almost instantly. In fact, if you chart any of MSFT, CSCO, IBM, MMM or DELL, the chart will come up almost instantly. This is because the data is now always accessed from your hard drive, not from the Internet.

Don't close Internet EOD Manager yet. We will show you a trick.

In Internet EOD Data Manager, double click on another symbol. Notice that your chart will be change to that symbol. Again, charting is almost done instantly. You can also use the Enter key on your keyboard to rotate through the symbols.

Now chart a symbol that is not listed in Internet EOD Manager. Notice that there will be a delay. The symbol is not managed, so NeoTicker® is loading data on demand from the Internet.

Tomorrow (or a few days later)

When a symbol is managed by Internet EOD Data Manager, NeoTicker® will not try to download data on demand. So by tomorrow, the latest data is not available. You can simply use Internet EOD Data Manager to retrieve the latest data again.

1 Open Internet EOD Data Manager.

2 Press the **All checked** button under **Request data for**.

Requesting data will automatically request data long enough to fill data cache for the managed symbol. So even if you haven't request data for months, as long as the data is still available on the Internet (they usually are), you will be able to request it.

For more information about Internet EOD Data Manager features, refer to ***Internet EOD Data Manager Operation Guide*** (on page 607).

Tutorial: EOD Commodities Chart

This tutorial is for users who use Internet EOD Data. This tutorial is not applicable if you use TC2005, Quotes Plus or a real-time feed for data.

NeoTicker® can retrieve free commodities (e.g S&P 500 e-mini futures contract) data from the Internet for charting purpose. Automatic rollover is supported for ES, NQ and YM contracts. Commodities chart work exactly the same as stock charts.

Prerequisite

You should already have NeoTicker® installed and running. You know how to create a chart.

Goal

After you finish this tutorial, you will:

- Know how to chart commodities data

Creating a Commodities Chart

The key thing of charting a commodity is getting the symbology right. The general format for a commodities symbol is:

```
@XXYYM
```

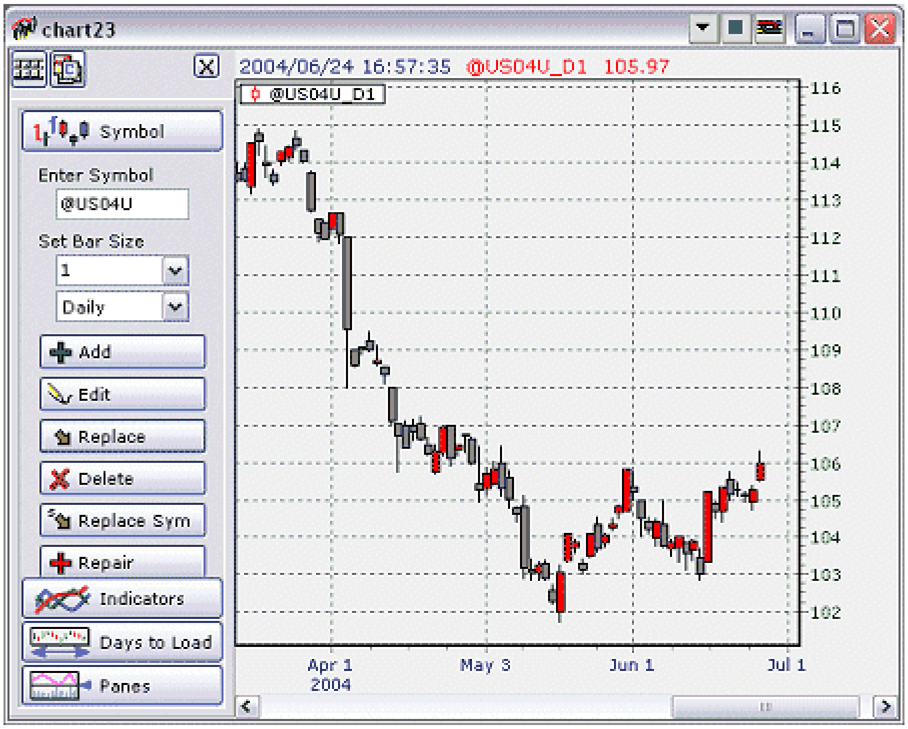
where XX is the symbol, YY is 2-digit year, M is month code.

For example, 2004 March 30-year US treasury bond symbol is `@US04H`. 2004 June S&P 500 e-mini contract is `@ES04M`.

You cannot chart expired contracts unless you have chart the expired contract before. The data is no longer available from the Internet so if you have chart expired contract before, the data is still on your hard drive.

Charting using User Panel

If you use User Panel to add data, simply enter the commodities symbol and press the **Add Data** button. The following chart is @US04U.

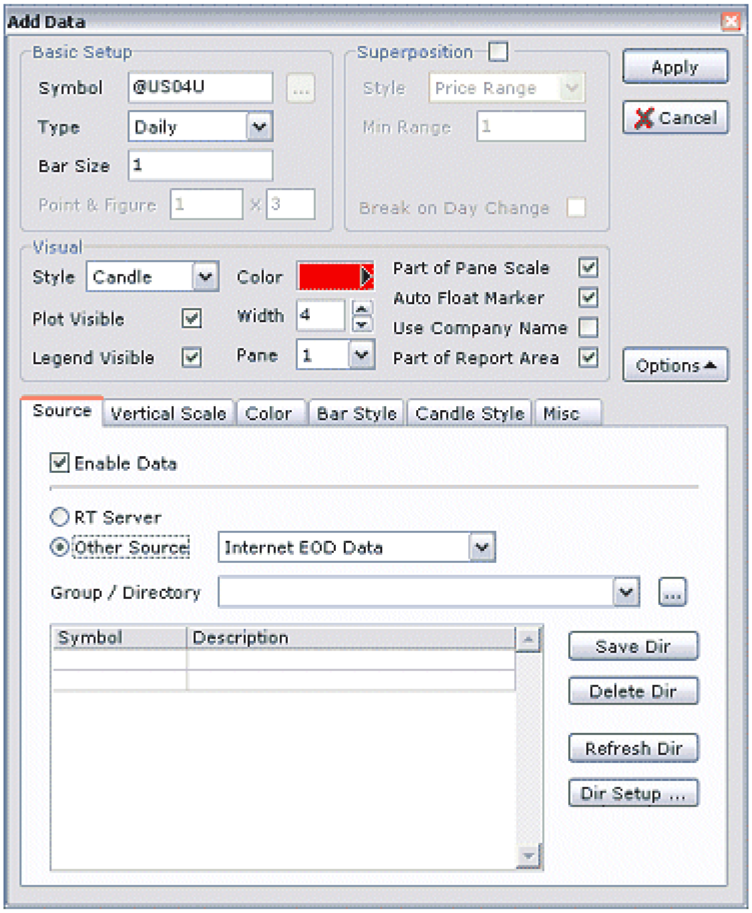


Enter Symbol: @US04U

Set Bar Size: 1, Daily

Charting by using Add Data Dialog

If you use Add Data dialog to add data, specify the data source to **Internet EOD Data** for commodities data (you may need to press the **Options** button to see the complete dialog).


Add Data

Basic Setup
Symbol @US04U
Type Daily
Bar Size 1
Point & Figure 1 x 3

Superposition □
Style Price Range
Min Range 1
Break on Day Change □

Visual
Style Candle
Color 🟥
Width 4
Pane 1
Plot Visible ☑
Legend Visible ☑
Part of Pane Scale ☑
Auto Float Marker ☑
Use Company Name □
Part of Report Area ☑

☑ Enable Data
○ RT Server
● Other Source Internet EOD Data
Group / Directory

Symbol	Description

Automatic Rollover Contracts

NeoTicker® provides automatic rollover symbol for the following contracts - ES, NQ, YM. You can simply type `@ES`, `@NQ` and `@YM` for these contracts.

Automatic rollover requires you to have expired contract data already on your hard drive (NeoTicker® already comes with some expired contract data). Basically if you have been charting these symbol on a day-to-day basis, data is already managed for you and you do need to worry about expired contract data.

Tutorial: Organizing Indicators

NeoTicker® has over 150 build-in indicators. If you write your own indicators, the number can be even larger. This makes finding the right indicator to use sometimes a difficult task.

To solve the problem, NeoTicker® provides a way to organize indicators into indicator lists. An indicator list is collection of indicators. By looking at a specific list, you will only look at indicators that you need.

Prerequisite

You have used indicators in charts, quote windows, etc.

Goal

After you finish this tutorial, you will know how to organize indicators, putting indicators you used most in a list.

Indicator List Manager

Overview

The indicator list manager is the place where you organize the indicators into indicator lists.

Indicator list is just a way of organizing indicators. There is a special indicator list called **all**. The **all** list is always available and lists all the indicators. You cannot add or remove indicators from the **all** list. You cannot remove the **all** list.

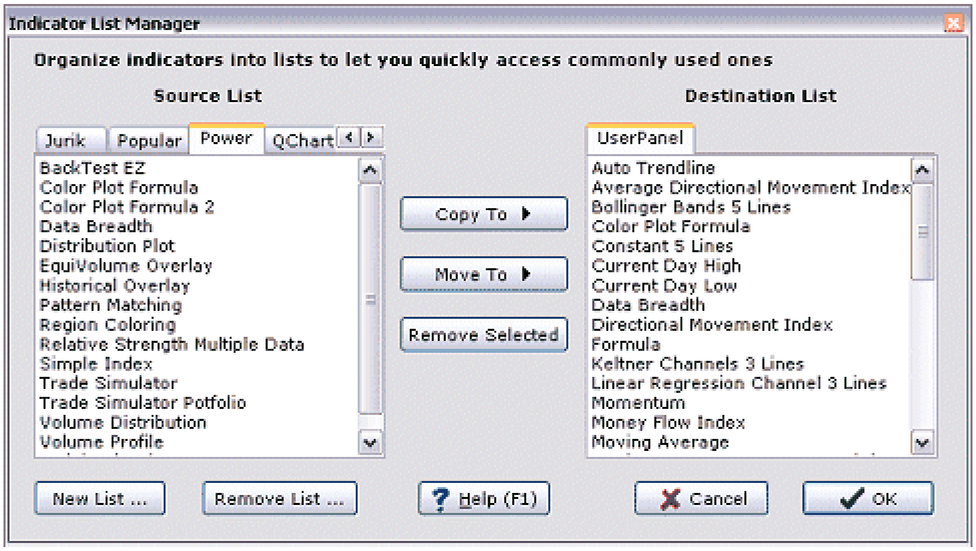
Other indicator list either comes with NeoTicker®, or are user created. The list are for you to organize indicators into different lists. You can freely create and remove these indicator list. Indicators can be added and remove from the indicator list and indicators can belong to more than one indicator list.

Indicator list is just a management tool. Indicator list does not affect the actual indicators.

Opening Indicator List Manager

To open indicator list manager:

- Choose **Manage>Indicator List** in the main window, or
- Press the **Manage** button. The **Manage** button is visible almost always when a list of indicators are shown.


Indicator List Manager

Organize indicators into lists to let you quickly access commonly used ones

Source List

BackTest EZ
Color Plot Formula
Color Plot Formula 2
Data Breadth
Distribution Plot
EquiVolume Overlay
Historical Overlay
Pattern Matching
Region Coloring
Relative Strength Multiple Data
Simple Index
Trade Simulator
Trade Simulator Potfolio
Volume Distribution
Volume Profile

Destination List

Auto Trendline
Average Directional Movement Index
Bollinger Bands 5 Lines
Color Plot Formula
Constant 5 Lines
Current Day High
Current Day Low
Data Breadth
Directional Movement Index
Formula
Keltner Channels 3 Lines
Linear Regression Channel 3 Lines
Momentum
Money Flow Index
Moving Average

Creating Indicator List

Press the **New List** button in indicator list manager.

Copying and Moving Indicators between Indicator Lists

In the indicator list manager, the left side is the source indicator list and the right side is the destination indicator list. When you copy or move indicators, the indicators are always moved from the left to the right.

To copy or move an indicator, select the indicator and press the **Copy To** or **Move To** buttons.

Removing Indicator List

To remove an indicator, select the indicator at the left and press the **Remove List** button.

Removing an indicator from an indicator list does not delete the indicator from NeoTicker®. The operation only makes the indicator unavailable to the indicator list.

Because you cannot add or remove indicators into the **all** indicator list. Therefore the **all** list is not shown at the right side.

Tutorial: Introduction to Power Indicators

NeoTicker® completely redefines what an indicator can do. NeoTicker®'s indicators can perform calculations that were possible previously only if you write your own charting application.

This tutorial goes through one of the power indicator: Weighted Index.

Prerequisite

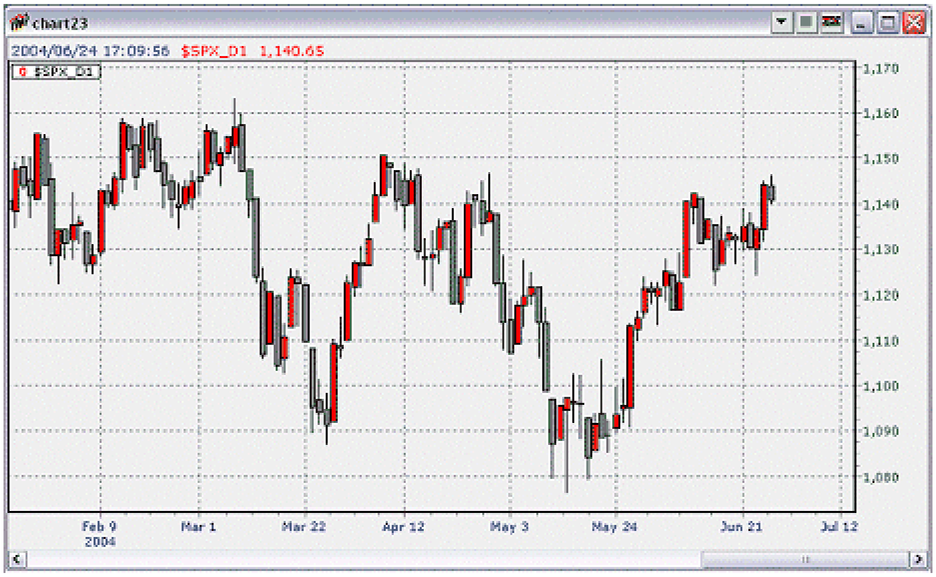
You know how to create a chart and add indicators to the chart.

Goal

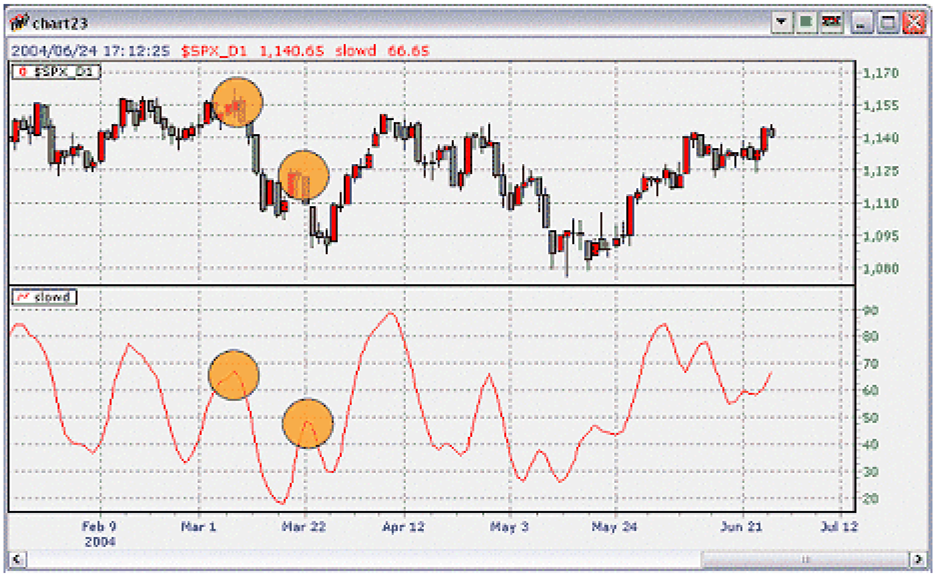
This tutorial goes through the usage of the power indicator Weighted Index.

Weighted Index

Lets say you are interested in analyzing the market condition. You created a new chart with the SP 500 index (our example uses $SPX, your data vendor may use a different symbol for S&P 500 index).

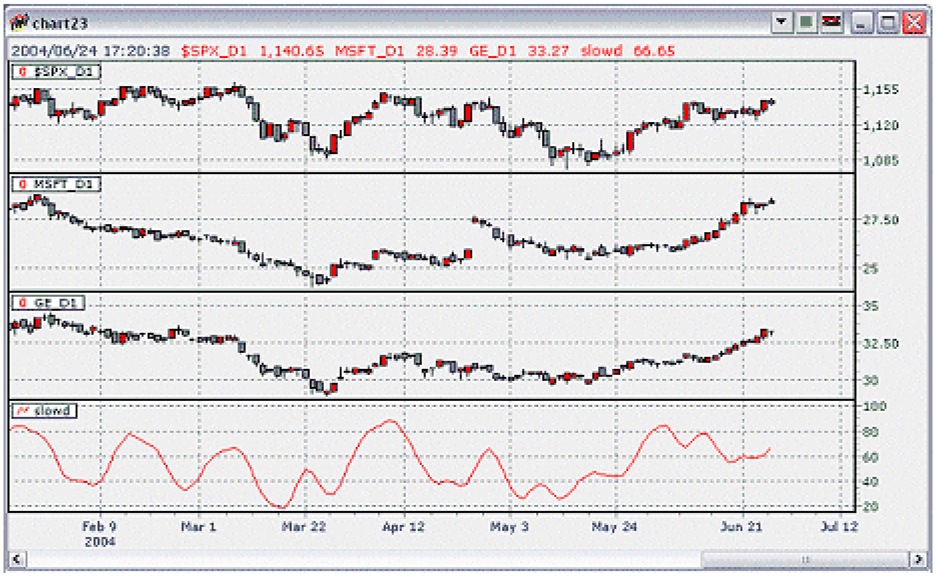


Add the stochastic SlowD indicator to the chart to check for overbought/oversold conditions.

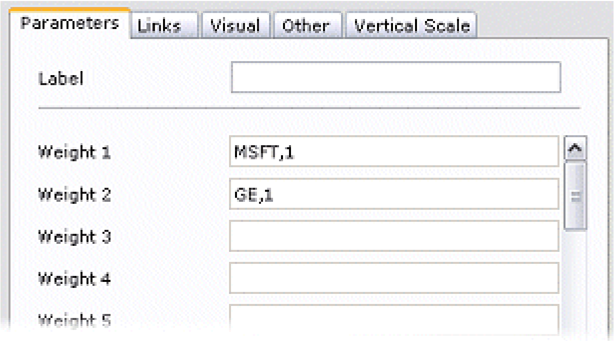


Notice that the SlowD does not really gives signals on time. At best it is a coincident indicator which turns when the price turns. Check the marked spots. By the time SlowD turns, price has already turned.

Lets see how we can improve SlowD performance by using our own index. Add two more symbols to the chart – GE and MSFT.

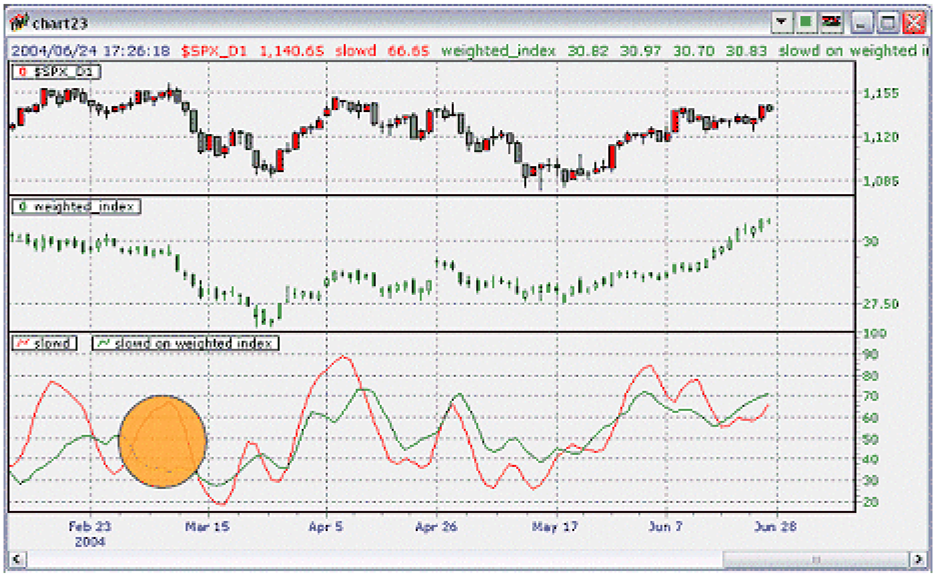


Add the Weighted Index indicator to the chart, with Weight settings as follows.



Label	
Weight 1	MSFT,1
Weight 2	GE,1
Weight 3	
Weight 4	
Weight 5	

Then hide the GE and MSFT series make the chart more readable. Also add SlowD onto the Weighted Index.



2004/06/24 17:26:18 $SPX_D1 1,140.65 slowd 66.65 weighted_index 30.82 30.97 30.70 30.83 slowd on weighted i

Weighted Index is a meta style indicator that makes the indicator behaves like a data series. Thus weighted index is displayed as candlesticks, and you can use weighted index just as if it is a data series.

The SlowD on S&P 500 and our weighted index usually closely match each other in terms of movements. However, on the marked spot in the chart, we see a divergence between the two - both value and direction goes the different way.

Since weighted index is based on MSFT and GE, two heavy weight components in the S&P 500, SlowD on their weighted index (green line) indicates these two have already topped. The rest of the component hasn't catch up yet, so the SlowD on the S&P 500 index (red line) has not topped.

The divergence suggests that either MSFT and GE have to go up to match the rest of the S&P 500, or S&P 500 has to go down to match the two. In this case, it is S&P 500 going down.

You can develop trading strategies based on this observation. Note that the divergence can also work against you (MSFT and GE catch up to S&P 500). So it is important to set stop loss strategy when this happens.

Want More

There are more power indicators with usage description in the section ***Power Indicators Guide*** (on page 1273).

Tutorial: Tool Bar Customization

Tool bars in NeoTicker® can be customized and you can pick any buttons to be included into a particular tool bar.

Prerequisite

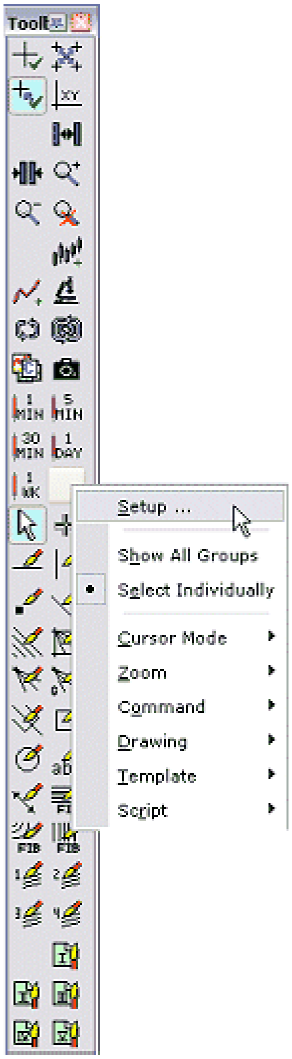
You know how to open a tool bar for a time chart.

Goal

You will know how to customize a tool bar.

Opening Tool Bar Setup Window

To setup your tool bar, choose **Setup** from the right-click pop up menu on the tool bar.



The tool bar setup window will be shown with your current settings. You can customize the buttons available in the tool bar.

Setting Time Chart Tool Button Groups

The **Time Chart** tab is used for setting the time chart related buttons. You can modify each group individually and adjust the individual button settings.

Suppose you want to hide the Un Zoom button in the tool bar.

1. Click on the **Select Individually** radio button under the **Zoom on** button group.
2. Un-check the check box beside **Un Zoom**.

Check the tool bar, the Un Zoom button is not longer available.

Tutorial: Visualize Trading Idea with Sub Series Indicator and Meta Plot Style

Visualization can be a very powerful proof of concept tool for trading ideas. In this tutorial, we will take a trading idea, and apply a series of analysis on the idea. We will be using the Sub Series indicator and meta plot style in this tutorial.

Prerequisite

You know how to create a chart and add indicators to the chart. You need to know how to work with panes. Some familiarity with formula helps.

Goal

This tutorial shows one way to visualize a trading concept in a chart.

Trading Idea

Suppose you have this trading idea: if price is going up, it's time to go long to follow the trend.

How do you tell if this idea makes sense?

One way to verify a trading idea is by testing the idea with a trading system, a mechanical trading model that follows rigid trading rules. However, writing a trading system can be a complex and time consuming task. We want a fast way to proof the concept without programming. Visualizing the idea with a time chart is a good way for proof of concept.

Making the Idea Concrete

First we need to turn the trading idea into something that can be visualized. In a chart, we need to represent the idea using bars.

There are two themes in this trading idea: price is going up, and long to follow the trend. We need to turn these two themes into something that can be represented by bars.

First theme is price is going up. There are many ways to define this theme. We will use the following definition:

- If the current bar's open price is higher than previous bar's close price, price is going up.

Second theme is long to follow a trend. A long is successful if it buys early in a low price, and sells later in a higher price. So for a successful long, we use the following definition:

- If the current bar's open price is lower than the current bar's close price, a long is successful. Otherwise, the long is a failure.

Isolating Pricing Going Up Bars

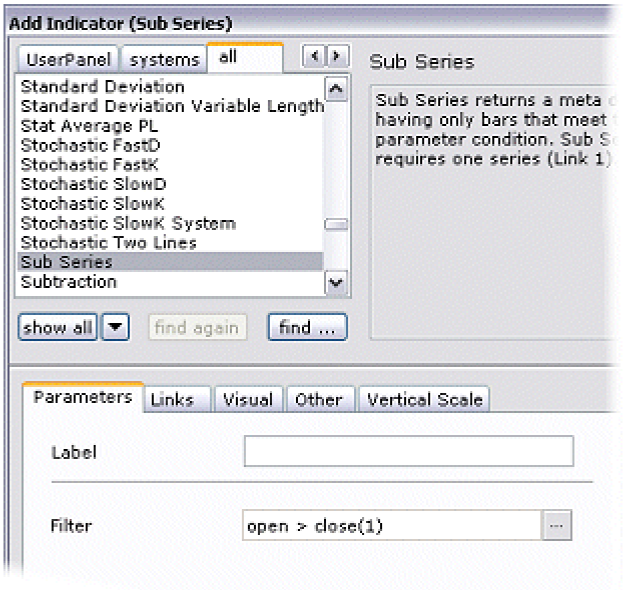
Let's isolate the bars that are going up. We will create a chart, and use the Sub Series indicator to do the isolation.

1 Create a time chart.

2 Add MSFT to the chart. You can use any time frame. In this example, we are using 1-minute with 5 days of data. If you do not have access to real-time data, you can use daily data.

3 In the chart, choose **Add Indicator** from pop up menu. Choose the Sub Series indicator. You can find the Sub Series indicator under the **All** tab when you add indicator. For filter parameter, enter:

```
open > close(1)
```



Sub Series returns a meta
having only bars that meet
parameter condition. Sub S
requires one series (Link 1)

Filter: open > close(1)

4 Under the **Visual** tab, set:

Pane to **New Pane**

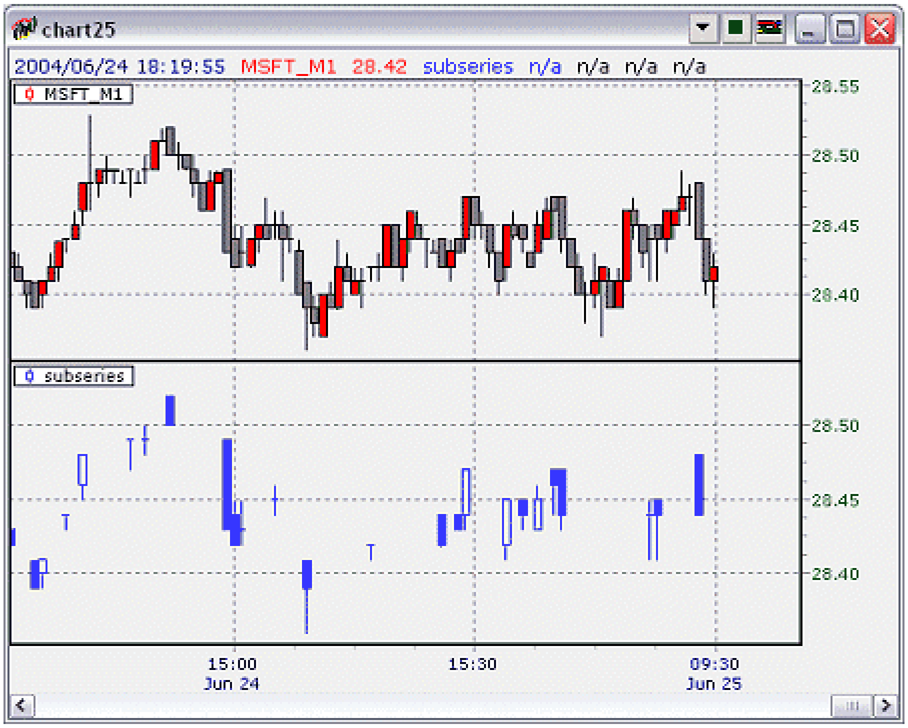
Style (the setting just under **Displace** in the upper right corner) to **Candle**

Color to blue

Width to 4

5 Press the **Apply** button.

The chart should something like:



What Happened

You have created a sub series based on MSFT. The sub series has the same bar as MSFT when the bar's open price is larger than the close price of 1 bar ago(set using the `formula open > close(1)`). In another words, the sub series contains only bars that have the price going up.

The sub series is actually an indicator. But when you specify the **Style** as a **Candle**, time chart will treat the sub series exactly like data. This way of converting an indicator to data series is known as Meta Plot Style.

Isolating Successful and Failed Longs

We will isolate the successful and fail longs from the sub series. We will do this by apply sub series on the sub series.

1 In the chart, click on legend (the upper left white rectangle) of the blue sub series. This will select the sub series and it will become yellow.

2 Open pop up menu, and choose **Add Indicator**.

3 In indicator setup, choose Sub Series, with the following settings:

Filter set to `close > open`

Pane set to **New Pane**

Style set to **Candle**

Color set to Green

Width set to 4

4 Press **Apply** button.

The green series contains the isolated up bars from the sub series. Recall that we consider these up bars successful longs. We will isolate the down bars from the sub series for the failed longs.

1 In the chart, click on legend (the upper left white rectangle) of the blue sub series. This will select the sub series and it will become yellow.

2 Open pop up menu, and choose **Add Indicator**.

3 In indicator setup, select Sub Series, with the following settings:

Filter set to `close <= open`

Pane set to **New Pane**

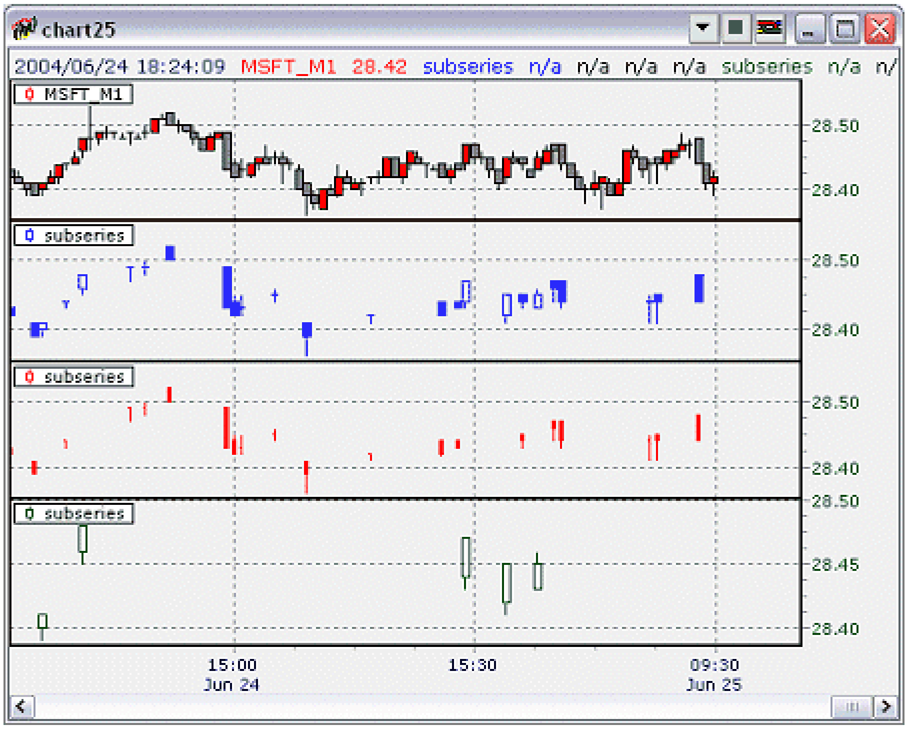
Style set to **Candle**

Color set to Red

Width set to 4

Press **Apply** button.

The chart should look like:



What Happened

You've isolated the up bars (green sub series) and down bars (red sub series) from the sub series. Notice that the blue series is exactly equal to the combination of the red and green series.

Comparing Successful and Failed Longs

Now let's count how many successful longs (up bars) and failed longs (down bars):

1 In the chart, click on legend (the upper left white rectangle) of the green sub series. This will select the sub series and it should become yellow.

2 Open pop up menu, and choose **Add Indicator**.

3 In indicator setup, choose **Valid Count**, with the following settings:

4 Under **Parameter** tab, **Period** set to a large number, e.g. 10000

5 Under **Visual** tab, **Color** set to Green.

6 Under **Vertical Scale** tab, check **Overlay**.

7 Press **Apply** button.

Similarly, count the failed longs:

1 In the chart, click on legend (the upper left white rectangle) of the red sub series. This will select the sub series and it should become yellow.

2 Open pop up menu, and choose **Add Indicator**.

3 In indicator setup, choose **Valid Count**, with the following settings:

4 Under **Parameter** tab, **Period** set to a large number, e.g. 10000

5 Under **Visual** tab, **Color** set to Red.

6 Under **Vertical Scale** tab, check **Overlay**.

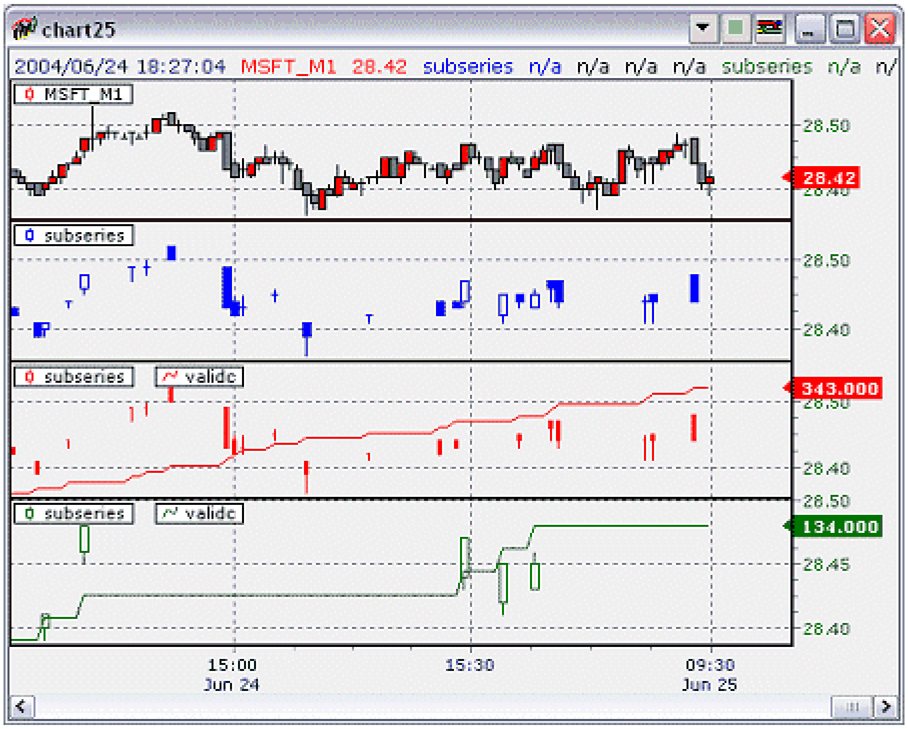
Press **Apply** button.

The chart will now have 2 indicators counting the occurrence of successful and failed longs. To make the counts easier to read:

1 Press ESC to un-select everything.

2 From the pop up menu, choose **Visual>Float Marker**

The chart should look like:



What Happened

The green line counts the occurrence of up bars, which are considered as successful longs. The red line counts the occurrence of the down bars, which are considered as failed longs.

In this chart, there are 134 successful longs but there are 343 failed longs. Also, if you observe the chart, the height of the red candles are in general taller than the height of the green candle. This means the amounts loss per trade is larger than than the amount win per trade.

These two conditions combined suggests that the long strategy may not be successful. In fact, the analysis suggests that a short strategy can be a winning strategy.

What you have done here is to quickly analyze a trading strategy without the need to write a trading system.

Tutorial: Variable Period Indicators

Variable period indicators use a second link to specify a variable length period.

Prerequisite

You know how to create a chart and add indicators to the chart. You need to know how to work with panes. Some familiarity with formula helps.

Goal

This tutorial shows you how to create a variable period moving, using indicator-on-indicator and formula.

Introduction

Variable period indicators allow the use of a second link to specify a variable length period. In general, these indicators have counterparts that have a fixed period parameter. Variable period indicators work like their fixed period counterparts, except the variable period indicators do not have period parameter. Instead, the second link is used to specify the period.

The variable length indicators are:

- Variable Highest High Bar
- Variable Linear Regression Slope
- Variable Linear Regression Value
- Variable Lowest Low Bar
- Variable Persist Condition
- Variable RSquare
- Variable Summation

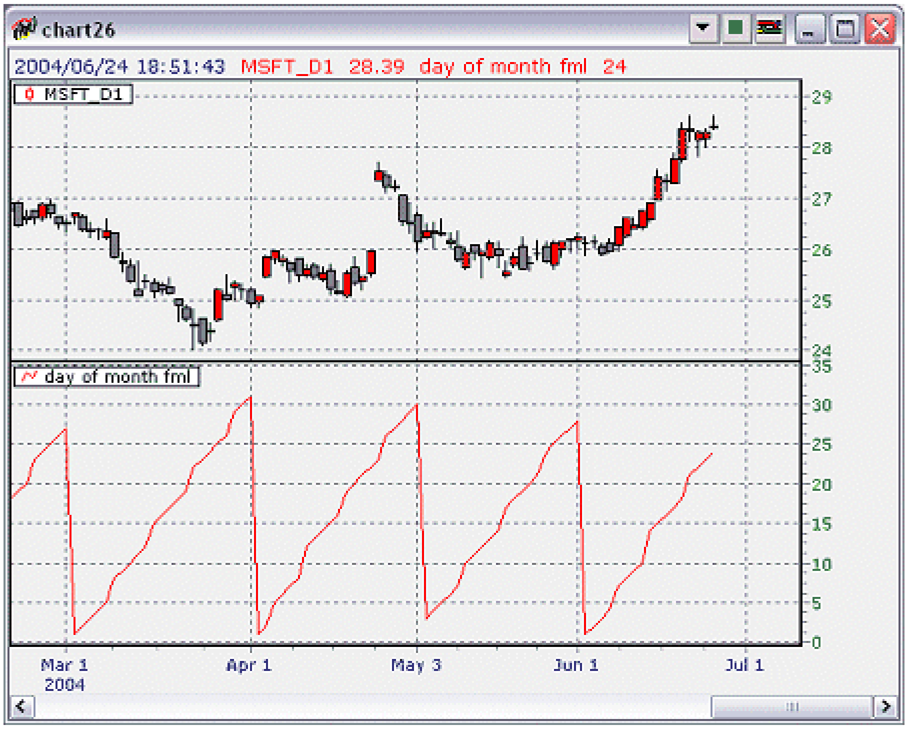
Examples

Example - Variable Period Moving Average

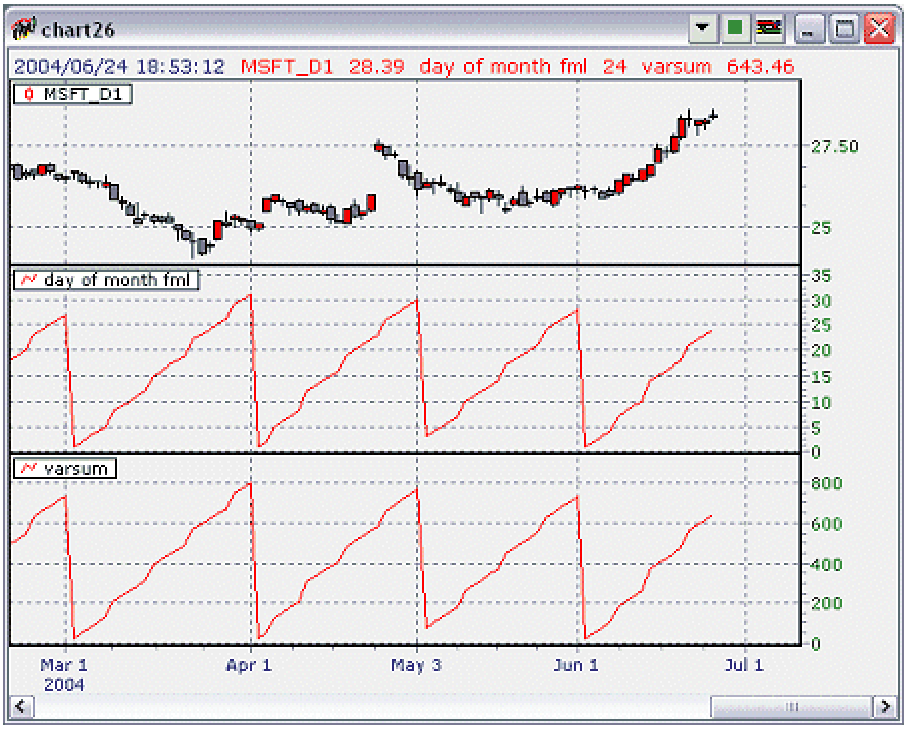
This example will setup a variable period moving average. The period of the moving average will increase near the end of the month. Thus the moving average becomes slower over time, and picks up speed again at the beginning of the month.

1 Create a MSFT daily chart.

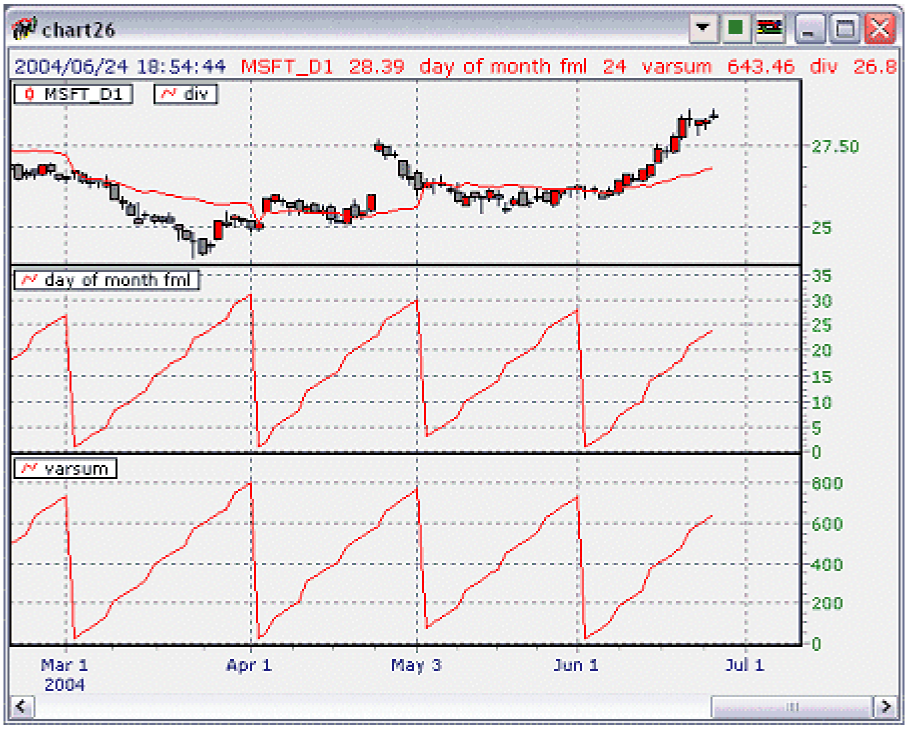
2 Apply a fml indicator to MSFT. Set **Label** to "day of month fml" and **Plot1** to `day(datetime)`. This will create a series that displays the day of month. We will use this series as the variable period.



3 Apply a Variable Summation indicator to the chart with Link1 set to MSFT and Link2 set to "day of month fml". This is the variable period summation for MSFT.



4 Finally, apply a division indicator, with Link 1 set to the variable sum and Link 2 set to "day of month fml". This is the variable period moving average. You can move it to the first pane and compare it with MSFT.



Example - Variable Period Moving Average Using Formula

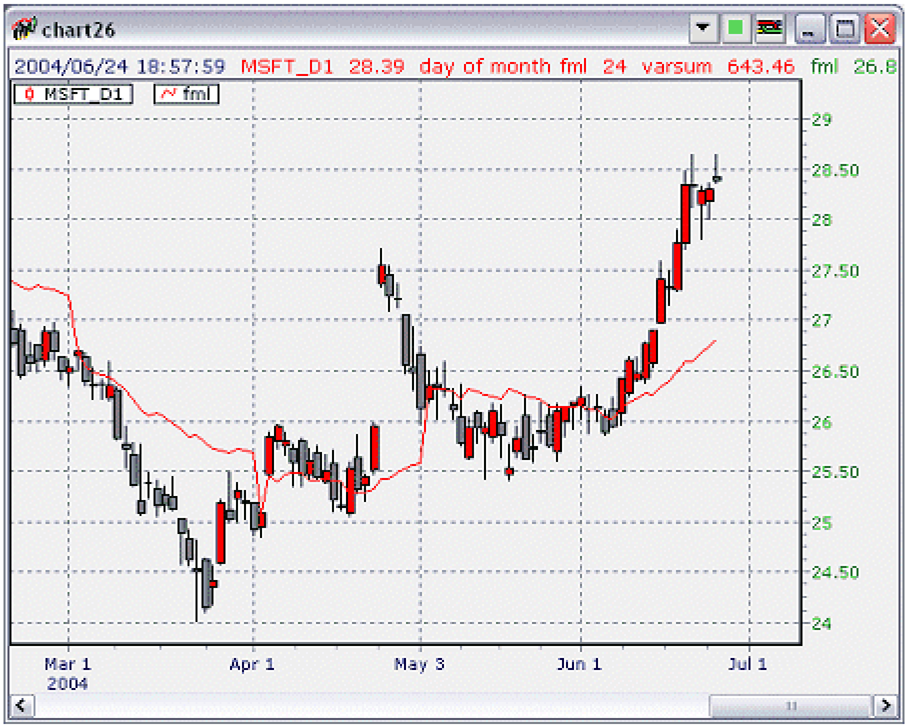
The second link in a variable period indicator is a series. Knowing this fact lets us create a variable period moving average using a single formula. The idea is to create a series in the formula using series assignment in formula, then use the series as the second link.

1 Create a MSFT daily chart.

2 Apply a fml indicator to MSFT, with **Plot 1** set to the following formula:

```
Period := day(datetime);
varsum(data1, Period) / Period
```

Here is the chart with the single formula version:



Tutorial: Installing Interactive Brokers Software

This tutorial is online. Go to ***Online Tutorial: Installing Interactive Brokers Software*** http://www.tickquest.com/NeoTicker/brokerIBhome.html.

Tutorial: Placing Orders to Your Broker

In this tutorial we will show you different methods to place orders in NeoTicker®. The orders will be send to your broker for execution.

Prerequisite

- You know basic NeoTicker® operations.
- NeoTicker® is connected to a real-time feed.
- Demo account with one of the supported brokers.

Goal

After you complete this tutorial, you will know how to place orders in NeoTicker®.

You should follow this tutorial during trading hours in orders for orders to be filled.

Connecting to Your Broker

First, you must configure NeoTicker® to connect to your broker. The steps are different for each broker. Follow the instructions in ***Connecting to Your Broker*** (on page 49).

After you complete the steps above,

- NeoTicker® should be connected to your broker through NT Order Server, and
- You should have a demo account from your broker ready. The demo account is crucial to follow this tutorial. Since you are learning, you should not use a real account.

If you haven't done so, start NeoTicker® now and login to your broker with demo account.

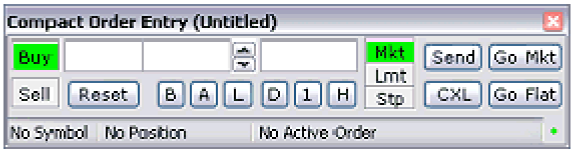
Placing Order with Compact Order Entry Form

When you set up your broker connection, you've probably tested it with Simple Order Entry form.

Here, we will show you how to use Compact Order Entry Form. Compact Order Entry Form is a bit more difficult to learn than Simple Order Entry form, but the payoff is well worth it. As the name suggests, Compact Order Entry Form is compact in size. It does not occupy a lot of screen space and you can easily place multiple forms to place order. This gives you tremendous trading advantages.

Opening Compact Order Entry Form

To open a new Compact Order Entry Form, in main window, choose **Order>New Order Form>Compact Order Entry**.



Placing Market Order

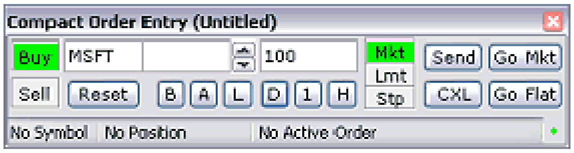
Let's place an market order.

1 Press **Buy**.

2 Enter MSFT in the left field.

3 Press **D** button.

4 Press **Mkt**.


Compact Order Entry (Untitled)

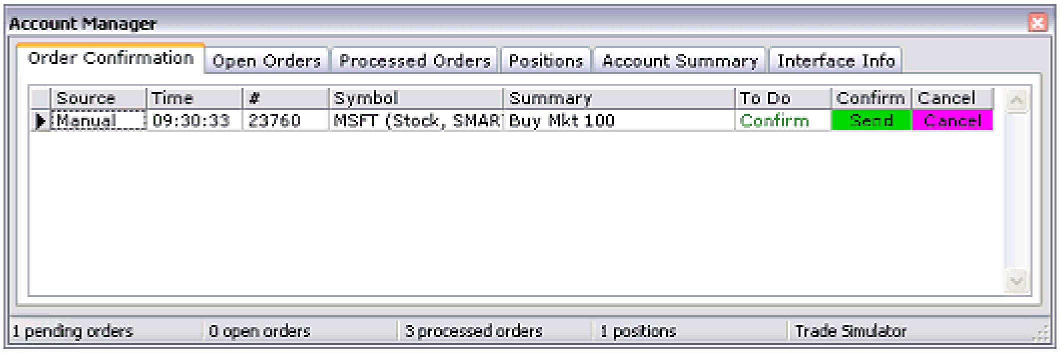
MSFT

100

You have just filled in market order for MSFT for 100 shares. When you pressed the **D** button, the default trade size for stock (100 shares) is filled in for you.

To transmit this order, press the **Send** button.

By default, all orders require confirmation before it will be sent to your broker. So Account Manager will open to let you confirm order:



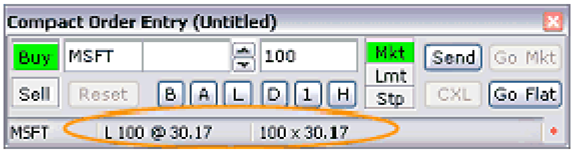
Source	Time	#	Symbol	Summary	To Do	Confirm	Cancel
Manual	09:30:33	23760	MSFT (Stock, SMAR	Buy Mkt 100	Confirm	Send	Cancel

1 pending orders | 0 open orders | 3 processed orders | 1 positions | Trade Simulator

In Order Manager, under the **Order Confirmation** tab, you can confirm the order to send your broker. But before you send the order, let's go through the columns in Account Manager to see what it tells you about the order.

- **Source** - it shows `Manual`. It means the order is sent by a manual order entry, i.e. from the Compact Order Form.
- **Time** - Time this order is submitted.
- **#** - Internal order number assigned to this order.
- **Symbol** - it shows `MSFT(Stock, SMART)`. This means the symbol is a stock, and the order will be sent to the SMART exchange.
- **Summary** - summary of the order, buy market 100 shares.
- **To Do** - action requires you to do.
- **Confirm** - action for confirmation
- **Cancel** - action for cancel.

Press **Send** under the **Confirm** column, the order will be sent to your broker. Since this is a market order, it will be filled almost right away. Order status is shown in bottom of the order form.



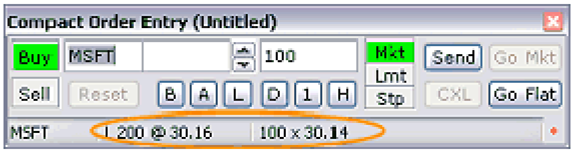
MSFT 100

MSFT | L 100 @ 30.17 | 100 x 30.17

The bottom show you three things:

- The symbol
- You are in a long position of 100 shares, at an average position cost.
- Number of shares filled and the fill price

Let's press the **Send** button again, this will buy another 100 shares of MSFT, adding to your position. Remember to confirm the order in Account Manager.



MSFT 100

MSFT L 200 @ 30.16 100 x 30.14

At the bottom you can see that you have a position of 200 shares.

Disabling Order Confirmation

Using Account Manager to confirm orders can be quite tedious once you become familiar with placing orders. You can disable order confirmation by:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

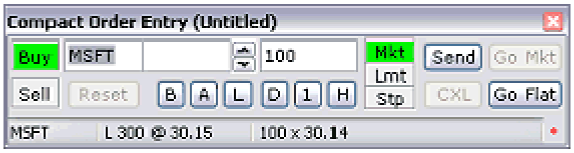
2 In Order Interface Setup,. press the **Confirmation** tab.

3 Under **Manual Order Confirmation**, **Order Form Entry**, choose **Skip Confirmation**. A warning dialog is open, press **Yes**.

4 Press **OK** button.

Let's try place another order. If you cannot find Compact Order Entry Form, choose **Order>Active Order Form** to help you find it.

In Compact Order Entry Form, press **Send** button. This will buy another 100 shares of MSFT, adding to your position to a total of 300 shares.



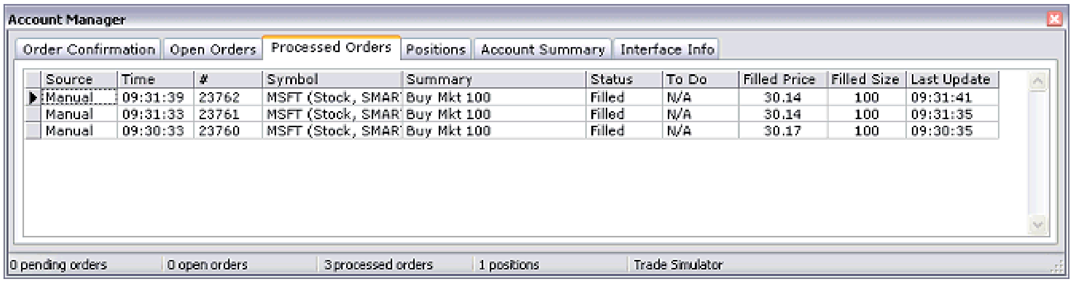
MSFT 100

MSFT L 300 @ 30.15 100 x 30.14

Reviewing Order with Account Manager

In addition to confirmation order, Account Manager also lets you review your orders and positions. In main window, choose **Order>Account Manager** to open Account Manager.

Press the **Processed Orders** tab. Under this tab, it will show you the orders that have been processed by Account Manager.

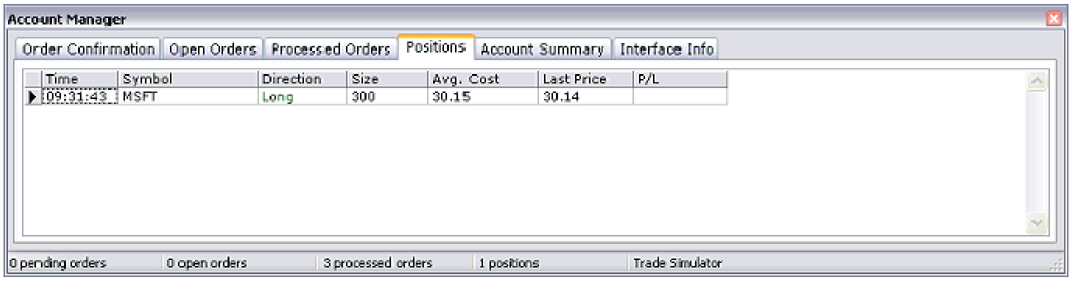


Source	Time	#	Symbol	Summary	Status	To Do	Filled Price	Filled Size	Last Update
Manual	09:31:39	23762	MSFT (Stock, SMAR	Buy Mkt 100	Filled	N/A	30.14	100	09:31:41
Manual	09:31:33	23761	MSFT (Stock, SMAR	Buy Mkt 100	Filled	N/A	30.14	100	09:31:35
Manual	09:30:33	23760	MSFT (Stock, SMAR	Buy Mkt 100	Filled	N/A	30.17	100	09:30:35

0 pending orders | 0 open orders | 3 processed orders | 1 positions | Trade Simulator

As you can see, the 3 buy orders are all shown here, with complete information about the orders shown.

Now press the **Positions** tab, it will show you your current positions.



Time	Symbol	Direction	Size	Avg. Cost	Last Price	P/L
09:31:43	MSFT	Long	300	30.15	30.14	

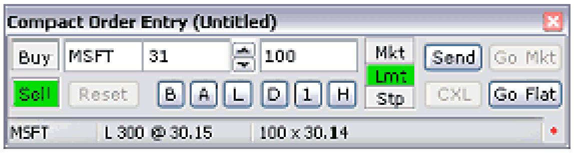
0 pending orders | 0 open orders | 3 processed orders | 1 positions | Trade Simulator

Placing Limit Order

Let's try place an limit order. If you cannot find Compact Order Entry Form, choose **Order>Active Order Form** to help you find it.

In Compact Order Entry form:

1. Press **Sell**.
2. In the middle field, enter a price much higher than the current price of MSFT, e.g. 31 in our case.
3. Keep the order size at 100.
4. Press **Lmt**.

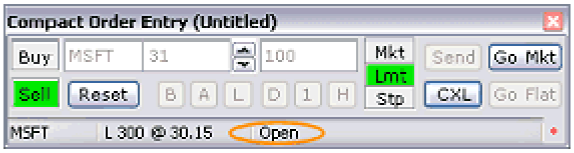


MSFT | 31 | 100

MSFT | L 300 @ 30.15 | 100 x 30.14

This is a limit order entry to sell MSFT at $31.00. Press **Send** button.

This is a price higher than what the market is willing to bear. So the order will not be filled and will stay open.


Compact Order Entry (Untitled)

MSFT 31 100

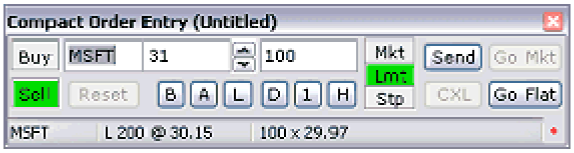
MSFT L 300 @ 30.15 Open

Notice that most controls are disabled in Compact Order Entry Form. This is because the form is tied to the order, and you cannot use the form until the order is filled/cancelled, or the form is disassociated from the order.

The only buttons available now are:

- **Reset** - Reset the form. This will disassociate the form from the order. The order will still be around, but you can now use the form to place other orders.
- **CXL** - Cancel the order.
- **Go Mkt** - Convert the order to market order.

Let's press the **Go Mkt** button. This will convert the limit order to market order. Because it is a market order, it will be filled quickly.


Compact Order Entry (Untitled)

MSFT 31 100

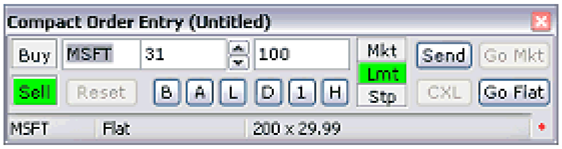
MSFT L 200 @ 30.15 100 x 29.97

Look at the bottom of the form. You now have only 200 shares longed (300 shares - 100 shares sold).

Go Flat

You can quickly cover your position by pressing the **Go Flat** button. Pressing the **Go Flat** button will issue buy/sell orders such that your position will become flat (0 shares).

Press the **Go Flat** button now. Because you current hold 200 shares, pressing the button will issue a market order to sell 200 shares.


Compact Order Entry (Untitled)

MSFT 31 100

MSFT Flat 200 x 29.99

Multiple Forms

You can set up multiple order forms for quick order entry.

1 In main window, choose **Order>Save Order Form**. A file dialog will open, enter a name of your choice and press **Save** button. Now your order form is saved and can re-open easily.

2 Choose **Order>New Order Form>Compact Order Entry** to open new order form.

3 Enter a symbol, e.g. INTC in the new order form.

4 Choose **Order>Save Order Form** to save the new order form.

5 Repeat steps 2 to 4 until you have several Compact Order Entry Forms, each for a different symbol.

Each of the the order form can let you enter and track one order. With several forms open, you can place orders for different symbols quickly.

The next step is to save the order forms together as a set. A set of order forms let you open them all at once.

1 In main window, choose Order>Save Set Of Order Forms As. A file dialog will open, enter a name of your choice as the name of the set and press **Save** button.

2 Choose **Order>Close All Order Forms**. This will close all order forms. You will be asked whether to save the individual forms. Since you have just saved the forms, you can choose **No**.

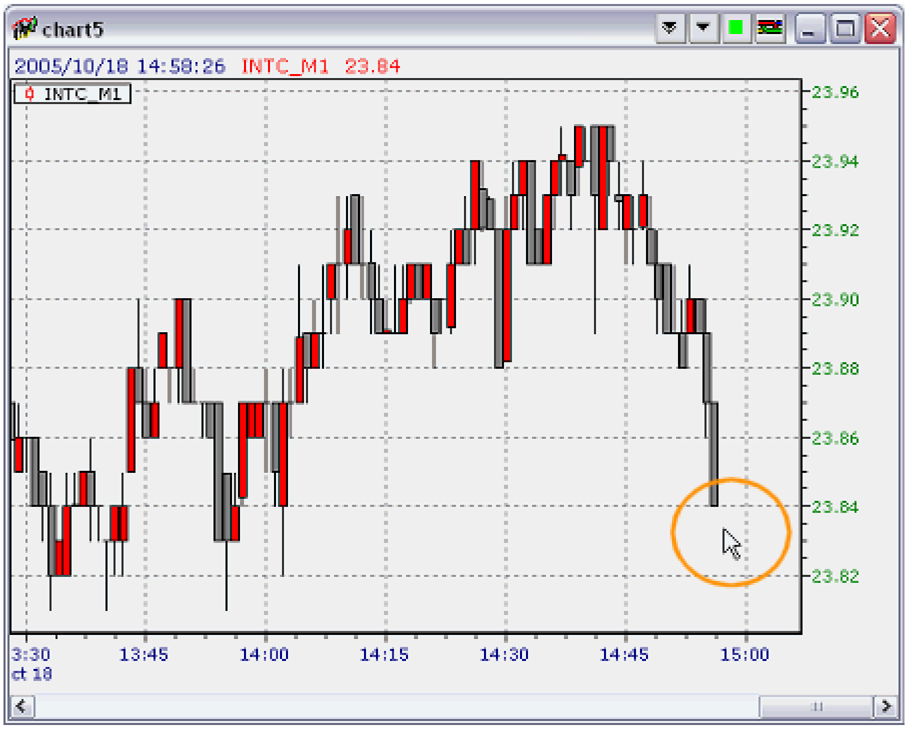
3 To re-open the set of order forms, choose **Order>Open Set Of Order Forms**.

Placing Order with Shortcut Keys

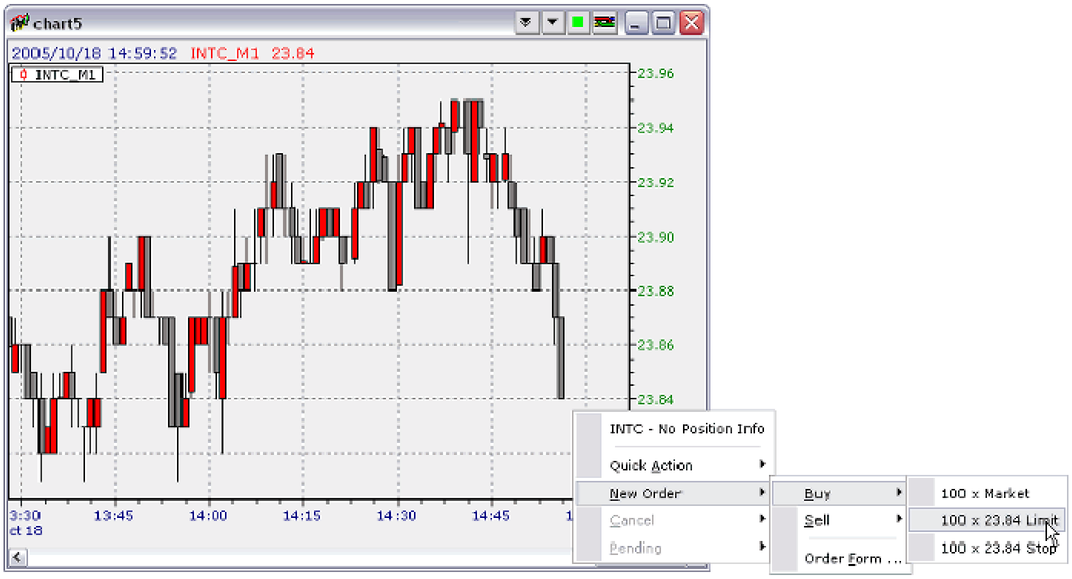
You can also use keyboard shortcut keys to place orders. F2 is used for buy, F3 is used for sell. Shortcut key relies on a function window (chart, quote window, etc) to learn the symbol and price you want to trade.

Placing a Buy Order

Open a 1-minute chart of INTC. Suppose you want to buy INTC at a price point. Move the mouse cursor to the price level, then press the F2 key.

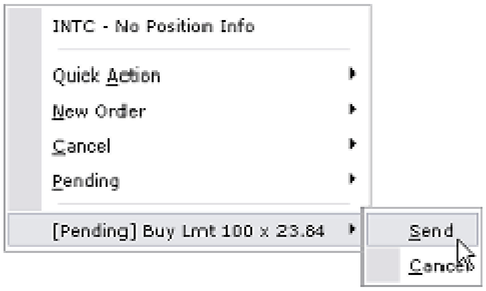


You will hear a "boing" sound and an order menu is opened. You can choose the type of order to place.

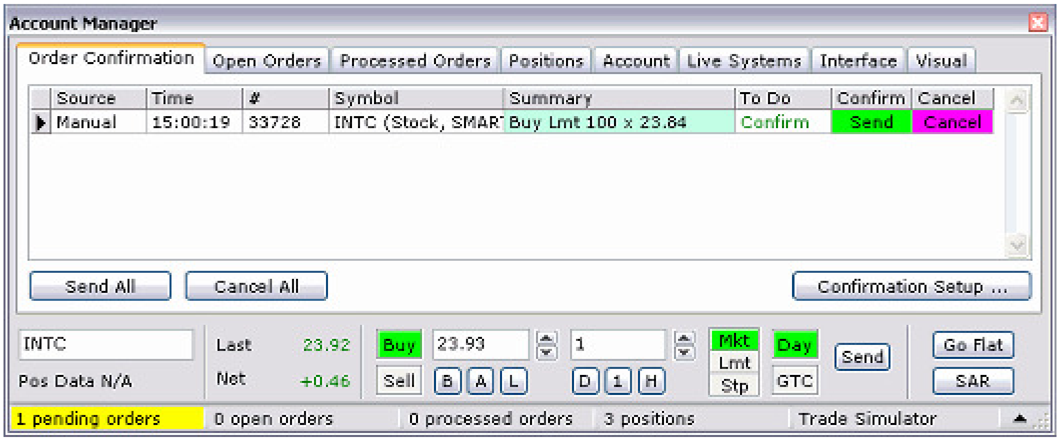


If pressing the F2 key does not open a menu, your F2 is set up to place order differently. This can happen if you install NeoTicker® before version 4.1 and has recently upgraded. Refer to ***Order Entry Shortcut Keys Setup*** (on page 739) for information on how to modify the F2 key behavior.

After you place the order, the menu will not close right away. Instead, a flashing menu entry is displayed to ask you to confirm the order.



In case you missed the menu, Account Manager is also opened to let you confirm the order. Notice that the source of the order is **Shortcut**. It means the order comes from pressing of F2/F3 key. To send the order, simply press **Send** under the **Confirm** column.


Account Manager

Source	Time	#	Symbol	Summary	To Do	Confirm	Cancel
Manual	15:00:19	33728	INTC (Stock, SMAR	Buy Lmt 100 x 23.84	Confirm	Send	Cancel

INTC
Last 23.92
Net +0.46
Pos Data N/A
23.93
1

1 pending orders | 0 open orders | 0 processed orders | 3 positions | Trade Simulator

Wait. Haven't you disabled order confirmation in the previous section? Yes, but what you have disabled is the order confirmation for orders coming from order form. You haven't disable confirmation for orders coming from shortcut keys (F2/F3).

The reason is simple. It is relatively safe to disable confirmation for order form entries. Because you type the symbol, price, size, etc., you know exactly what is in the order.

On the other hand, F2/F3 key use a lot of intelligence to fill in the order for you. It is possible F2/F3 may not guess what you want correctly. For example, the price in the figure above shows $23.84. This may not be the price you have in mind.

If you are comfortable with the default behavior of F2/F3, you can also disable shortcut key confirmation in Order Interface.

Setting Shortcut Options

You can use Order Interface Setup to setup order entry shortcut keys behavior:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

2 Press the **Shortcut** tab.

For more information on exactly how the options work, refer to ***Order Entry Shortcut Keys Setup*** (on page 739).

CHAPTER 10

Formula Topics and Tutorials

NeoTicker®'s formula is a powerful and versatile tool that can be used in many places in NeoTicker®. This chapter is a collection of tutorials to help you learn writing formulas.

Quote Windows

A formula column lets you perform calculations using quote fields and indicators. In addition, quote window alert conditions are specified in formula.

Time Charts

You can quickly plot custom expressions by using the `formula`, `formula2` and `formula3` indicators. For extra flexibility, you can write indicators and trading systems using formula. Time chart alert conditions are specified in formula.

Also, chart level bad tick filter is user configurable with formula.

Cluster Windows

Cluster window uses quote window formula for X, Y coordination specification.

Cluster alerts are specified in formula.

Dynamic Grid in Time and Sales Window and Time Chart

The dynamic grid in time and sales window and time chart can contain cells that evaluate quote window formula.

Pattern Scanner Users

Pattern scanner support time chart style formula. So any formula you develop for time chart can be reused in pattern scanner.

Power Indicators and Trading Systems

NeoTicker®'s set of power indicators and the Backtest EZ trading system use formula as the backbone. You can use formula (even with indicator inside the formula) as parameters to power indicators and Backtest EZ.

Backtest EZ allows you to construct a wide variety of trading systems without any programming.

Script and External Programs

You can evaluate formulas in scripts and in external programs through the `MakeIndicator` call.

In This Chapter

Understanding Formulas

Formulas are widely used in NeoTicker®, in areas such as calculations, indicator constructions, trading signals, tick filtering.

Formulas provide the following advantages:

- Very easy to specify. They are just simple mathematics formulas. NeoTicker® has taken the extra step to make formula easy to use for non-programmers.
- Very quick to evaluate. Formula is comparable in speed to internally defined language.
- Flexible. While maintaining the simple to use nature, formula is flexible enough for complex calculations.

Formula Examples

Formulas look and behave like the classic formulas in math books.

Examples:

Formula	Result
`4 + 5 * 3`	This formula evaluates to 19
`5 * 3 ^ 2`	This formula evaluates to 45
`30 > 80`	This formula evaluates to false

Formulas are much more useful when you use functions. Functions looks something like:

Formula	Result
`last`	Used in a formula column, the last function returns the last price of the symbol
`open`	Used in a formula column, the open function returns the open price of the symbol
`if(last > 70, 10, 5)`	Used in a formula column, the if function returns 10 if the last price is larger than 70. It returns 5 otherwise.

Functions that have parameters have the parameters enclosed in brackets. Functions that have no parameters do not need brackets. Here are some more examples of using functions in formulas:

Formula	Result
`last * 2`	Used in a quote window formula column, this formula multiplies the last price by 2.
`last + 2 * open`	Used in a quote window formula column, this formula returns the last price plus 2 times the open price
`last > 70`	Used as a quote window alert condition, this will trigger the alert when the last price is larger than 70

`Close(0) > TL(0)`	Used as a time chart alert condition, this will trigger the alert when the close price is larger than the trend line

Formula is not case sensitive. The following formula are considered identical:

```
last > 70

LAST > 70

Last > 70
```

Formula Operators

Arithmetic Operators

The basic operators are addition (+), subtraction (-), multiplication (*), and division (/). Multiplication and division has higher priority over addition and subtraction.

For example,

```
4 + 5 * 3
```

is evaluated to 19.

The power operator (^) has higher priority over the other arithmetic operators.

```
5 * 3 ^ 2
```

is evaluated to 45.

Comparison Operators

The following comparison operators are supported,

Operator	Meaning
>	greater than
>=	greater than or equal to
=	equal to
<=	less than or equal to
<	less than
<>	not equal to

For example,

```
4 + 5 * 13 < 5 * 3 ^ 2
```

is evaluated to 0 or false.

Comparison operators have lower priority than the arithmetic operators.

Boolean Operators

The following boolean operators are supported,

```
and
or
not
```

Two constant boolean values are predefined,

```
true
false
```

You can mix these two constants into any boolean expressions.

For example,

```
4 + 5 * 13 < 5 * 3 ^ 2 or true
```

is evaluated to true.

Boolean Operators have lower priority than the comparison operators.

When displayed, a boolean value is shown as 0 or 1. This is a display issue. Formula makes a clear distinction between boolean values and integer values. Thus you cannot mix boolean with integer with the boolean operators. For example, suppose the indicator mysignal returns 0 and 1 as signal, the following formula is not valid:

```
mysignal(data1) and data1.close > 30
```

Instead, you must make explicit conversion to translate the integer to a boolean value such that it can be used in a boolean operator. The following formula is valid:

```
mysignal(data1) > 0 and data1.close > 30
```

Functions

There are two type of functions, Generic functions and Context-Driven functions.

Generic functions are available in all formula definition. For example, the if function takes the following syntax,

```
if (boolean expression, answer1, answer2)
```

when `boolean expression` is evaluated to true, the `if` function will return the `answer1` value as its value; otherwise the value of `answer2` is used instead.

Context-Driven functions are available in specific situation and function window. For example, when you are specifying an alert for the quote window. All the Quote window fields are available as functions for your usage. They are not available in say the alert condition you've defined for a time chart.

For details on what context-driven functions are available in the specific usage you are looking for, refer to their own sections for more details.

Comments

A special type of statement is allowed in the formula language. It is the comments. You can freely put comments in between lines of statements. To start a comment line, type in the single quote (') character. For example, the first line in the following formula is a comment:

```
' my first try of writing an indicator
MyAvg := average ((High + Low) / 2, 10);
Plot1 := MyAvg;
Plot2 := average (MyAvg, 30); ' apply average to my average
again
```

If Condition and Choose Function

One of the most useful function in formula is the `if` function. The `if` function evaluates a condition. If the condition is true, the `if` function will return one expression. If the condition is false, the `if` function will return another expression.

For example, the formula

```
if (close > 30, high, low)
```

will return the `high` value if `close` is above 30. The formula will return the `low` value if `close` is below 30.

There is a also a `Choose` function that lets you choose from multiple conditions. Choose function takes a list of conditions and results as input, and returns result if the corresponding condition is true. For example,

```
Choose($a > $b, 1, $a < $b, -1, 0)
```

returns 1 if $a > $b, -1 if $a < $b, 0 if $a = $b.

A Note On Evaluation

With both `if` and `choose` functions, all parameters are evaluated, even though only one value is returned. For example, when formula encounters the following statement:

```
if (close > 30, myfunc1, myfunc2);
```

The following execution sequence happens:

1. `close > 30` is evaluated.
2. `myfunc1` is evaluated
3. `myfunc2` is evaluated
4. if the condition is true, the value from step 2 is returned. Otherwise, the value from step 3 is returned.

As a consequence, if `myfunc1` and `myfunc2` has side effects (e.g. printing out a debug message, placing order), the side effects will be carry out, regardless of the condition. In another words, `if` and `choose` functions do not control program flow.

In summary, you should only use `if` and `choose` functions as calculation tools. They are not designed to carry out actions that are condition dependent.

Local Variable Assignment

You can assign values to local variables. Using local variables can make your formula easier to read and more efficient.

Local variables are identified by the dollar sign in front of the variable name, e.g. `$a`, `$b`, `$myvariable`.

The following formula is an example:

```
$a := (H + L) / 2;
$b := C / $a * $a;
$b * $b
```

The above formula is equivalent to the following more complex formula:

```
C / (((H+ L) / 2) * ((H+L)/2)) * C / (((H+ L) / 2) * ((H+L)/2))
```

Local Variable Behavior in Tick-by-Tick Updating Charts

When a formula is used in a real-tick, tick-by-tick updating chart, local variables are not restored. For example, if you create the following formula with fml indicator (set to updated-by-tick):

```
$a := $a + 1
```

The local variable $a will increase by 1 when each tick arrives. Thus you can use this behavior for tick level analysis in formula in high time frame charts (15-sec, 1-min, etc). Basically local variables behave like pheap in script/IDL programming.

If you require restoring, you should use series variables (variables without the $ sign) instead.

Formula in Charts vs. Formula in Quote Windows

When using indicator in a formula, there are differences between the syntax used in quote windows and in charts. The differences are due to the data available to the formula.

In time chart, the input of the formula can come from different data and indicators. You must specify the data source to the formula, and the time frame of the formula is dictated by the data source. For example, if you specify the input to be 1-minute MSFT, then the formula will have a 1-minute time frame. So in the indicator, it requires a source data link. For example:

```
RSIndex(0,data1,10)
```

In quote window, the input of the formula must be the symbol in the row or an optional symbol. So you do not need to supply a data source. On the other hand, the time frame is unclear. Therefore, you must specify the time frame in a quote window formula. So in the indicator, it requires a time frame specification. For example,

```
RSIndex(0, M1, 10)
```

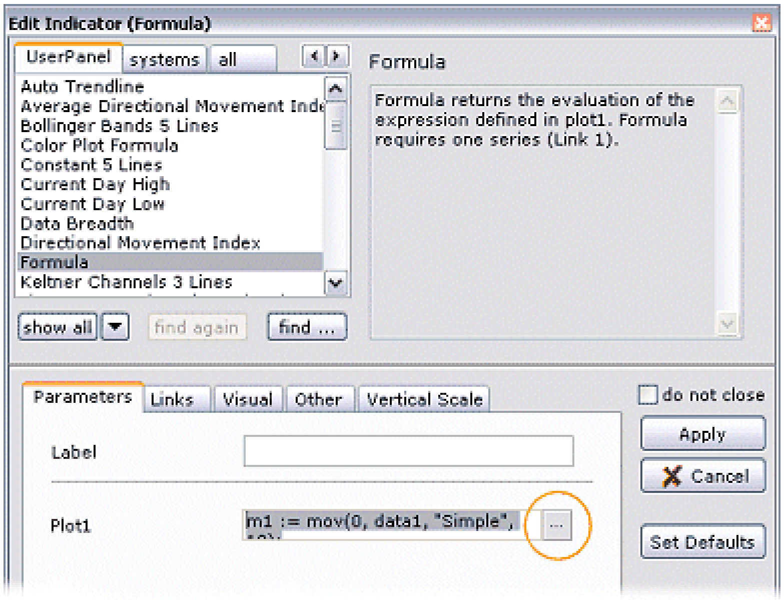
The good thing is, while you should understand the differences between the two, there is no need for you to memorize the detail syntax as Indicator Wizard will take care of that for you. See ***Tutorial: RSI in Quote Window Formula*** (on page 344) as an example.

Formula Design Time Debugging

Many formula problems can be pinpointed in design time (before the formula is put to use). You can use the **Verify** button in formula editor, or the verify feature of script editor to help you.

Formula Editor

For places you can enter a formula, a typically a ... button is shown. For example, you can enter a formula for plot1 in the fml indicator.

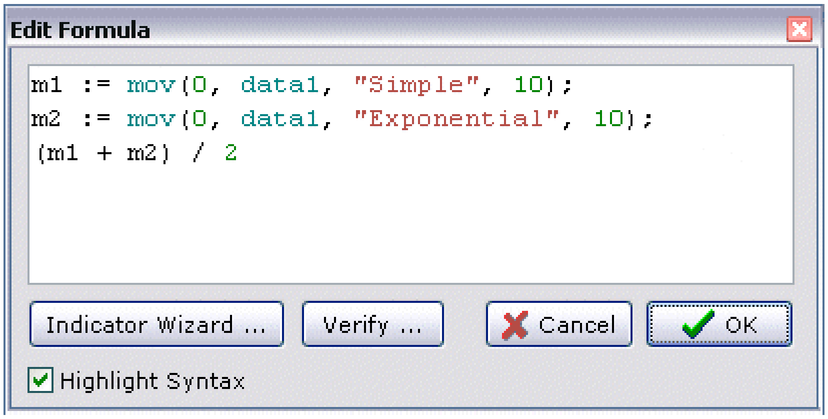


Formula returns the evaluation of the expression defined in plot1. Formula requires one series (Link 1).

Label

Plot1 m1 := mov(0, data1, "Simple", 10)

If you press the ... button, you will open formula editor. Formula editor allows you to enter formula with color coded key words, and tools to help you isolate problems.

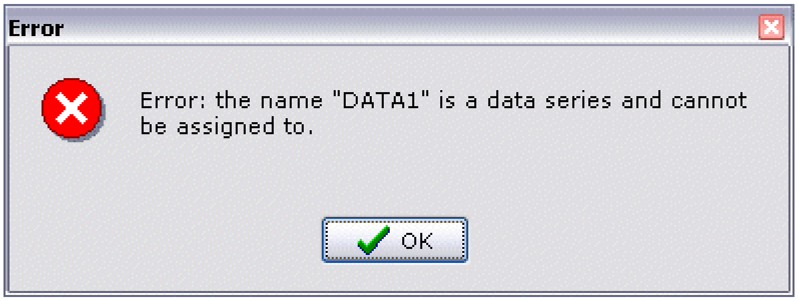


```
m1 := mov(0, data1, "Simple", 10);
m2 := mov(0, data1, "Exponential", 10);
 (m1 + m2) / 2
```

If **Highlight Syntax** is checked, then key words are highlighted. Functions, indicators, data sources are highlighted in teal. Literal strings is highlighted in dark red. Constants is highlighted in green.

If you press the **Verify** button, formula editor will verify the syntax and performs checking on the validity of the formula. For example, in fml, the following formula will produce an error when you press the **Verify** button:

```
data1 := mov(0,data1,"Simple",10);
```



Error: the name "DATA1" is a data series and cannot be assigned to.

Script Editor

If you are writing an indicator using formula, you will be using script editor. Formula in script editor is highlighted in identical ways as formula editor. To verify a formula in script editor, choose **Indicator>Verify**.

Formula Run Time Debugging

Even if a formula is perfect in syntax, the logic can be incorrect. You can use the generic function `Debug` to help you debug a formula in runtime. The syntax for `Debug` function is:

```
Debug(item1, item2, ...)
```

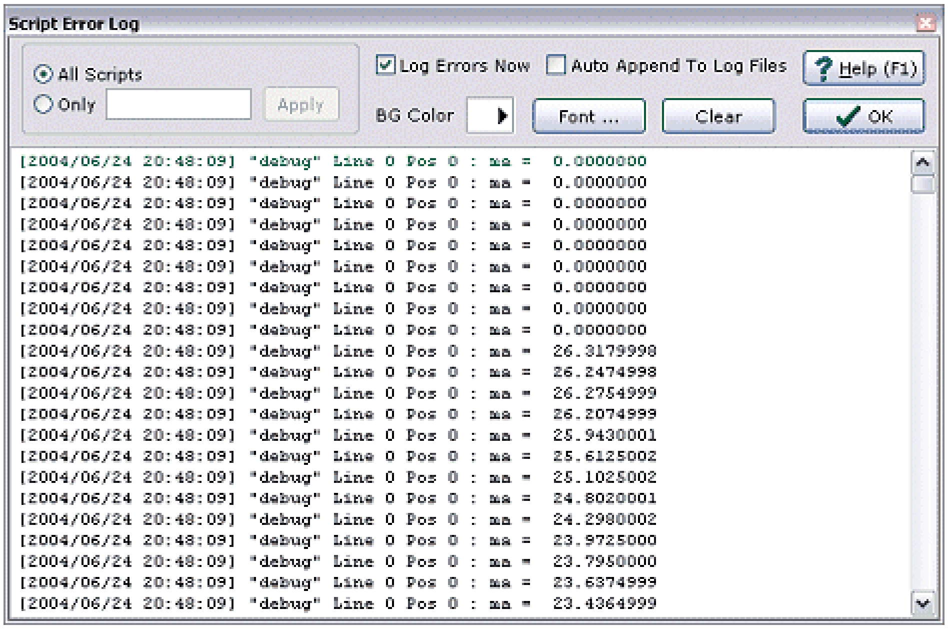
where `item1, item2`, etc are any double quote string or formula expression in a debug message. Debugging messages is shown in Script Error Log (**Program>Script Error Log**).

Example

1. Open script error log by choosing **Program>Script Error Log** in the main window.
2. In script error log, make sure **All Scripts** and **Log Errors Now** are selected.
3. Open a chart, add a data series to it. Add an `fml` indicator to the data series.
4. In the `fml` indicator, type in the following formula:

```
a := mov(0, data1, "Simple", 10);
Debug("ma = ", a);
a
```

When `fml` recalculates, the debug messages are shown in script error log.



● All Scripts
○ Only
☑ Log Errors Now
□ Auto Append To Log Files

```
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  26.3179998
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  26.2474998
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  26.2754999
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  26.2074999
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  25.9430001
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  25.6125002
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  25.1025002
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  24.8020001
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  24.2980002
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  23.9725000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  23.7950000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  23.6374999
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  23.4364999
```

Real Time Usage

If you indicator is set to update-by-tick, as each tick arrives, a formula that contains the `Debug` function will put out a message to script error log. This can easily create hundreds of message in a minute. Thus you should restrict the usage of `Debug` function only to strategic locations when you are debugging with real-time data.

Playing Sound in Formula

You can play sound in formula with the `PlaySound` function. `PlaySound` function has the following syntax:

```
PlaySound(Condition, SoundFileName)
```

`Condition` is a formula evaluated to 0 or 1. Sound is played only if `Condition` evaluates to 1.

`SoundFileName` is a double quote string that is the path to the sound file. If `SoundFileName` is a relative path, the sound file will be searched from NeoTicker®'s recognized Media folders.

Example

Apply the following formula in a chart with `fml` indicator. The formula will play a "boing" sound when close price is larger than 25. The sound is played only once per bar:

```
PlaySound(
    (C > 25) and
    islastbar > 0 and
    samebarupdate(data1) <= 0,
    "boing.wav");
```

The condition `islastbar > 0` returns true only if the data series is at the last bar. This condition will make sound not playing for historical bars.

The condition `samebarupdate(data1) <= 0` returns true only if the tick that updating the bar is the first tick of the bar. For subsequent ticks that update the bar, this condition is false. This condition will make sound not playing for every tick in the same bar.

Triggering Alerts in Formula

You can trigger alerts directly formula using the `Alert` function.

`Alert` function is a context function that lives within an indicator. Thus you can use `Alert` function in time chart and pattern scanner. To use `Alert` function in quote window, cluster or dynamic grid, you need to wrap the function up using an `fml` indicator or an indicator written using formula.

Alert function has the following syntax:

```
Alert(Condition, Priority, Message, fgColor, bgColor, Blink)
```

`Condition` is a formula. If condition evaluates to true, the alert is triggered.

`Priority` is a double quote string. This value is displayed in the **Priority** column in alert list.

`Message` is a double quote string. This value is shown when the alert is triggered.

`fgColor, bgColor` are integer values representing the alert's color scheme. You can use the pre-defined clRed, clBlue, etc value.

`Blink` is either true or false. It specifies whether the alert should be blinking when shown.

Example

1 Open a time chart.

2 Add data series to the chart.

3 Add `fml` indicator to the data series.

4 In the `fml` indicator, enter the following formula:

```
avg1 := average (data1, 10);
avg2 := average (data1, 20);

   alert (
      islastbar > 0
      and samebarupdate (data1) <= 0 ' enforce check on new tick
arrive
      and avg1 > 0 ' series must be ready
      and avg2 > 0
      and xabove (1, avg1, avg2) > 0,
      "High",
      "XAbove",
      clRed,
      clYellow,
      true);
```

This formula will trigger alert when the 10-period moving average cross above the 20-period moving average.

This example utilizes some tricks to make the alert not firing for historical data, nor for every tick in real-time update.

The condition `islastbar > 0` returns true only if the data series is at the last bar. This condition will make the alert not firing for historical data.

The condition `samebarupdate(data1) <= 0` returns true only if the tick that updating the bar is the first tick of the bar. For subsequent ticks that update the bar, this condition is false. This condition will make the alert not firing for every tick in a bar when the cross above happens.

Detecting Update-by-Tick Condition in Formula

Indicator can be updated by tick. In many scenario, you want to detect the update-by-tick status within the formula. For example, in alerts, you only want to trigger an alert once per bar for an update-by-tick indicator. You do not want the alert to be trigger for every tick in the bar.

You can use the samebarupdate indicator for this purpose. The syntax is:

```
samebarupdate(datasource)
```

When updated by tick, this indicator returns false when the computation is at the first tick of a bar. Subsequent ticks for the bar will return true.

When not updated by tick indicator, this indicator always returns false.

For usage examples, see ***Playing Sound in Formula*** (on page 267) and ***Triggering Alerts in Formula*** (on page 267).

`Samebarupdate` is an indicator, not a formula function.

Handling Date Time in Formula

Date Time Format

Formula uses a float number to represent a date time. The integral portion of the number represents the dates. Each calendar day has a value of 1. Time within a day is represented in the fraction portion of the number.

This means you can directly compare time. For example, if `time1` and `time2` are date time, and `time1` is earlier than `time2`, then the following expression is true:

```
time1 < time2
```

Encoding Date Time

To create a date time value, use the functions `makedatetime`, `makedate` or `maketime`.

For example,

```
makedatetime(2003, 2, 15, 11, 30, 15)
```

will create a date time value for 2003/2/15 11:30:15.

Comparing Date Time

You can directly compare date time by regular comparison operators. You can also use one of the build-in function to compare date time to make your formula more readable. The functions are: `intimerange, indaterange, indatetimerange`.

For example,

```
intimerange( mytime, maketime(11, 30, 00), maketime(11, 59, 59)
)
```

returns 1 if `mytime` is between 11:30:00 and 11:59:59, returns 0 otherwise.

Access to Specific Indicator Plot Series

For some indicators, they have multiple plots. By default, if you reference an indicator, plot1 of the indicator will be used. If you want to reference the second or other plot series, you can access them using the extended format,

```
IndicatorName.PlotN (Link1, … Param1, …)
```

The `N` in `PlotN` should be replaced with the proper numeric value for the plot series you wanted to access.

For example, you are using the indicator Bollinger Bands 3 Lines (short form bbands3) and would like to access the third plot series, then you can do the following,

```
bbands3.plot3 (data1, 100, 1)
```

or using the short form,

```
bbands3.p3 (data1, 100, 1)
```

Access to Previous Values of Indicator

If you need to directly access the value of an indicator from a few bars ago, you can simply put a bars ago parameter at the beginning of the parameter list.

For example, you are interested in calculating the difference between the current simple moving average value against the one from 2 bars ago. Then you can do the following,

```
Average (data1, 10) - average (2, data1, 10)
```

Access to Previous Values of Predefined Series Names

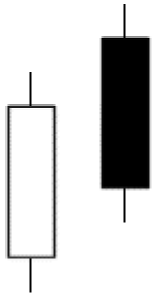
Say you want to check if a specific price pattern has happened, you will need to access the previous values of the pre-defined series open, high, low, close, etc. To reference the previous bar values of a pre-defined series, the general format is as follows,

```
SeriesName (BarsAgo)
```

If the BarsAgo is 0, which means you are trying to refer to the current bar information, you can drop the parameter listing totally, as follows,

```
SeriesName
```

For example, if you are trying to check for the candlestick price pattern “Dark Cloud Cover”, which looks like this



You can use the following expression to check for the pattern,

```
High (1) < Open and Low (1) < Low and Open (1) < Close (1)
and Open > Close and Close (1) > Close
```

Passing String Parameters in Nested Formula

If you call the fml indicator from within a formula, you will need to specify something like:

```
fml(data1, "XXX")
```

where `XXX` is the parameter passed to fml. Now `XXX` itself is a formula, and it can launch indicator calculation. This presents a problem if `XXX` calls an indicator that expects a string parameter, e.g. the moving average indicator.

You cannot simply specify the formula as:

```
fml(data1, "mov(0,data1,"Simple",10)")
```

It is because formula language will think the second quote (") ends the string parameter and creates a syntax error.

The solution is to use two quotes (""). When two consecutive quotes appear inside a string, formula will treat it as a single quote and as part of the string.

Therefore, the above formula can be correctly specified as:

```
fml(data1, "mov(0,data1,""Simple"",10)")
```

Formula in Time Chart, Pattern Scanner and Indicator Created with Formula

This section is a collection of topics about using formula in time chart, pattern scanner and indicator created with formula.

Statements

An indicator written in the formula language is composed of statements. You can have as few as one single statement to carry out all the calculation you need, or you can have multiple statements to construct complex calculations with multiple plots. Each statement is separated by the semicolon (;) character.

Following is an example of one line formula indicator.

```
Plot1 := close
```

Following is an example of multiple lines formula indicator.

```
MyAvg := average ((High + Low) / 2, 10);
Plot1 := MyAvg;
Plot2 := average (MyAvg, 30);
```

Semicolon is optional for the last formula within an indicator.

Each statement in the indicator is an assignment or expression.

Series Assignment

An series assignment takes the form of,

```
SeriesName := expression
```

`SeriesName` is a name that you have given to the expression on the right side of the assignment (:=) operator. For example, the following assignment statement assigns the closing price of the current bar to plot1 of your indicator.

```
Plot1 := c
```

Plot1 is known as a pre-defined series name which carries special meaning and usage. When you assign an expression to Plot1, you indicate that you want to make the first plot series of the indicator to use the value from that particular expression.

If the series name you have chosen to use is not a pre-defined one, then formula language will recognize this name in the statements that follow and you will be able to refer to this name to construct more complex formulas. Referring to our previous example,

```
MyAvg := average ((High + Low) / 2, 10);
Plot1 := MyAvg;
Plot2 := average (MyAvg, 30);
```

MyAvg is a user defined series name and it is reused in the assignment of both Plot1 and Plot2.

Assignment can help reduce the use of memory and CPU as the calculation of the assigned series is not duplicated.

Order of Assignment

Strictly speaking, assignment is a definition of a series. Formula language does not impose a strict order on the having the series declared before use. Therefore, the following formula:

```
Plot1 := MyAvg;
Plot2 := average (MyAvg, 30);
MyAvg := average ((High + Low) / 2, 10);
```

is a valid formula, performing the same calculation as the formula earlier in this section.

Series vs. Local Variable

You can assign values to series and local variables, but what is the difference?

- You can back reference values in a series, but not local variable, e.g. `A(1)` is valid, but `$A(1)` is not.
- You can use series as the input to indicator, e.g. `average(MyAvg, 30)` is valid, but `average($MyAvg, 30)` is not.
- Series is slower than local variables.
- Series uses more memory than local variables.
- Series can be used only in charts, pattern scanners and within formula indicator. Local variables can be used in all formula.

Thus if you are storing values for temporary access, use local variables. Otherwise, use series.

Series vs. MakeIndicator

Series variable is a single series. Conceptually it looks like a single plot. So if you create a series of a multiple plot indicator, (e.g. `MyBand := bbands3(data1, 10, 1)`), only the first plot is captured in the series. So in this example, you cannot apply `PlotValue` or `PlotCount` on `MyBand`.

When you create a series, it always inherit the time frame of the hosting indicator. This is usually sufficient unless you use compress time frame in the formula (see ***Compressing Series to Higher Time Frame*** (on page 283)). In this case, you need to put some thinking into the time frame of an indicator, and the time frame of the hosting indicator may not be what you want.

Both of these issues can be overcome by using the MakeIndicator function (see ***MakeIndicator in Formula*** (on page 287)) instead of a straight series assignment.

Handling Invalid Bars

NeoTicker® makes the distinction between valid bars (bars with data) and invalid bars (bars with no data, but still occupies a time slot).

When a formula is evaluated against an invalid bar, the result of the particular bar becomes an invalid.

Reference to Indicator

Consider the formula:

```
MyAvg := average ((High + Low) / 2, 10);
Plot1 := MyAvg;
Plot2 := average (MyAvg, 30);
```

`Average` is an internal indicator. It is the Simple Moving Average indicator.

In general the format of using an indicator is as follows,

```
IndicatorName (Link1, Link2, … Parameter1, Parameter2, …)
```

Going back to our example, Simple Moving Average indicator takes a single link and one parameter **Period**. Thus the expression:

```
Average ((High + Low) / 2, 10)
```

means you are trying to take a 10 period simple moving average on the expression

```
(High + Low) / 2
```

For indicators that require multiple links, you will list all the link items first before listing the parameters. For example, the Subtraction indicator (`diff`) takes two links and no parameters, the format of using this indicator will look like the following,

```
Diff (data1, data2)
```

Indicator Plots

Assigning Values to Different Plots

Use the pre-defined series `PlotN` to assign values to specific plots.

For example,

```
Plot1 := 1;
Plot2 := 3;
Plot3 := 5
```

will assign the constants 1, 3, 5 to a three-plot indicator.

Making a Bar Invalid

NeoTicker® makes the distinction between valid bars (bars with data) and invalid bars (bars with no data, but still occupies a time slot). To mark a bar in an indicator plot invalid, use the pre-defined series `SuccessN`.

For example, if you are writing an indicator that highlights specific price pattern, you will not want the indicator to plot on every bar. So you will use `SuccessN` to control which bar you want to draw your marker.

Lets work on a simple example. You are interested in highlighting bars that has a higher low than 4 bars ago. You would like to know if it could highlight the current trend for you. So you create an indicator using formula with 2 plots and the Meta plot style High Low which enable you to connect plot 1 and plot 2 vertically on a chart.

You can type in the following to complete the task.

```
Plot1 := high;
Plot2 := Low;
MyTrend := low > low (4);
Success1 := MyTrend;
Success2 := MyTrend;
```

Visual Breaks

You can set visual break in an indicator plot. For example:

```
VisualBreak1 := true
```

will create a break in plot 1.

```
VisualBreak2 := true
```

will create a break in plot 2.

Visual break is applicable to time charts only. The indicator plot must be set to either line or square style, with the break property set to **Visual**. See ***Broken Lines*** (on page 1123) for information on how to set up in a chart.

Accessing Data Links and Fields

When you want to refer to the generic value of the linked series in your indicator, you will use Data1, Data2, etc. to represent the values from the first link, second link, and so forth.

If you need to access specific field of the data series, you can access them using the following format,

```
DataN.FieldName
```

If DataN is omitted, Data1 is assumed, e.g. C is equivalent to Data1.C.

Normal Fields

FieldName is one of the following:

Long Form	Short Form	Notes
Open	O	
High	H	
Low	L	
Close	C	
Volume	V	
Tick	T	
OpenInt	OI	
Date	No short form	
Time	No short form	
Barsnum	No short form	Bar number of the linked series
DateTime	DT	

See section below for additional fields.

The other situation that you will use DataN series is that an indicator that you are going to use requires access to all the information that the series contain, like high, low, close, and volume values instead of the designated field. Then DataN can be used to ensure the indicator can access all the fields.

For example, the indicator Avgrange requires a high value and a low value to calculate average range. If the data linked into Avgrange is not a data series, then the result is always zero as the range of each bar of the linked series will always be zero, as the high value and low value of the linked series is always the same. Following is one such example.

```
Avgrange (c, 10)
```

that always return 0. If dataN is used instead, then the range can be calculated property:

```
Avgrange (data1, 10)
```

If `dataN` is not specified and only the `FieldName` is used, then it is assumed you are trying to refer to the fields of data1.

Latest Bid/Ask/Trade Fields

You can use fields listed here to access latest bid/ask/last trade information, regardless of the underlying series' time frame.

To prevent accidental future peeking, bid/ask/trade fields are restricted. You can only access them in real-time or when the user is doing a tick replay (see ***Replay by Tick*** (on page 1123).). They are not available when recalculating the indicator using historical data. You can use the corresponding 'has' function to determine the availability of data.

Field	Meaning
`AskPrice`	Returns the latest ask price. Use `HasAsk` to determine if this property is available.
`AskSize`	Returns the latest ask size. Use `HasAsk` to determine if this property is available.
`BidPrice`	Returns the latest bid price. Use `HasBid` to determine if this property is available.
`BidSize`	Returns the latest bid size. Use `HasBid` to determine if this property is available.
`HasAsk`	Returns true if latest ask information is available.
`HasBid`	Returns true if latest bid information is available.
`HasLastTrade`	Returns true if last trade information is available.
`LastTradeDateTime`	Returns last trade date/time. Use `HasLastTrade` to determine if this property is available.
`LastTradePrice`	Returns last trade price. Use `HasLastTrade` to determine if this property is available.
`LastTradeSize`	Returns last trade size. Use `HasLastTrade` to determine if this property is available.
`SideAsk`	This property complements `UpAsk`. Query this property to see if the latest ask is actually a side ask. Use `HasAsk` to determine this property is available.
`SideBid`	This property complements `UpBid`. Query this property to see if the latest bid is actually a side bid. Use `HasBid` to determine this property is available.

`UpAsk`	Returns true when the latest ask is an up ask. Use `HasAsk` to determine this property is available. For side ask, it retains the previous up ask value. You can use `SideAsk` to determine if the ask is actually a side ask.
`UpBid`	Returns true when the latest bid is an up bid. Use `HasBid` to determine this property is available. For side bid, it retains the previous up bid value. You can use `SideBid` to determine if the bid is actually a side bid.

Tick Statistics Fields

Fields listed here are tick level statistics that are collected on a tick-by-tick basis. In another words, unless the chart has been running in real-time for a long time, you will need to perform a tick replay to get an accurate value. For more information on tick replay, see ***Replay by Tick*** (on page 1123).

These fields have long forms only. There are no short forms (e.g. Data1.DayTick16). All the fields with names started with the prefix "Day" are reset on every trading day.

Field	Meaning
`DayCancelledBidVolume`	This field accumulates the cancelled bid size at the highest bid.
`DayCancelledAskVolume`	This field accumulates the cancelled ask size at the lowest ask.
`DayVWAP`	Volume Weighted Average Price.
`DayTWAP`	Tick Weighted Average Price.
`DayTotalTicks`	Total number of ticks.
`DayTotalUpTicks`	Total number of up ticks.
`DayTotalDownTicks`	Total number of down ticks.
`DayTotalUpSideTicks`	Total number of side ticks (up side ticks only).
`DayTotalDownSideTicks`	Total number of side ticks (down side ticks only).
`DayTotalBidTrades`	Total number of trades at bid price.

`DayTotalAskTrades`	Total number of trades at ask price.
`DayTotalVolume`	Total volume.
`DayTotalUpVolume`	Total volume from up ticks.
`DayTotalDownVolume`	Total volume from down ticks.
`DayTotalUpSideVolume`	Total volume from up side ticks.
`DayTotalDownSideVolume`	Total volume from down side ticks.
`DayTotalBidTradeVolume`	Total volume on trades at bid price.
`DayTotalAskTradeVolume`	Total volume on trades at ask price.
`DayTick16`	Statistics on the last 16 ticks. Each up tick counts as 1. Each down tick counts as -1. 0 for down tick.
`DayBATick16`	Statistics on the last 16 ticks. Each trade at ask counts as 1. Each trade at bid counts as -1. 0 otherwise.

Symbol

`Symbol` field returns the symbol of the linked series. It is useful for reporting purpose. For example,

```
   report("", true, Data1.symbol)
```

will display the symbol of Data1 in the default report window.

Accessing User Specified Parameters

User specified parameters can be accessed using the `ParamN` series. First parameter is called `param1`. Second parameter is `param2`, and so forth.

For example, you are trying to customize the built-in moving average indicator by using the average of high, low and close instead of just a linked value.

You create a new indicator with 2 parameters to emulate what the built-in moving average indicator will take. Thus the first parameter is a string with default value “sle” and the second parameter takes integer.gt.1 with default value of 20.

Your indicator will look like the following,

```
   Plot1 := Mov ((h + l + c) / 3, param1, param2)
```

NeoTicker® automatically recognizes that the indicator Mov requires its first parameter to be string and will take `param1` as a string instead of evaluating its value.

ParamIs Function

The `ParamN` series always returns a numerical value. But what if the user parameter is set as a string? You can use the `ParamIs` function to compare a user parameter to a string.

The following example compares the first user parameter, i.e. parameter 1, with the string "Go Long". So if parameter 1 is the string "Go Long", 1 is returned, otherwise 0 is returned.

```
ParamIs(1, "go long")
```

Note that the `ParamIs` comparison is non-case sensitive, e.g. "Go Long" will match "go long".

Compressing Series to Higher Time Frame

For real-time version of NeoTicker®, you can compress all intraday and EOD time frames to higher time frames.

For NeoTicker® EOD, you can compress all EOD time frames to higher time frames.

Compress series is a technique to create a 'virtual' series that is of a higher time than the source series.

For example, in a chart, you can create a compress 5-minute data series from a 1-minute data series. The compress 5-minute series will return values at 5-minute interval like a normal 5-minute series. However, the compress series will return the intermediate value of constructing the 5-minute interval at each 1-minute point.

In this case, there are two advantages of using compress series:

1. You do not need to manually add a 5-minute data series to the chart in order to apply any analysis that is based on a 5-minute time frame.
2. The 1-minute point allows you to sample the value of 5-minute series before the construction is complete. This sampling technique can be used to create trading system that analyze data using 5-minute time frame, but firing orders using 1-minute time frame.

You can compress to a higher time frame as long as the higher time frame is a true multiple of the source series. This means you can compress from 1-minute to 5-minute, 1-minute to 1-hour, but not from 2-minute to 5-minute.

In some cases, success of the compression will depend on the starting point of trading hour. For example, 2-minute to 1-hour compression will not be successful if the trading hour is set to an odd minute.

The following table lists several examples:

Source Series	Compress Series	Validity
1-minute	5-minute	Valid - true multiple
2-minute	5-minute	Invalid - not a true multiple
2-minute	10-minute	Valid - true multiple
1-minute	1-hour	Valid - true multiple
1-minute	1-day	Valid - true multiple
2-minute	1-hour	Validity depends on trading start time
2-minute	1-day	Validity depends on trading hours

Syntax

The following is the syntax for compress series:

compressseries(name, source, period, size)

The parameter `name` is the name you assigned to the series.

The parameter `source` is data source of the series. The source can be any of the linked series (e.g. `data1`, `date2`, etc), indicators series you have created in the formula, expression (e.g. `data1 + 5`) or another compress series you have created in the formula.

The parameter `period` is the period of the compress series to be created. It can be one of:

- `ppTick`
- `ppSec`
- `ppMin`
- `ppHour`
- `ppDaily`
- `ppWeekly`
- `ppMonthly`

- `ppQuarterly`
- `ppYearly`

The parameter `size` is an integer that specifies the size of compress series.

Example

The following formula will compress a 5-minute series, assuming data1 is linked to a 1-minute series.

```
compressseries(myseries, data1, ppMin, 5);
```

This formula will compress a 5-minute series called `myseries`.

You can access myseries as if it is any regular series, even apply indicator calculation on myseries. For example:

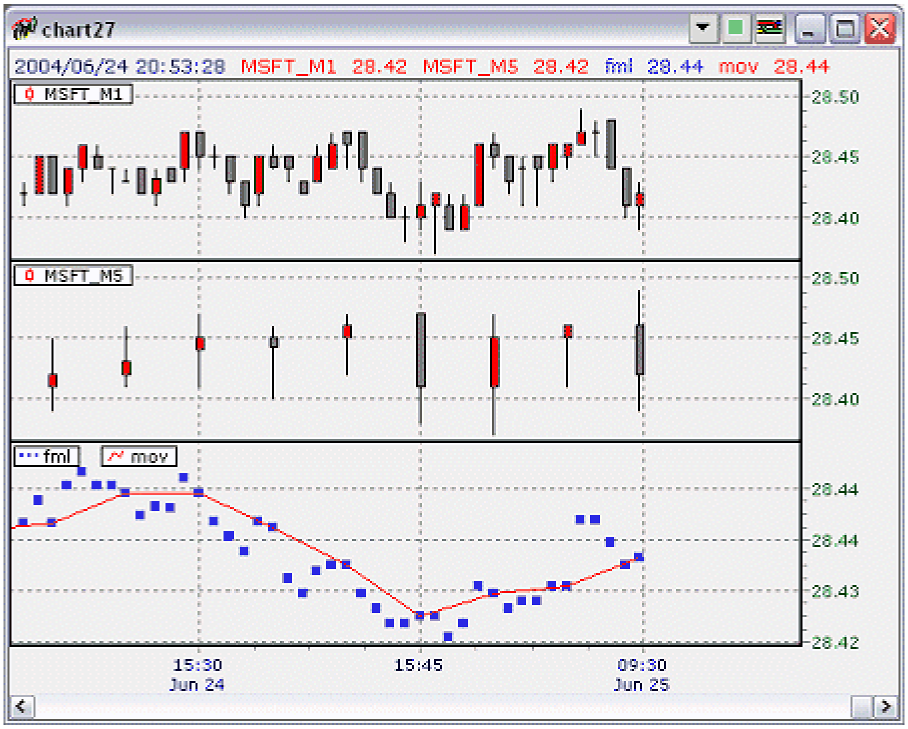
```
mov(0, myseries, "Simple", 10);
```

This formula will apply a 10-period simple moving average on `myseries`.

You can type in this example using the `fml` indicator. The complete code is as follows:

```
compressseries(myseries, data1, ppMin, 5);
mov(0, myseries, "Simple", 10);
```

The following figure shows the above formula in action:



In the bottom pane, the blue dots are from the formula above. The red line is a 10-period simple moving average applied on 5-minute data. As you can see, the compress series (blue dots) matches the indicator on 5-minute data (red line) exact at each 5-minute point. In addition, the blue dots provide sampled values at each 1-minute point.

Fields

Compress series have access to all fields of a regular series. For example, `myseries.high` returns the high value of the compressed series. The table below lists the valid fields:

Long Form	Short Form	Notes
`Open`	`O`	
`High`	`H`	
`Low`	`L`	
`Close`	`C`	
`Volume`	`V`	
`Tick`	`T`	
`OpenInt`	`OI`	
`Date`	No short form	
`Time`	No short form	
`Barsnum`	No short form	Bar number of the linked series
`DateTime`	`DT`	

MakeIndicator

Note that when you apply a moving average to a series of compressed time frame, you should not assign the indicator into a series variable (see ***Series Assignment*** (on page 273)). Because a series always inherit the time frame of the hosting indicator, if you do a series assignment, the series will be using the time frame of the hosting indicator, not the compressed time frame. In most cases, this is not what you want. Instead, if you want a storage for an indicator using compressed time frame, you should use the MakeIndicator function (see ***MakeIndicator in Formula*** (on page 287)).

Accessing the Current Bar Number

When a formula is executed, it is working on the current bar. You can determine the bar number by using the `CurrentBar` function.

`CurrentBar(N)` returns a sequential index starting from 1. When the script is first called to do calculation, usually that is the first bar right after the indicator's Min Bars setting, the current bar is set to 1.

For example, CurrentBar(0) returns the bar number of the current bar. CurrentBar(1) returns the bar number of 1-bar ago.

MakeIndicator in Formula

When to Use MakeIndicator

Use MakeIndicator if you want to create a temporary storage for an indicator that requires:

- Multiple plot capability, or
- Having a time frame different from hosting indicator

Otherwise, if you just want to create an indicator in formula, it is simpler to use ***Series Assignment*** (on page 273).

Syntax

The syntax for MakeIndicator is:

```
MakeIndicator(Indicator Instance Name, Indicator Name, Link 1
{, Link 2}* {, Param 1}*)
```

`Indicator Instance Name` is the name you give to the indicator you are going to make with `MakeIndicator`.

`Indicator Name` is the short name of the indicator to be used for making the indicator.

`Link 1`, `Link 2`, etc are expressions that serve as data input to make the indicator.

`Param 1`, `Param 2`, etc are the parameter for the indicator.

Example: Expression as Link

Just like series assignment, the link can be an expression. The following formula is an example that can be used directly in an fml indicator in a chart.

```
MakeIndicator(mymov, mov, data1 + 10, "simple", 10);
mymov
```

Example: Multiple Plot

The following formula illustrates the multiple plot capability of MakeIndicator. The formula can be used directly in an fml indicator in a chart.

```
' Creates a 3-band Bollinger Band Indicator
MakeIndicator(mybband, bbands3, data1, 10, 1);
```

```
' Access values of different bands
PlotValue(0, mybband, 1) - PlotValue(0, mybband, 2)
```

Example: Multiple Time Frame

The following formula illustrates the multiple time frame capability of MakeIndicator. The formula can be used directly in an fml indicator in a chart. Note that data1 must be 1-minute time frame.

```
' Create a 5-min series
compressseries(myseries, data1, ppMin, 5);

' MakeIndicator preserves 5-min time frame
MakeIndicator(mymov1, mov, myseries, "simple", 10);
' Series assignment will convert back to 1-minute.
mymov2 := mov(myseries, "simple", 10);

' Apply indicator on mymov1. The moving average is based
' on the 5-min data of mymov1
mov5min := mov(mymov1, "simple", 10);

' mov1min is based on mymov2, which is 1-minute time frame
mov1min := mov(mymov2, "simple", 10);

' So although mymov1 is exactly the same as mymov2 in values,
' mov5min and mov1min are different because of the time frame
' differences
mov5min - mov1min
```

Using Random Stream

To generate random number:

1. Decide a random stream to use. There are 100 random streams available, with stream id from 0 to 99. If you want to reproduce a particular stream of random number, you need to choose a unique stream for the formula.
2. Decide whether you want to use `randomStreamRandomize` or `randomStreamSeed` to initialize the random stream.
3. In the random stream initialization function's condition, use the `firstcall` function to determine if it is the first time the indicator is called. `Firstcall` works only for indicator formula.
4. Use `randomStreamDouble` or `randomStreamInteger` to generate random number.

See ***Generic Functions - Random Numbers*** (on page 302) for random function reference.

See ***Context-Driven Functions for Time Chart, Pattern Scanners and Inside Formula Indicators*** (on page 308) for `firstcall` reference.

Formula in Quote Window Column, Dynamic Grid, Cluster Coordinates

Column Names / Quote Fields

To use a column name within a formula, simply use the field name in the formula. For example, you want to calculate today's traded range, and then you can use the following formula to do the calculation:

```
High - Low
```

`High` and `Low` are column names that you can use freely within custom formula columns.

You can refer to these column names as many times as you want. You can also construct complex formulas by referring to calculated results from another custom formula column. Each custom formula column must be named with a unique name. You can use this name exactly like any other column names.

Special Character Conversion

When referring to a column name that has the percentage (%) sign as the suffix, replace "%" with "Pct".

For example, the field Range% can be used in the formula language using the name `RangePct`.

When a field name starts with a number, add an extra f character at the beginning.

For example, the field 52WeekHigh can be used in the formula language using the name `f52WeekHigh`.

Data Series

Data series refer to time series that are dynamically defined when you specify them in a formula. Direct access to data series values enable you to check for specific price formation without using a chart. You can also use these data series values for comparison purpose against indicator values.

For example, `close (0, M5)` is a data series function referring to a 5-minute bar based data series and you want to obtain its latest value. To refer to the previous closing price of the same 5-minute bar data series, you can use `close (1, M5).`

The general format for accessing data series is,

```
SeriesName (BarsAgo, TimeFrame)
```

`SeriesName` is one of the following:

Long form	Short form
`Open`	`O`
`High`	`H`
`Low`	`L`
`Close`	`C`
`Volume`	`V`
`Tick`	`T`
`OpenInterest`	`OI`
`DateTime`	`DT`

`BarsAgo` is a value evaluated to an integer greater than or equal to zero. `BarsAgo` refers to the number of bars ago you want to access. For example, `barsago` equals 0 means you are referencing the latest bar, `barsago` equals 1 means you are referencing one bar ago.

`TimeFrame` is an identifier that specifies the time frame type and time frame period. For example, M5 represents a 5-minute series. D2 represents a 2-day series. See ***Time Frame Specification*** (on page 293) for a complete description of time frame.

Now lets look at a few examples using the data series functions.

Say you are trying to identify the current price that has exceeded the previous trading day high, the formula will look like this,

```
Last > H (1, D)
```

Now say you want to ensure that yesterday’s price bar is a down close. Then you need to refer to both the previous trading day and the day before that. The formula will look like this.

```
C(2, D) > C(1, D) and Last > H(1, D)
```

Another example: you can construct a price pattern matcher using just data value function.

```
c (0, m5) > c (1, m5) and l (0, m5) > l (1, m5)
```

The above expression will evaluate to 1 (true) when you have a higher close and a higher low on a 5-min basis; otherwise, it will evaluate to 0 (false).

Indicators

Rather than specifying indicators yourself. You can use Indicator Wizard to help you.

All indicators can be used in custom formula columns. That means all built-in indicators, script based indicators, formula based indicators and IDL indicators can be used.

The general form of indicator function is,

```
IndicatorShortName (BarsAgo, TimeFrame.LinkType {, param1 ...})
```

`IndicatorShortName` is the indicator unique name that you can find out from either the indicator reference section or from the Indicator Manager. There is also the Indicator Quick Reference that you can access from the **Help** menu in the main program window.

`BarsAgo` refers to the number of bars ago to return the value from.

`TimeFrame.LinkType` can be written without the `LinkType` portion, in that case, it is assumed that you want to use the close data of each bar as the link series. Thus M5 means 5-min close series.

You can specify a `LinkType`, which is one of the following,

Long form	Short form
Open	O
High	H
Low	L
Close	C
Volume	V
Tick	T
OpenInteres t	OI

So, `D1.C` is the closing prices of daily data.

`{, param1 ...}` is the parameters that the indicator usually takes.

For example, when you want to use the simple moving average of 10 periods on 3-min bars, then the indicator can be represented by,

```
average (0, m3, 10)
```

Or, say you want to refer to the highest high value within the past 10 bars of 15-min data, then you can use

```
hhv (0, m15.high, 10)
```

Maybe you are interested in monitoring the stochastic slow D indicator with 5, 3 parameters on 10-min data, then it will look like the follow,

```
slowd (0, m10, 5, 3)
```

For indicators that do not have parameters, you will still need to specify the bars ago and link information. For example, the indicator truerange does not need any indicator, thus its format is as follows,

```
truerange (M15)
```

which means you are requesting for the latest value of the truerange indicator based on 15-minute timeframe.

```
truerange (2, W)
```

represents the value of true range indicator based on weekly data from 2 bars ago.

Combination

You can combine all the available functions to construct interesting real-time filters. Lets look at a few examples here.

You are interested in highlighting the scenario that a new high was made in the previous trading day and somehow a 5-min stochastic is bottoming. The following formula will get you started,

```
H(1,D) > H(2,D) and slowd (1, M5, 5, 3) < 20 and slowd (m5, 5,
3) >= 20
```

You want to monitor the price action of the symbols that are very close (within 50 cents) to their moving averages.

```
abs (Last -  average (m5, 50)) < 0.5 and abs (Last – xaverage
(m5, 20)) < 0.5
```

Cross Symbol Reference

To refer to another symbol’s data and indicator values, you can use the following format of link access,

```
[Symbol]TimeFrame.LinkType
```

For example, you want to calculate the spreads of all the symbols against the symbol IBM, and then you can write the formula as follows,

```
Last – C([IBM]D)
```

You can also use calculated results from indicators based on another symbol. Say you want to compare the stochastics values between the symbols and the Nasdaq 100 index, then you can do the following,

```
slowd (M5, 5, 3) < slowd ([$NDX]M5, 5, 2)
```

Time Frame Specification

In quote formula, you need to specify time frame when using data series or indicators, e.g.

```
close(0, M5)
hhv (0, M15, 10)
```

Time frame is specified by a code followed by a number.

Code can be one of the following.

Code	Meaning
T	Tick
S	Second
M	Minute
H	Hour
D	Daily
W	Weekly

Time frame period is a positive integer.

For example, M5 represents a 5-minute series. D2 represents a 2-day bar based data series.

Overriding Number of Days

For intraday series (tick, second, minute, hour), you can specify an override number of days in the time frame by using a `:N` suffix. For example, in

```
xaverage(0,M15:20,200)
```

The time frame is `M15:20`. This means the time frame is 15-minutes, and 20-days of data is loaded. This feature is used for indicators that require long data to stabilize, and you do not want to adjust RAM Cache settings for to include more data.

When you are linking to a specific series (OHLC) of the data, use the `:N` before the series specification, e.g.

```
xaverage(0,M15:20.high,200)
```

Bidtick Support

The bidtick field is displayed a string as quote window field.

If you query the bidtick field in formula, the field will return a value. The value 0 represents a downtick and the value 1 represents an uptick.

Bidtick in formula should be used only to derive other values. To properly display bidtick value for exchange compliant display, you should directly add the bidtick field to a quote window, dynamic grid, etc.

Functions in Formula

Generic Functions

Some functions are available to all formulas, regardless of where the formula is applied.

For example, the if function has a syntax:

```
if(boolean expression, answer1, answer2)
```

The `if` function returns `answer1` if `boolean expression` evaluates to true. It returns `answer2` if `boolean expression` evaluates to false.

Generic Functions - Date Time Related

Formula (and NeoTicker® in general) uses Windows's internal date time format for internal processing. Internal date time format is a floating point number. The integral part of the number is a day and the fraction part of the number is the time. Therefore if you add 1 to an internal date time value, the sum is the date time for the next day, same time.

Formula provides a full suite of functions to help you encode and decode date time into internal format.

day(datetime)

Takes an internal date time value, returns the day of month in integer.

dayofweek (expression)

Returns day of week (Sunday - 0, Monday - 1, ... Saturday - 6) from the expression that evaluates to internal date time format.

indaterange(date, from, to)

Whether the given date is inside the from to time. Parameters are in internal date time format.

indatetimerange(datetime, from, to)

Whether the given date time is inside the from to time. Parameters are in internal date time format.

intimerange(time, from, to)

Whether the given time is inside the from to time. Parameters are in internal date time format.

makedate(y,m,d)

Creates an internal date time value from date parameters.

makedatetime(y,m,d,h,n,s)

Creates an internal date time value from date and time parameters.

maketime(h,n,s)

Creates an internal date time value from time parameters.

month(datetime)

Takes an internal date time value, returns the month in integer.

NTNow

Returns the current time of the server.

year(datetime)

Takes an internal date time value, returns the year in integer.

hour(datetime)

Takes an internal date time value, returns the hour in integer.

minute(datetime)

Takes an internal date time value, returns the minute in integer.

second(datetime)

Takes an internal date time value, returns the second in integer.

Time2Number(datetime)

Takes an internal date time value, return time as a fraction (0.hhnnss). Note that internal date time value is represented by a number that can not be easily interpreted by human. This function reformats the internal date time value in a readable form. For example, if datetime represents 2005/12/24 3:45:10pm, Time2Number(datetime) returns 0.154510.

Date2Number(datetime)

Takes an internal date time value, return date as an integer (yyyymmdd). Note that internal date time value is represented by a number that can not be easily interpreted by human. This function reformats the internal date time value in a readable form. For example, if datetime represents 2005/12/24 3:45:10pm, Time2Number(datetime) returns 20051224.

DateTime2Number(datetime)

Takes an internal date time value, return date time as floating point value (yyyymmdd.hhnnss). Note that internal date time value is represented by a number that can not be easily interpreted by human. This function reformats the internal date time value in a readable form. For example, if datetime represents 2005/12/24 3:45:10pm, Time2Number(datetime) returns 20051224.154510.

Generic Functions - Math Related

absvalue (expression)

Returns the absolute value of the expression.

arccos (expression)

Returns arc cosine of expression. Result is in radian.

arcsin (expression)

Returns arc sine of expression. Result is in radian.

arctan (expression)

Returns arc tangent of expression. Result is in radian.

cos (expression)

Returns cosine of expression. Expression is in radian.

deg2rad (expression)

Converts degree to radian. Expression is in degree. Result is in radian.

exp (expression)

Returns e to the power expression.

exp10 (expression)

Returns 10 to the power expression.

exp2 (expression)

Returns 2 to the power expression.

frac (expression)

Returns fraction portion of the expression.

int (expression)

Returns integer portion of the expression.

ln(expression)

Returns Natural Log.

log10(expression)

Returns Log 10.

log2(expression)

Returns Log 2.

maxlist(expression1, ... ,expressionN)

Returns the maximum value from a list of expressions.

minlist(expression1, ... ,expressionN)

Returns the minimum value from a list of expressions.

mod (expression1, expression2)

Returns the value of expression1 in modules of expression2.

pi

Returns the constant pi.

power (expression1, expression2)

Returns expression1 to the power expression2.

rad2deg (expression)

Converts degree to radian. Expression is in degree. Result is in radian.

round (expression)

Returns the rounded integer of the expression.

signvalue (expression)

Returns 1 for positive input, -1 for negative input and 0 otherwise.

sin (expression)

Returns sine of expression. Expression is in radian.

sqrt (expression)

Returns square root of the expression.

tan (expression)

Returns tangent of expression. Expression is in radian.

hex2int(hexstring)

Hexstring is a quoted string representing a hexadecimal number. Hex2int returns the integer value of the hex number. For example, hex2int("FF") returns 255.

Generic Functions - Color Related

If a formula is expected to return a color, the color must be an integer value of internal color representation. Functions in this section help you construct internal color values.

rgb(r, g, b)

Converts RGB color values to internal color value. The parameters r, g, b must be from 0 to 255. For example,. rgb(255, 255, 0) returns a yellow color.

hsl(h, s, l)

Converts a color in HSL color space to internal color value. The parameters h, s, l must be from 0 to 1.

hsv(h, s, v)

Coverts a color in HSV color space to internal color value. The parameters, h, s, v must be from 0 to 1.

Generic Functions - GHeap Related

GHeap is a common storage that is global within NeoTicker®. Different parts of NeoTicker® can use GHeap to communicate to each other. For example, a chart indicator can calculate a value, store it in GHeap and a quote window can read that value.

GHeap is read-only in formula. To access a "slot" of GHeap, it is required you know a pre-defined slot number or slot name. These must be obtained from the item that stores the value in the GHeap.

See ***Heap, PHeap and GHeap Objects*** (on page 1549) for GHeap programming topics.

gHeapValue(slot)

Returns a real number stored in GHeap. Slot is an integer, which is the pre-determined slot number.

gHeapReal(SlotName)

Returns a real number stored in GHeap. Slotname is a string, which is the pre-determined slot name.

gHeapInt(SlotName)

Returns an integer stored in GHeap. Slotname is a string, which is the pre-determined slot name.

gHeapRealList(Slotname, Index)

Returns an item of a real number list stored in GHeap. Slotname is string, which is the pre-determined slot name. Index is a 0-based index into the list.

gHeapIntList(Slotname, Index)

Returns an item of an integer list stored in GHeap. Slotname is string, which is the pre-determined slot name. Index is a 0-based index into the list.

gHeapRealTable(Slotname, Row, Column)

Returns a cell in a real number table stored in GHeap. Slotname is string, which is the pre-determined slot name. Row and column are a 0-based index into the table.

gHeapIntTable(Slotname, Row, Column)

Returns a cell in an integer table stored in GHeap. Slotname is string, which is the pre-determined slot name. Row and column are 0-based index into the table.

Generic Functions - Random Numbers

randomStreamRandomize(condition, streamid)

When `conidtion` evaluated to 1, initialize the random stream `streamid` randomly. Generally you want to use the `firstcall` function for condition. 100 streams are available with `streamid` from 0 to 99.

randomStreamSeed(condition, streamid, seed)

When `conidtion` evaluated to 1, initialize the random stream `streamid` with random seed `seed`. Generally you want to use the `firstcall` function for condition. 100 streams are available with `streamid` from 0 to 99.

randomStreamDouble(streamid)

randomStreamInteger(streamid, ainteger)

Generic Functions - Others

Choose (condition1, result1, condition2, result2, ... , otherwise)

`Choose` function takes a list of conditions and results as input, and returns result if the corresponding condition is true. For example,

```
Choose($a > $b, 1, $a < $b, -1, 0)
```

returns 1 if $a > $b, -1 if $a < $b, 0 if $a = $b.

@ChooseString (SingleString)

@ChooseString (condition1, result1, condition2, result2, ... {, otherwise})

@ChooseString function takes a list of conditions and results as input, and returns string result if the corresponding condition is true. For example,

```
@ChooseString($a > $b, "Apple", $a < $b, "Banana", "Cranberry")
```

returns "Apple" if $a > $b, "Banana" if $a < $b, otherwise "Cranberry".

If there is only one single parameter provided (`SingleString`), then the parameter is returned as the string result.

If there is no `otherwise` parameter provided, the original non-string result from the formula will be returned.

This function is useful for use in quote formulas.

Debug(item1, item2, item3, ...)

Displays debug information in script error log (**Program>Script Error Log**). Item1, item2, etc can be double quote string, or formula.

If (condition, expression1, expression2)

Returns the value of expression1 if condition is not equal to zero. Otherwise expression2 will be returned.

PlaySound(condition, soundfilename)

Plays sound when condition is true. Condition is a formula, soundfilename is a double quote string. If soundfilename is a relative path, the file will be searched from recognized media folder.

Report(ReportWindow, Cond, Item1, Item2, ...)

Displays information in a report window.

ReportWindow is the name of the report window in double quote string. If ReportWindow is an empty string, the items are sent to the first report window.

Cond is the condition for display. When Cond is true (1), items are displayed. When Cond is false(0), items are not displayed. Cond can be any formula.

Item1, Item2, etc are items to be displayed. They can be double quote string, or formula.

ReportFormat(ReportWindow, Cond, format1, Item1, format2, Item2, ...)

Identical to Report function, except ReportFormat requires an additional formatting string to format the items for display.

The format string follows a C-style convention. For example,

```
ReportFormat("", 1, "%d",  $a, "%f", $b)
```

will format $a as an integer, $b as a float number.

UDSUpdateWithVolume (Cond, Symbol, Price, Volume)

`UDSUpdateWithVolume` will update the User Define Symbol of External type, `Symbol`, with the latest price set to `Price`, and trade volume set to `Volume`, provided the condition `Cond` is evaluated to true (non-zero).

What is Context

Formula is used many place in NeoTicker®. The context of the formula is the environment the formula is running in. Context affects the functions that are available to the formula.

For example, quote window column is a context. In a formula that runs in quote window column has the access to quote window fields such as net, last day close price, etc. These values are not available in other context such as time charts.

Context also affects indicator parameters. In a quote window column, you need to provide information about bar size and time, e.g. `bband(M1, 10, 0)` evaluates the bband indicator in quote window column. In a time chart, you need to provide a data link, e.g. `bband (data1, 10, 0)` evaluates the bband indicator in a time chart.

Color Constants

For formula that handles color (e.g. color bars), the following list are the pre-defined color constants (the color constants use standard Windows color naming):

```
clAqua
clBlack
clBlue
clDkGray
clFuchsia
clGray
clGreen
clLime
clLtGray
clMaroon
clNavy
clOlive
clPurple
clRed
clSilver
clTeal
clWhite
clYellow
```

You can also use integer for color. The integer color is decimal representation of the following hexadecimal number BBGGRR, where BB is 2-digit hex representing blue, GG is 2-digit hex representing green, RR is 2-digit hex representing red. For example, 000066 (hex) = 102 (decimal). This is a dark red color.

Context-Driven Functions

Many formula functions are context-driven functions. Context-driven functions are available in specific situation and function window. For example, when you are specifying an alert for the quote window, all the quote window fields are available as functions for your usage. These functions are not available in time chart alerts.

As a specific example, the functions `last`, `net`, etc. can be used in quote window alerts but have no meaning in time chart alerts. In contrast, the functions `data1`, `indicator1`, `TL`, etc. are available in time chart alerts but have no meaning in quote window alerts.

Context-Driven Functions for Quote Window Formula Columns

The followings are context-driven functions for quote window formula columns:

- All quote window fields. The function name is the same as field names. For example, the last function returns the same value as the last field.
- Data values when the symbol is treated as a data series. Data values such as `open`, `high`, `low`, `close` are available and can be referenced with time frame information. For example, `close(0, M5)` is the current 5-minute bar.
- All indicators. For example, average(0, m3, 10) returns the current value of the 3-minute 10 period moving average.
- The `symbolis` function. `Symbolis` takes one string parameter and returns 1, 0 if the symbol matches the current symbol in context.
- The Level 2 functions

Symbol Info Functions

These functions return information about a symbol. These information is entered in Symbol Info Manager.

Symbol Info Function	Meaning
MinTickSize, MinTickSize(Symbol)	Returns minimum tick size of symbol. If no symbol is specified, the default symbol is used
DefaultOrderSize, DefaultOrderSize(Symbol)	Returns default order size of symbol. If no symbol is specified, the default symbol is used
ScaleOrderSize, ScaleORderSize(Symbol)	Returns scale order size of symbol. If no symbol is specified, the default symbol is used

Level 2 Functions

The Level 2 functions return Level related values so they can be used in quote window formula.

You can only receive Level 2 data if you subscribe to the Level 2 service of your data vendor.

Level 2 Function	Meaning
`L2Bid(n)`	The `n`-th best bid offer (`n` starts from 0)
`L2Ask(n)`	The `n`-th best ask offer (`n` starts from 0)
`L2BidSize(n)`	The size of the `n`-th best bid offer

`L2AskSize(n)`	The size of the `n`-the best ask offer
`L2BidOrders(n)`	The number of orders of the n-th best bid offer. This function is usually useful only for exchange that broadcast number of orders like CME
`L2AskOrders(n)`	The number of orders of the n-th best ask offer. This function is usually useful only for exchange that broadcast number of orders like CME
`L2BidAccOrders(depth)`	Accumulated total number of bid orders from best bid to best bid - depth
`L2AdkAccOrders(depth)`	Accumulated total number of ask orders from best ask to best ask + depth
`L2BidVWAP(v)`	Volume weight average price from the best bid to best bid - v
`L2AskVWAP(v)`	Volume weight average price from the best bid to best bid + v
`L2BidVol(depth)`	For a depth (e.g. 0.5), returns total bid size from top bid to depth
`L2AskVol(depth)`	For a depth (e.g. 0.5), returns total ask size from top bid to depth
`L2NetBidAskVol(depth)`	Returns L2BidVol - L2AskVol. Positive value indicates bullishness. Negative value indicates bearishness

Context-Driven Functions for Quote Window Alerts

The following functions are context-driven functions for quote window alerts:

- All quote window fields. The function name is the same as field names. For example, the last function returns the same value as the last field.

Context-Driven Functions for Dynamic Grid

Dynamic grid uses the same formula context as quote window formula. See ***Context-Driven Functions for Quote Window Formula Columns*** (on page 306).

In addition, Dynamic Grid supports functions that access cell level information.

- `cell(index)` returns cell value. `Index` is an integer ranges from 0 to number of cells - 1.
- `cell(row, column)` returns cell value. `Row` is an integer ranges from 1 to number of rows. `Column` is an integer ranges from 1 to number of columns.
- `rcell(offset)` returns value of cell that is offset from current cell. `Offset` is an integer.
- `rcell(row_offset, col_offset)` returns value of cell that is offset from current cell. `Row_offset` and `col_offset` are integers.
- `rownum` returns the row number of current cell.
- `colnum` returns the column number of current cell.
- `cellbylabel(label)` returns the value of cell identified by `label. Label` is defined in cell setup, under the **Content** tab. There is no need to put quotes around label, e.g. you will use `cellbylabel(hello)` and `cellbylabel(goodbye)` to refer to cells identified by the labels `hello` and `goodbye.`

Context-Driven Functions for Time Chart, Pattern Scanners and Inside Formula Indicators

The following are the context-driven functions for formulas in time charts, pattern scanners and formula indicators (fml, fml2, fml3) :

- All indicators are functions. Use the indicator's short name (e.g. mov) to call the indicator. Use Indicator Wizard to help you specify the parameters.
- Data and indicator links, e.g. `data1`, `data2`, etc and their fields (O, H, L, C, latest bid/ask/trade, tick statistics, etc). See ***Accessing Data Links and Fields*** (on page 278).
- `Futuredatetime` function returns the date time value of a specified number of bars in the future. For example:

  `futuredatetime(3)` returns the date time 3 bars into the future.

  `futuredatetime(0)` returns the date time of current bar.

- CurrentBar(N) returns a sequential index starting from 1. When the script is first called to do calculation, usually that is the first bar right after the indicator's Min Bars setting, the current bar is set to 1.
- CurrentDateTime returns the current date time (in internal format) in the locale of the hosting function window. If the hosting window is using the NeoTicker default locale, then the returning value is the same as the generic function NTNow.
- Firstcall returns 1 if it is the first time the indicator is called, 0 otherwise.

Symbol Info Functions

The following functions let you query symbol information entered in Symbol Info Manager.

- ScaleOrderSize returns the Scale Order Size value of the symbol.

Time Chart Alert Functions

The following are context-driven functions for time chart alerts:

- Data series functions let you access values of any data series in the time chart. For example, close(0) returns the close value of the current bar of the first data series.
- Indicator functions let you access values of any indicator in the time chart. For example, Indicator1(0) returns the value of the current bar of the first indicator.
- Drawing tool functions let you access values of trend lines (TL) and horizontal lines (HL) in the time chart. For example, TL1(0) returns the value of the first trend line.
- Alert function for triggering alerts within the formula.

Misc

ParamIs is a string comparison function that allows you to compare the user parameter for formula indicator and a specified string. The syntax is ParamIs(ParamIndex, String). See ***Accessing User Specified Parameters*** (on page 282) for more information.

Context-Driven Functions for Clusters

Cluster window uses formula in two places: X, Y coordinates and region alerts.

X, Y coordinates uses the same context as quote window formulas, see ***Context-Driven Functions for Quote Window Formula Columns*** (on page 306) for more information.

Region alerts support a set of functions that help determine whether a series is inside a region or not, see ***Regions and Alerts*** (on page 489) for more information.

Context-Driven Functions for Tick Filter

Formula-based tick filter has context function to query the price, volume and time stamp of a tick. For more information, refer to ***Tick Filter Formula Reference*** (on page 994).

Context-Driven Functions (Time Frame Related)

Functions listed here are valid when the underlying context has a time frame, e.g. time chart, quote window.

TradingTimeStart

Returns a number representing the time trading starts in Windows internal date time format. See ***Generic Functions - Date Time Related*** (on page 296) on how to interpret the value.

TradingTimeEnd

Returns a number representing the time trading ends in Windows internal date time format. See *Generic Functions - Date Time Related* **(on page 296) on how to interpret the value.**

TradingTime24Hours

Returns 1 if it is trading 24 hours a day. 0 otherwise.

Creating Trading System with Formula

Usage

You can create trading system with formulas.

Advantages:

- Easy to use
- More flexible than Backtest EZ.

How It Works

Trading system with formulas is a specialize type of formula indicator. Basically you will create an indicator with formula language, and the indicator contains trading commands.

How to Create

1. Create an indicator using formula language. You can read ***Tutorial: Making an Indicator with Formula Language, Example 1*** (on page 364) to see how this is done.
2. When you set up the indicator in the indicator setup dialog, check **Trading System UI**.
3. Write the system, install and run just as if it is a regular indicator.

In step 2, you specify that the indicator is to be treated as trading system. This means you will be able to use trading system specific feature in charts, such as trading system reporting.

Thinking in Positions

Formula trading system works on position level. You will be thinking in terms of long and short positions. This saves you from the details of individual buy/sell orders and lets you focus on trading system design.

NeoTicker® will do the conversion from position changes to buy sell orders for you.

Fill or Kill

All orders in formula trading systems are fill or kill orders. This means if an order is not filled at the next bar (e.g. due to limit price), the order is cancelled.

Example

The following listing is an example of a stock trading system written in formula language:

```
buysignal := xabove (data1, average (data1, 20));
sellsignal := xbelow (data1, average (data1, 20));
```

```
longatmarket (buysignal, 100);

shortatmarket (sellsignal, 100);

longexitlimit (
   openpositionlong > 0 and (OpenPositionAbsSize >= 100),
   openpositionaverageentryprice * 1.05,
   OpenPositionAbsSize / 2);

longexitstop (
   openpositionlong > 0 and (OpenPositionAbsSize < 100),
   openpositionaverageentryprice,
   OpenPositionAbsSize);

shortexitlimit (
   openpositionshort > 0 and (OpenPositionAbsSize >= 100),
   openpositionaverageentryprice * 0.95,
   OpenPositionAbsSize / 2);

shortexitstop (
   openpositionshort > 0 and (OpenPositionAbsSize < 100),
   openpositionaverageentryprice,
   OpenPositionAbsSize);

plot1 := CurrentEquity;
```

We will dissect the trading system. We will only look at the long side of the system as the short side of the system is symmetric to the long side. First, the trading system defines the buy signal:

```
   buysignal := xabove (data1, average (data1, 20));
```

The next part defines when to long. The trading system will long at market price when the buy signal becomes true:

```
   longatmarket (buysignal, 100);
```

The next part defines when to exit the long position. The following line specifies when the system is in a long position for equal or more than 100 shares, it will exit half the long position when the position is making 5% profit (1.05).

```
   longexitlimit (
      openpositionlong > 0 and (OpenPositionAbsSize >= 100),
      openpositionaverageentryprice * 1.05,
      OpenPositionAbsSize / 2);
```

The next part is the protection for the long position:

```
   longexitstop (
      openpositionlong > 0 and (OpenPositionAbsSize < 100),
      openpositionaverageentryprice,
      OpenPositionAbsSize);
```

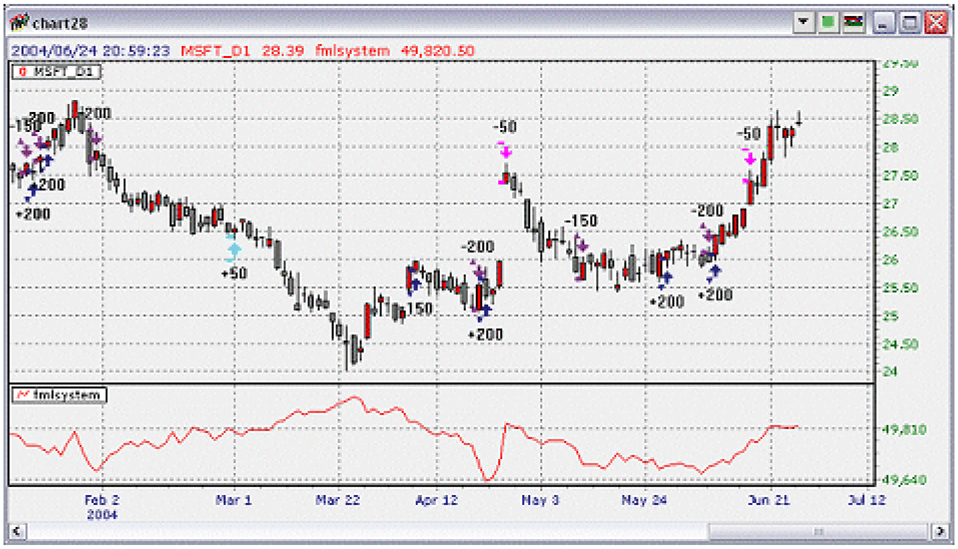
The system uses the average entry price as the stop price. Note that because the protection is triggered only when the open position is strictly less than 100 shares, the logic follows that the system will not have a stop order unless some profit has already been taken by the limit order. This is due to the fact that the system enters the position at 100 shares, and the only way to reduce the 100 shares is by the limit order.

What the stop order does is once the price hits the average entry price, the system will exit the long position by issuing a market order.

The next part is what the system returns. It is customary for trading systems to return the current equity position so the output of the trading system will be its equity curve.

```
plot1 := CurrentEquity;
```

This is how the trading system would look like in a chart:



Monitoring a Trading System

After you apply a trading system to a chart, you will want to monitor if there is any new orders.

For real-time version of NeoTicker®, you can use the system monitor window. System monitor is particularly suitable for monitoring intraday trading systems.

If you want to monitor an EOD trading system or if you use NeoTicker® EOD, you can use the reporting facility in the chart:

1 Open a report window.

2 Select the trading system in the chart.

3 Choose either **Trading System>Quick Status** or **Trading System>Current Status**.

For an EOD system, you will want to do this after the latest bar from the current trading day is available.

Formula Order Placement Methods (Regular)

LongAtMarket (Condition, PosSize)

LongAtMarket (Condition, PosSize, Comment)

Long at market price when `Condition` is true for `PosSize` shares/contracts. `Condition` and `PosSize` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my market long"`. The comment is shown in reporting to help you distinguish different orders.

LongLimit (Condition, EntryPrice, PosSize)

LongLimit (Condition, EntryPrice, PosSize, Comment)

Long at `EntryPrice` price when `Condition` is true for `PosSize` shares/contracts `.Condition, EntryPrice` and `PosSize` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my limit long"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a limit order not filled at the specified price. In this case the order is cancelled.

LongStop (Condition, StopPrice, PosSize)

LongStop (Condition, StopPrice, PosSize, Comment)

Issue a stop order to enter a long position when `Condition` is true. A stop order is triggered when `StopPrice` is hit, then a market order will be issued for `PosSize` shares/contracts. `Condition, StopPrice` and `PosSize` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my long stop"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a stop order not triggered, i.e. when the stop price is not hit. In this case, the stop order is cancelled.

LongExitAtMarket (Condition, PosSize)

LongExitAtMarket (Condition, PosSize, Comment)

Issue a market order to exit a long position when `Condition` is true for `PosSize` shares/contracts. `Condition` and `PosSize` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my long exit market"`. The comment is shown in reporting to help you distinguish different orders.

LongExitLimit (Condition, ExitPrice, PosSize)

LongExitLimit (Condition, ExitPrice, PosSize, Comment)

Issue a limit order to exit a long position at `ExitPrice` when `Condition` is true for `PosSize` shares/contracts. `Condition, ExitPrice` and `PosSize` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my long exit limit"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a limit order not filled at the specified price. In this case the order is cancelled.

LongExitStop (Condition, StopPrice, PosSize)

LongExitStop (Condition, StopPrice, PosSize, Comment)

Issue a stop order to exit a long position when `Condition` is true. A stop order is triggered when `StopPrice` is hit, then a market order will be issued for `PosSize` shares/contracts. `Condition, StopPrice` and `PosSize` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my long stop"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a stop order not triggered, i.e. when the stop price is not hit. In this case, the stop order is cancelled.

LongNextClose(Condition, PosSize)

LongNextClose(Condition, PosSize, Comment)

Long at the next bar's close price when `Condition` is true for `PosSize`. You can include an optional string parameter `Comment`, e.g. "my next close long". The comment is shown in reporting to help you distinguish different orders.

Trading system explicitly forbids orders that depends on current bar's close to ensure maximum consistency between historical testing and real-time execution of trading system.

LongExitNextClose(Condition, PosSize)

LongExitNextClose(Condition, PosSize, Comment)

Exit long position at the next bar's close price when `Condition` is true for `PosSize`. You can include an optional string parameter `Comment`, e.g. "my next close exit long". The comment is shown in reporting to help you distinguish different orders.

Trading system explicitly forbids orders that depends on current bar's close to ensure maximum consistency between historical testing and real-time execution of trading system.

ShortAtMarket (Condition, PosSize)

ShortAtMarket (Condition, PosSize, Comment)

Short at market price when `Condition` is true for `PosSize` shares/contracts. `Condition` and `PosSize` are any expressions. You can include an optional string parameter `Comment`, e.g. "my market short". The comment is shown in reporting to help you distinguish different orders.

ShortLimit (Condition, EntryPrice, PosSize)

ShortLimit (Condition, EntryPrice, PosSize, Comment)

Short at EntryPrice price when Condition is true for PosSize shares/contracts . Condition, EntryPrice and PosSize are any expressions. You can include an optional string parameter Comment, e.g. "my limit short". The comment is shown in reporting to help you distinguish different orders.

It is possible for a limit order not filled at the specified price. In this case the order is cancelled.

ShortStop (Condition, StopPrice, PosSize)

ShortStop (Condition, StopPrice, PosSize, Comment)

Issue a stop order to enter a short position when Condition is true. A stop order is triggered when StopPrice is hit, then a market order will be issued for PosSize shares/contracts. Condition, StopPrice and PosSize are any expressions. You can include an optional string parameter Comment, e.g. "my short stop". The comment is shown in reporting to help you distinguish different orders.

It is possible for a stop order not triggered, i.e. when the stop price is not hit. In this case, the stop order is cancelled.

ShortExitAtMarket (Condition, PosSize)

ShortExitAtMarket (Condition, PosSize, Comment)

Issue a market order to exit a short position when Condition is true for PosSize shares/contracts. Condition and PosSize are any expressions. You can include an optional string parameter Comment, e.g. "my short exit market". The comment is shown in reporting to help you distinguish different orders.

ShortExitLimit (Condition, ExitPrice, PosSize)

ShortExitLimit (Condition, ExitPrice, PosSize, Comment)

Issue a limit order to exit a short position at ExitPrice when Condition is true for PosSize shares/contracts. Condition, ExitPrice and PosSize are any expressions. You can include an optional string parameter Comment, e.g. "my short exit limit". The comment is shown in reporting to help you distinguish different orders.

It is possible for a limit order not filled at the specified price. In this case the order is cancelled.

ShortExitStop (Condition, StopPrice, PosSize)

ShortExitStop (Condition, StopPrice, PosSize, Comment)

Issue a stop order to exit a short position when Condition is true. A stop order is triggered when StopPrice is hit, then a market order will be issued for PosSize shares/contracts. Condition, StopPrice and PosSize are any expressions. You can include an optional string parameter Comment, e.g. "my short stop". The comment is shown in reporting to help you distinguish different orders.

It is possible for a stop order not triggered, i.e. when the stop price is not hit. In this case, the stop order is cancelled.

ShortNextClose(Condition, PosSize)

ShortNextClose(Condition, PosSize, Comment)

Short at the next bar's close price when `Condition` is true for `PosSize`. You can include an optional string parameter `Comment`, e.g. "my next close short". The comment is shown in reporting to help you distinguish different orders.

Trading system explicitly forbids orders that depends on current bar's close to ensure maximum consistency between historical testing and real-time execution of trading system.

ShortExitNextClose(Condition, PosSize)

ShortExitNextClose(Condition, PosSize, Comment)

Exit short position at the next bar's close price when `Condition` is true for `PosSize`. You can include an optional string parameter `Comment`, e.g. "my next close exit short". The comment is shown in reporting to help you distinguish different orders.

Trading system explicitly forbids orders that depends on current bar's close to ensure maximum consistency between historical testing and real-time execution of trading system.

Formula Order Placement Methods (Next Open)

Next open order placement methods are not available in NeoTicker® EOD.

The following methods place order with a price based on next open. An offset value is added to the next open for the ordering price.

For example, `LongNextOpenOffsetLimit` turns the system to long using next open price plus an offset as the limit price.

Here is the filling mechanism when testing against historical data:

1 In the current bar, a next open order is placed. The order is not filled.

2 At the next bar, the OHLC values are available. As long as the next bar prices is better than or equal to the limit price (open price + offset), the order is considered to be filled.

When you place a next open order in real-time, here is the filling mechanism:

1 In the current bar, a next open order is placed. The order is not filled.

2 When the current bar is completed when the first tick of next bar arrives, the order price becomes available. At this stage, if the trading system is updated by tick, the order will be filled when the arriving ticks matches the price. If the trading system is non-updated by tick, the order will be filled similarly to historical testing.

3 If the trading system is connected to a real-life broker or Trade Simulator through Order Interface, when the first tick of the next bar arrives, the order now has the order price and is sent to broker or Trade Simulator for filling.

Warning: although technically orders based on next open do not reveal future information, it is possible to use such orders combined with stop orders to create systems that are close to impossible to execute in real-time. You need to take extra caution when creating and deploying systems that are based on next open.

LongNextOpenOffsetLimit(Condition, Offset, Size)

LongNextOpenOffsetLimit(Condition, Offset, Size, Comment)

Long at next open plus `Offset` when `Condition` is true for `Size` shares/contracts . `Condition, Offset` and `Size` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my limit long"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a limit order not filled at the specified price. In this case the order is cancelled.

LongNextOpenOffsetStop(Condition, Offset, Size)

LongNextOpenOffsetStop(Condition, Offset, Size, Comment)

Issue a stop order to enter a long position when `Condition` is true. A stop order is triggered when next open plus `Offset` is hit, then a market order will be issued for `Size` shares/contracts. `Condition, Offset` and `Size` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my long stop"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a stop order not triggered, i.e. when the stop price is not hit. In this case, the stop order is cancelled.

LongExitNextOpenOffsetLimit(Condition, Offset, Size)

LongExitNextOpenOffsetLimit(Condition, Offset, Size, Comment)

Issue a limit order to exit a long position at next open plus `Offset` when `Condition` is true for `Size` shares/contracts. `Condition, Offset` and `Size` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my long exit limit"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a limit order not filled at the specified price. In this case the order is cancelled.

LongExitNextOpenOffsetStop(Condition, Offset, Size)

LongExitNextOpenOffsetStop(Condition, Offset, Size, Comment)

Issue a stop order to exit a long position when `Condition` is true. A stop order is triggered when next open plus `Offset` is hit, then a market order will be issued for `Size` shares/contracts. `Condition, Offset` and `Size` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my long stop"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a stop order not triggered, i.e. when the stop price is not hit. In this case, the stop order is cancelled.

ShortNextOpenOffsetLimit(Condition, Offset, Size)

ShortNextOpenOffsetLimit(Condition, Offset, Size, Comment)

Short at next open plus `Offset` when `Condition` is true for `Size` shares/contracts . `Condition, Offset` and `Size` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my limit short"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a limit order not filled at the specified price. In this case the order is cancelled.

ShortNextOpenOffsetStop(Condition, Offset, Size)

ShortNextOpenOffsetStop(Condition, Offset, Size, Comment)

Issue a stop order to enter a short position when `Condition` is true. A stop order is triggered when next open plus `Offset` is hit, then a market order will be issued for `Size` shares/contracts. `Condition, Offset` and `Size` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my short stop"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a stop order not triggered, i.e. when the stop price is not hit. In this case, the stop order is cancelled.

ShortExitNextOpenOffsetLimit(Condition, Offset, Size)

ShortExitNextOpenOffsetLimit(Condition, Offset, Size, Comment)

Issue a limit order to exit a short position at next open plus `Offset` when `Condition` is true for `Size` shares/contracts. `Condition, Offset` and `Size` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my short exit limit"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a limit order not filled at the specified price. In this case the order is cancelled.

ShortExitNextOpenOffsetStop(Condition, Offset, Size)

ShortExitNextOpenOffsetStop(Condition, Offset, Size, Comment)

Issue a stop order to exit a short position when `Condition` is true. A stop order is triggered when next open plus `Offset` is hit, then a market order will be issued for `Size` shares/contracts. `Condition, StopPrice` and `Size` are any expressions. You can include an optional string parameter `Comment`, e.g. `"my short stop"`. The comment is shown in reporting to help you distinguish different orders.

It is possible for a stop order not triggered, i.e. when the stop price is not hit. In this case, the stop order is cancelled.

Formula Order Placement Methods (Time In Force)

Time In Force

Time In Force (TIF) orders are extensions to regular and next open orders. TIF orders has an additional parameter to specify TIF for the order. TIF is the life span of the order. Three types of TIF is supported:

- `otfDay` - Day order. The order, if not filled, will be cancelled at the end of trading day.
- `otfFillOrKill` - Fill or kill. The order, if not filled in the bar it is placed, it will be cancelled. The behavior is identical to the behavior of regular and next open orders.
- `otfGoodTilCancel` - The order will not be cancelled until the cancel function is called

In addition, you can place an optional comment parameter in a TIF order. The comment is shown in various places that track orders, e.g. Account Manager, and can be used for cancelling TIF orders.

The following are examples of TIF orders:

`LongAtMarketTIF(1, 100, otfDay)` - Day market order

`LongStopTIF(1, 200, otfGoodTilCancel, "My Stop")` - Good til cancel stop order with comment

When you place a TIF order, an order id is returned that you can use to reference the order later.

Cancelling An Order

You can cancel `otfDay` and `otfGoodTilCancel` orders.

`CancelOrder($mycond, $myid)` - Cancel the order referenced by `$myid` when `$mycond` is true.

`CancelOrdersByComment($mycond, "My Stop")` - Cancel the order with the comment `"My Stop"` when `$mycond` is true

Reference (TIF Order Placement Functions)

TIF order placement functions work exactly the same as regular and next open orders, except with the TIF and optional comment parameters. You can refer to ***Formula Order Placement Methods (Regular)*** (on page 315) and ***Formula Order Placement Methods (Next Open)*** (on page 318).

The following list are all the TIF order placement functions (curly brackets denote optional comment parameter):

- `LongAtMarketTIF (cond, size, timeinforce {, comment})`
- `LongExitAtMarketTIF (cond, size, timeinforce {, comment})`

- ShortAtMarketTIF (cond, size, timeinforce {, comment})
- ShortExitAtMarketTIF (cond, size, timeinforce {, comment})
- LongNextCloseTIF (cond, size, timeinforce {, comment})
- LongExitNextCloseTIF (cond, size, timeinforce {, comment})
- ShortNextCloseTIF (cond, size, timeinforce {, comment})
- ShortExitNextCloseTIF (cond, size, timeinforce {, comment})
- LongLimitTIF (cond, price, size, timeinforce {, comment})
- LongExitLimitTIF (cond, price, size, timeinforce {, comment})
- ShortLimitTIF (cond, price, size, timeinforce {, comment})
- ShortExitLimitTIF (cond, price, size, timeinforce {, comment})
- LongStopTIF (cond, price, size, timeinforce {, comment})
- LongExitStopTIF (cond, price, size, timeinforce {, comment})
- ShortStopTIF (cond, price, size, timeinforce {, comment})
- ShortExitStopTIF (cond, price, size, timeinforce {, comment})

- LongNextOpenOffsetLimitTIF (cond, offset, size, timeinforce {, comment})
- LongExitNextOpenOffsetLimitTIF (cond, offset, size, timeinforce {, comment})
- ShortNextOpenOffsetLimitTIF (cond, offset, size, timeinforce {, comment})
- ShortExitNextOpenOffsetLimitTIF (cond, offset, size, timeinforce {, comment})
- LongNextOpenOffsetStopTIF (cond, offset, size, timeinforce {, comment})
- LongExitNextOpenOffsetStopTIF (cond, offset, size, timeinforce {, comment})
- ShortNextOpenOffsetStopTIF (cond, offset, size, timeinforce {, comment})
- ShortExitNextOpenOffsetStopTIF (cond, offset, size, timeinforce {, comment})

- `BuyAtMarketTIF (cond, size, timeinforce {, comment})`
- `SellAtMarketTIF (cond, size, timeinforce {, comment})`
- `BuyNextCloseTIF (cond, size, timeinforce {, comment})`
- `SellNextCloseTIF (cond, size, timeinforce {, comment})`
- `BuyLimitTIF (cond, price, size, timeinforce {, comment})`
- `SellLimitTIF (cond, price, size, timeinforce {, comment})`
- `BuyStopTIF (cond, price, size, timeinforce {, comment})`
- `SellStopTIF (cond, price, size, timeinforce {, comment})`

- `BuyNextOpenOffsetLimitTIF (cond, offset, size, timeinforce {, comment})`
- `SellNextOpenOffsetLimitTIF (cond, offset, size, timeinforce {, comment})`
- `BuyNextOpenOffsetStopTIF (cond, offset, size, timeinforce {, comment})`
- `SellNextOpenOffsetStopTIF. (cond, offset, size, timeinforce {, comment})`

Reference (Supplementary Functions)

Only open orders that are placed within the same formula trading system can be cancelled.

`OpenOrderIDByComment(comment)`

Search for an order by `comment`. Returns order ID. If not found -1 is returned.

`CancelOrder(cond, orderid)`

Cancels an order with `orderid` when `cond` is true.

`CancelAllOpenOrders(cond)`

Cancels all open orders when `cond` is true.

`CancelAllExitOrders(cond)`

Cancels all open exit orders when `cond` is true.

`CancelDayOrders(cond)`

Cancels all open day orders when `cond` is true.

`CancelAllBuyOrders(cond)`

Cancels all open buy orders when `cond` is true.

`CancelAllSellOrders(cond)`

Cancels all open sell orders when `cond` is true.

`CancelOrdersByComment(cond, comment)`

Cancels all open orders with matching comment when `cond` is true.

Misc Formula Trading System Functions

The functions listed here works with the primary data series of the trading system.

OpenPositionSize

Returns the open position size. A positive value indicates a long position, 0 indicates no position, a negative value indicates a short position.

OpenPositionAbsSize

Returns the open position size in absolute value, i.e. OpenPositionAbsSize = abs(OpenPOsitionSize)

OpenPositionLong

Returns 1 or 0, depends on whether there is an open long position. The value 1 indicates there is an open long position, 0 otherwise.

OpenPositionShort

Returns 1 or 0, depends on whether there is an open short position. The value 1 indicates there is an open short position, 0 otherwise.

OpenPositionFlat

Returns 1 or 0, depends on whether the open position is flat (no open position). The value 1 indicates the position is flat, 0 otherwise.

OpenPositionAverageEntryPrice

Returns the average entry price of the open position.

OpenPositionBestPriceLevel

Returns the best price level of the open position has ever reached.

OpenPositionWorstPriceLevel

Returns the worst price level of the open position has ever reached.

OpenPositionPL

Returns the current profit/loss of the open position.

OpenPositionCost

Returns the average cost of the open position.

OpenPositionEntryPrice

Returns the first entry price of the open position.

OpenPositionEntryBar

Returns the bar number of the first entry of the open position.

OpenPositionEntrySize

Returns the size (e.g. number of shares/contracts) of the first entry of the open position.

OpenPositionEntryCommentIs (Comment)

Returns true (1) or false (0) based on the first entry comment of the open position matching the parameter Comment.

CurrentEquity

Returns the current equity of the trading system. The current equity equals the available cash plus the market value of the positions.

DefaultOrderSize

Returns the default order size for the system. Default order size is set up either in Symbol Info Manager for the symbol, or in **System** tab when applying the trading system.

MaxSizeAfterCoveringPositions

Returns maximum order size after current position is covered.

MaxSizeUsingAvailableCash

Returns maximum order size using available cash.

HasTradedThisBar

Returns 1 if a trade has been executed within this bar. 0 otherwise.

HasTradedToday

Returns 1 if a trade has been executed on current trading day. 0 otherwise.

PriceMultiple

Returns the price multiple of the underlying instrument.

MinTickSize

Returns the minimum tick size of the underlying instrument.

Broker Functions

Functions listed here are for querying position information from a live broker or Trade Simulator. Note that these functions are not the same as the "normal" position functions. Broker position functions returns what is going on in a real-life/simulated brokerage whereas the "normal" position functions are for handling the logic of a trading system.

For brokerage connection, see ***Connecting to Your Broker*** (on page 49). For Trade Simulator, see ***Trade Simulator*** (on page 755).

Time Charts, Pattern Scanner and Indicator Created with Formula

BrokerPosSize

Returns the position size from your broker. The symbol is that of the primary data series.

BrokerPosAvgEntry

Returns the average entry price of the position from your broker. The symbol is that of the primary data series.

Quote Window Formulas, Dynamic Grid, Cluster

See ***Position Information as Quote Fields and Formula Functions*** (on page 782).

Deployment

If you want to deploy your trading system for real-life trading, you should read the section ***Trading System Life Deployment Guide*** (see "Trading System Live Deployment Guide" on page 785).

Formula in Quote Windows

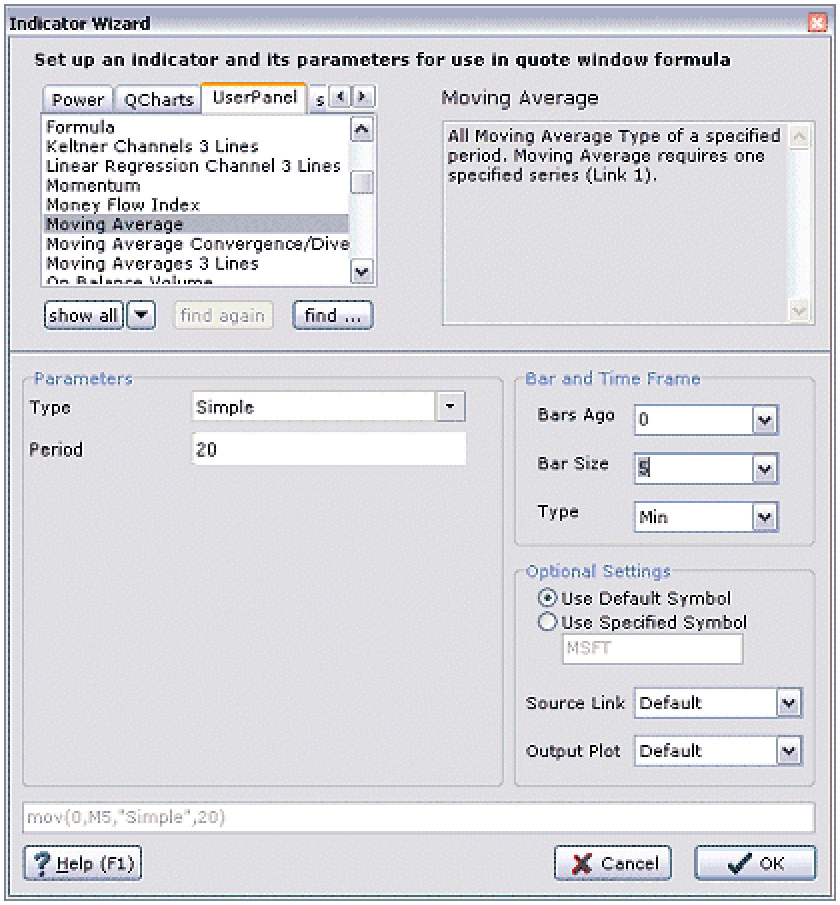
Tutorial: Using the Indicator Wizard in Quote Window Formula Column

Specifying indicators is a difficult task for many formula writers. There are different parameters and ways of specifying them, making it difficult to specify indicators correctly.

NeoTicker® provides an Indicator Wizard to help you specify indicators in formulas. You can simply fill in the parameters and options and the Indicator Wizard will specify the indicator for you.

To use the Indicator Wizard for a formula column, in the formula column properties (double click the formula column to launch the column properties), press the **Indicator Wizard** button.

The Indicator wizard will be opened.



Indicator Wizard

Set up an indicator and its parameters for use in quote window formula

Formula
Keltner Channels 3 Lines
Linear Regression Channel 3 Lines
Momentum
Money Flow Index
Moving Average
Moving Average Convergence/Dive
Moving Averages 3 Lines
On Balance Volume

Moving Average

All Moving Average Type of a specified period. Moving Average requires one specified series (Link 1).

Parameters

Type: Simple
Period: 20

Bar and Time Frame

Bars Ago: 0
Bar Size: 5
Type: Min

Optional Settings

● Use Default Symbol
○ Use Specified Symbol
MSFT

Source Link: Default
Output Plot: Default

mov(0,M5,"Simple",20)

Simply choose the indicator, fill in the parameters and the options. Notice that near the bottom of the Indicator Wizard, the indicator specification is previewed.

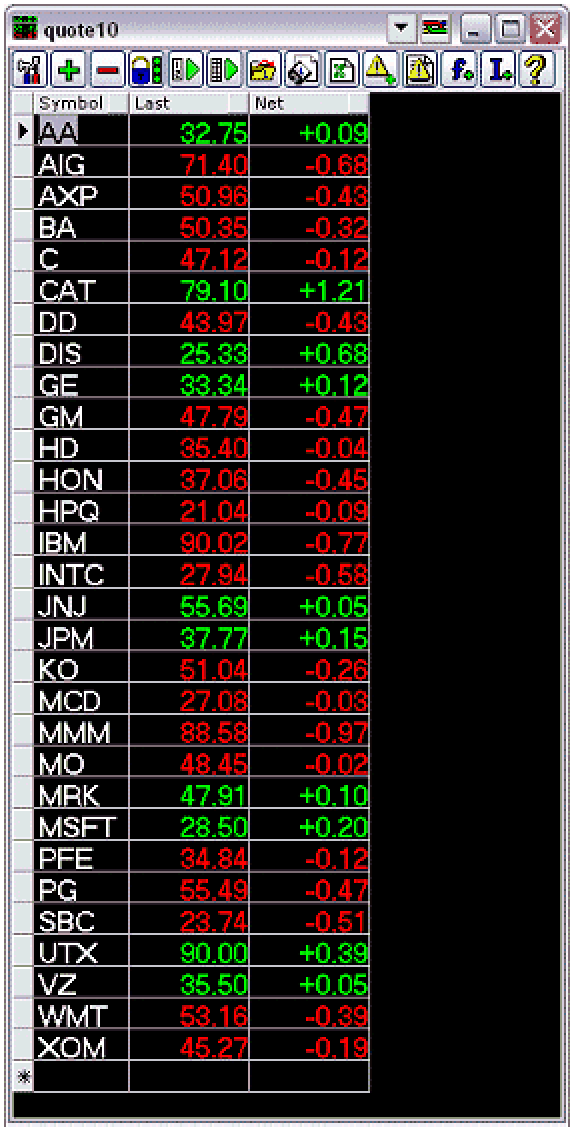
Press the **OK** button to accept the specification and the specification will be filled into the formula.

Tutorial: Advanced Formula in Quote Window

You may have seen some quote window formula examples in Beginner's Tutorials. This tutorial is a continuation that shows more advance quote window formula examples. This tutorial also provides more explanation on quote window formula.

In this tutorial we will setup two formula columns. The first formula column we want to show a solid green color when a symbol has gone past the 20 days highest high. In the second formula column we want to show a solid red color when a symbol has fell below the 20 days lowest low.

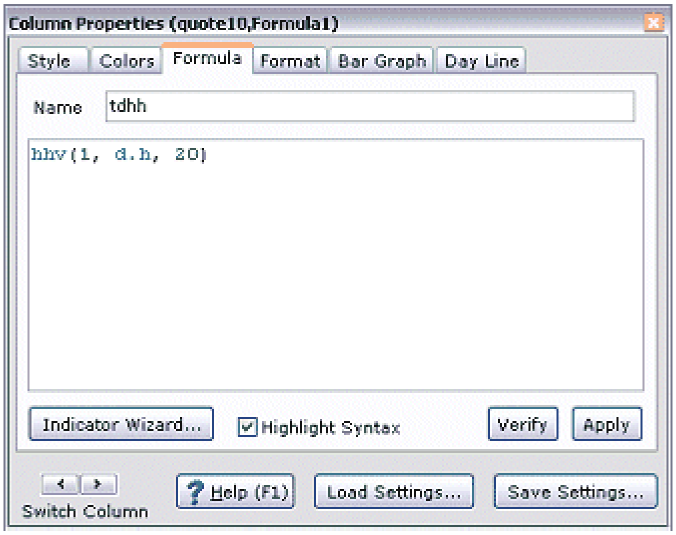
For this example we will use Dow Jones 30. Open a new quote window and import the Dow Jones 30 symbol list.



Symbol	Last	Net
AA	32.75	+0.09
AIG	71.40	-0.68
AXP	50.96	-0.43
BA	50.35	-0.32
C	47.12	-0.12
CAT	79.10	+1.21
DD	43.97	-0.43
DIS	25.33	+0.68
GE	33.34	+0.12
GM	47.79	-0.47
HD	35.40	-0.04
HON	37.06	-0.45
HPQ	21.04	-0.09
IBM	90.02	-0.77
INTC	27.94	-0.58
JNJ	55.69	+0.05
JPM	37.77	+0.15
KO	51.04	-0.26
MCD	27.08	-0.03
MMM	88.58	-0.97
MO	48.45	-0.02
MRK	47.91	+0.10
MSFT	28.50	+0.20
PFE	34.84	-0.12
PG	55.49	-0.47
SBC	23.74	-0.51
UTX	90.00	+0.39
VZ	35.50	+0.05
WMT	53.16	-0.39
XOM	45.27	-0.19

Add a formula column. Open the Column Properties window. Change the name to `tdhh` and in the formula area enter:

```
hhv(1, d.h, 20)
```



Name: tdhh

```
hhv(1, d.h, 20)
```

☑ Highlight Syntax

Press the **Apply** button to see the 20 days highest high calculate up to yesterday.

You can think of formula as a mathematical function. The components include name to identify the formula and parameters to change its behaviors.

Part	Explanation
hhv	is the name of a NeoTicker® built-in indicator. Indicator Quick Reference window lists all the available indicators. You can use the name under the "Function" column to include them in your formula
First Parameter (1)	is the periods ago result this indicator return. In this case a "1", which means the result is the value of 1 period ago
Second Parameter (d.h)	is the time frame and data series this indicator calculation base on. "d.h" means the time frame is daily and the calculation is base on the "High" price
Third Parameter (20)	is the number of period the Highest High Value base on. This parameter is specific to “Highest High Value” indicator.

You can use the Indicator wizard to help you specify indicators in formula. Simply press the **Indicator Wizard** button to launch the wizard.

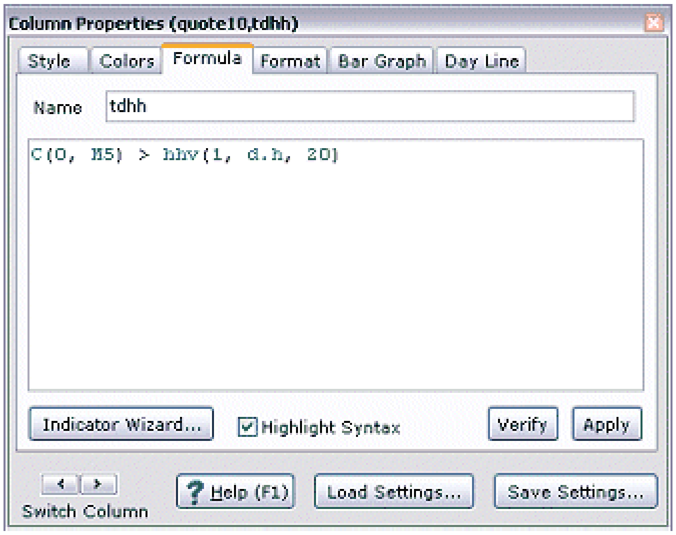
The number of parameters varies between indicators. The first two parameters are mandatory.

The next step is to setup a formula to determine if the closing price of current 5-minutes bar is greater than 20 days Highest High.

Double click on the formula to change the formula, add

```
C(0, M5) >
```

in front of `hhv( 1, d.h, 20)`.



Name tdhh

```
C(0, M5) > hhv(1, d.h, 20)
```

☑ Highlight Syntax

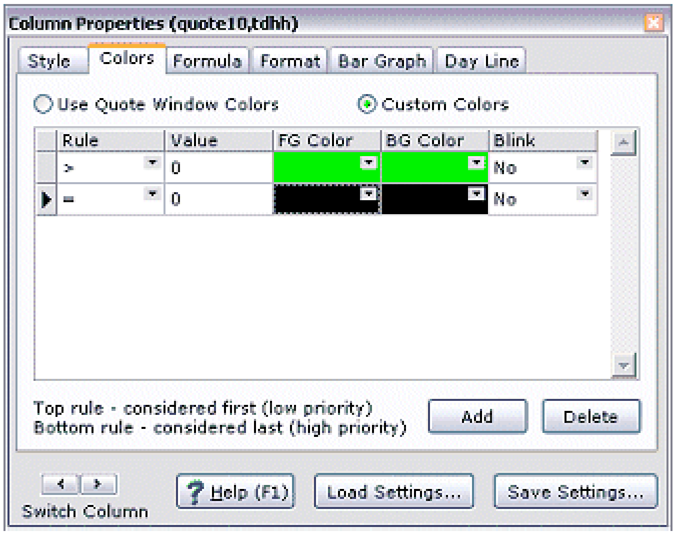
Press the **Apply** button. The value in the column will be either a 1 or 0. This is because the formula has become a comparison operation, so the result with return a 1 for true or a 0 for false.

This completes the condition evaluation.

Next step is to set the column to show green when 1 and black when 0.

Double click on the formula column to open the Column Properties window. Click on the **Colors** tab to set the column color, set the **FG color** to green, **BG color** to green, **Rule** to **>** and **Value** to 0. This will make the column become solid green when the condition `C(0, M5) > hhv(1, d.h, 20)` is true.

Press the **Add** button to add another color setting line. Set the **FG color** to black, **BG color** to black, **Rule** to **=** and **Value** to 0. The result color setting will look like the following:

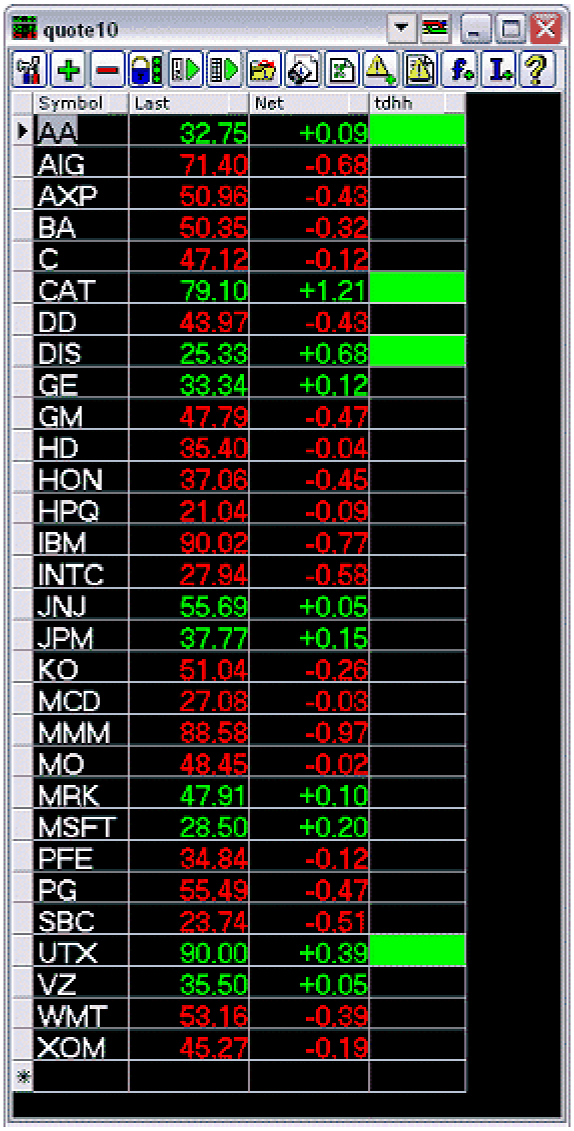


○ Use Quote Window Colors ● Custom Colors

Rule	Value	FG Color	BG Color	Blink
>	0	🟩	🟩	No
=	0	⬛	⬛	No

Top rule - considered first (low priority)
Bottom rule - considered last (high priority)

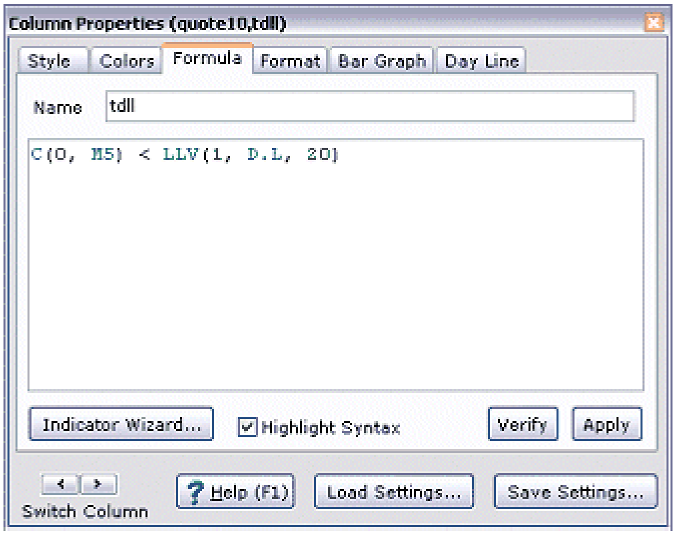
In the quote window, the green cell marks the close price is above 20 days highest high. It looks like:



Symbol	Last	Net	tdhh
AA	32.75	+0.09	🟩
AIG	71.40	-0.68	
AXP	50.96	-0.43	
BA	50.35	-0.32	
C	47.12	-0.12	
CAT	79.10	+1.21	🟩
DD	43.97	-0.43	
DIS	25.33	+0.68	🟩
GE	33.34	+0.12	
GM	47.79	-0.47	
HD	35.40	-0.04	
HON	37.06	-0.45	
HPQ	21.04	-0.09	
IBM	90.02	-0.77	
INTC	27.94	-0.58	
JNJ	55.69	+0.05	
JPM	37.77	+0.15	
KO	51.04	-0.26	
MCD	27.08	-0.03	
MMM	88.58	-0.97	
MO	48.45	-0.02	
MRK	47.91	+0.10	
MSFT	28.50	+0.20	
PFE	34.84	-0.12	
PG	55.49	-0.47	
SBC	23.74	-0.51	
UTX	90.00	+0.39	🟩
VZ	35.50	+0.05	
WMT	53.16	-0.39	
XOM	45.27	-0.19	

Now you can set up another column that show red when the last price is lower then 20 days lowest low.

You should set the column properties as follows:

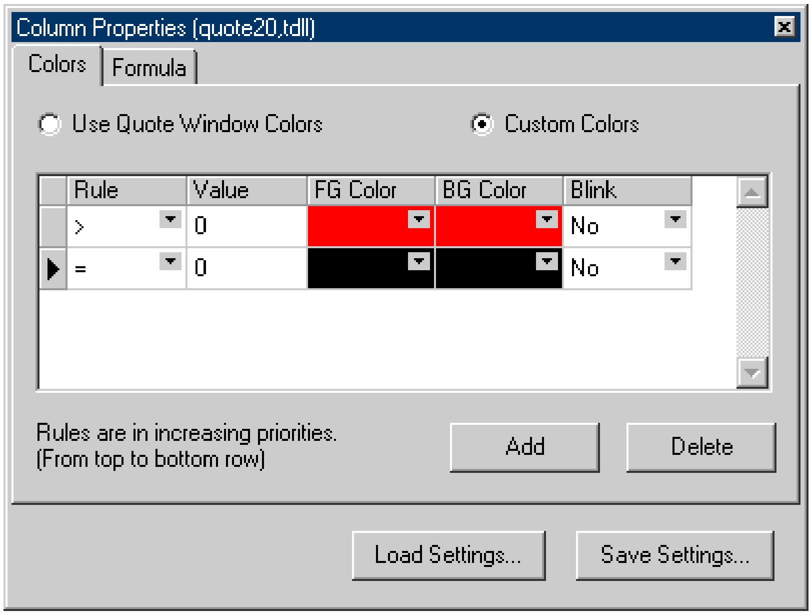


Column Properties (quote10,tdll)

Name: tdll

```
C(0, M5) < LLV(1, D.L, 20)
```

☑ Highlight Syntax



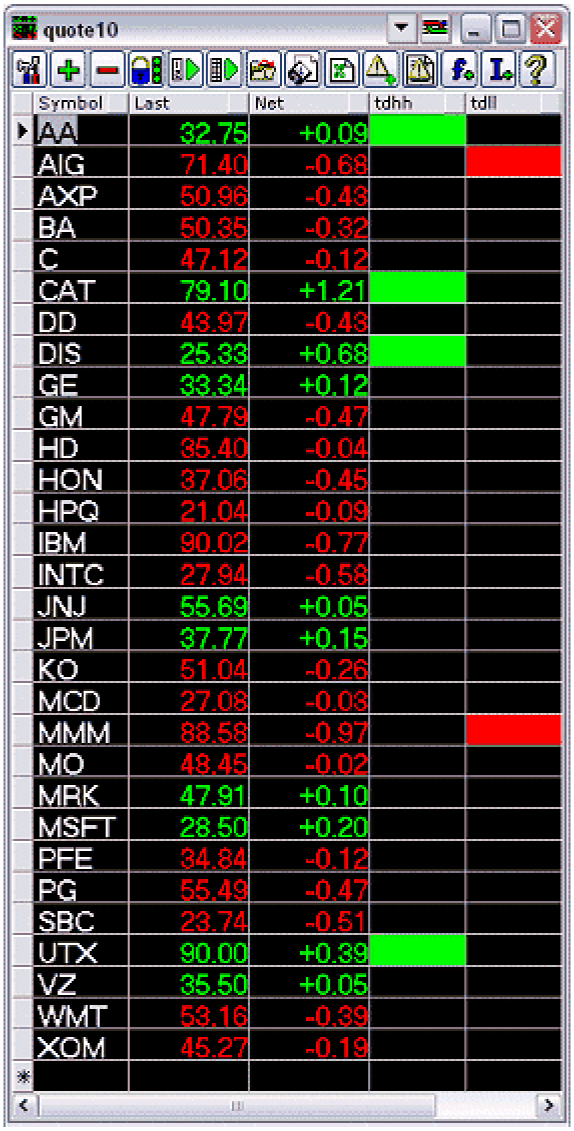
Column Properties (quote20,tdll)

○ Use Quote Window Colors ● Custom Colors

Rule	Value	FG Color	BG Color	Blink
>	0	🟥	🟥	No
=	0	⬛	⬛	No

Rules are in increasing priorities.
(From top to bottom row)

The resulting quote window should look like this:



Symbol	Last	Net	tdhh	tdll
AA	32.75	+0.09	🟩	
AIG	71.40	-0.68		🟥
AXP	50.96	-0.43		
BA	50.35	-0.32		
C	47.12	-0.12		
CAT	79.10	+1.21	🟩	
DD	43.97	-0.43		
DIS	25.33	+0.68	🟩	
GE	33.34	+0.12		
GM	47.79	-0.47		
HD	35.40	-0.04		
HON	37.06	-0.45		
HPQ	21.04	-0.09		
IBM	90.02	-0.77		
INTC	27.94	-0.58		
JNJ	55.69	+0.05		
JPM	37.77	+0.15		
KO	51.04	-0.26		
MCD	27.08	-0.03		
MMM	88.58	-0.97		🟥
MO	48.45	-0.02		
MRK	47.91	+0.10		
MSFT	28.50	+0.20		
PFE	34.84	-0.12		
PG	55.49	-0.47		
SBC	23.74	-0.51		
UTX	90.00	+0.39	🟩	
VZ	35.50	+0.05		
WMT	53.16	-0.39		
XOM	45.27	-0.19		

We will add another formula column to monitor two conditions: volume spike and the last price has passed the 20 days highest high or the 20 days lowest low. Color of the column will only change when one of the condition is true.

Add a new formula column. Double click on the column to edit the formula. Enter:

```
Vol(0, m5) > 1.5 * average (1, m5.v, 10)
```

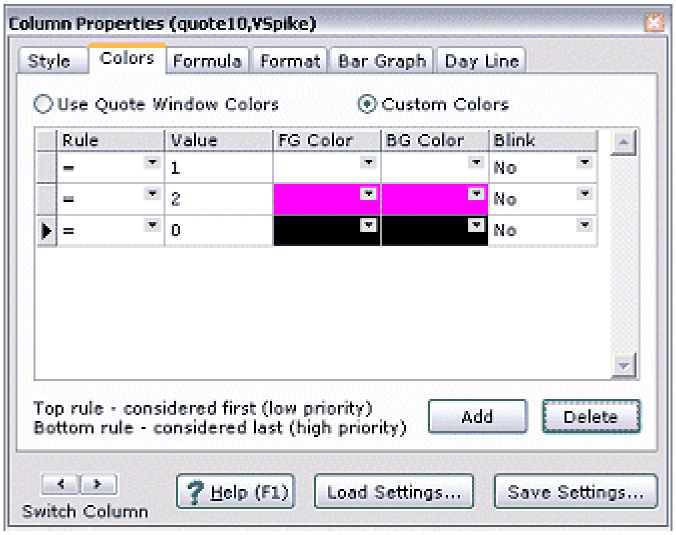
and change the **Name** to `VSpike`. This formula is based on the following criteria. When volume of the current 5-minutes bar is 1.5 times greater than average of the last 10 5-minutes bars, the formula will return 1, else 0. Press the **Apply** button you will see a bunch of 1s and 0s in the quote window.

Next we will add more evaluation to make the column display different colors. We want to set the column to white when volume spike and 5-minute close pass 20 days highest high. Set column color to magenta when volume spike and 5-minute close pass 20 days lowest low.

Since we already have 20 days highest high and 20 days lowest low calculated in another column, to save time we can reference those results. Double click on the formula column to change the formula to:

```
if(Vol( 0, m5) > 1.5*average (1, m5.v, 10), if(tdhh, 1,
if(tdll, 2, 0)), 0)
```

Next setup the color properties as follow:



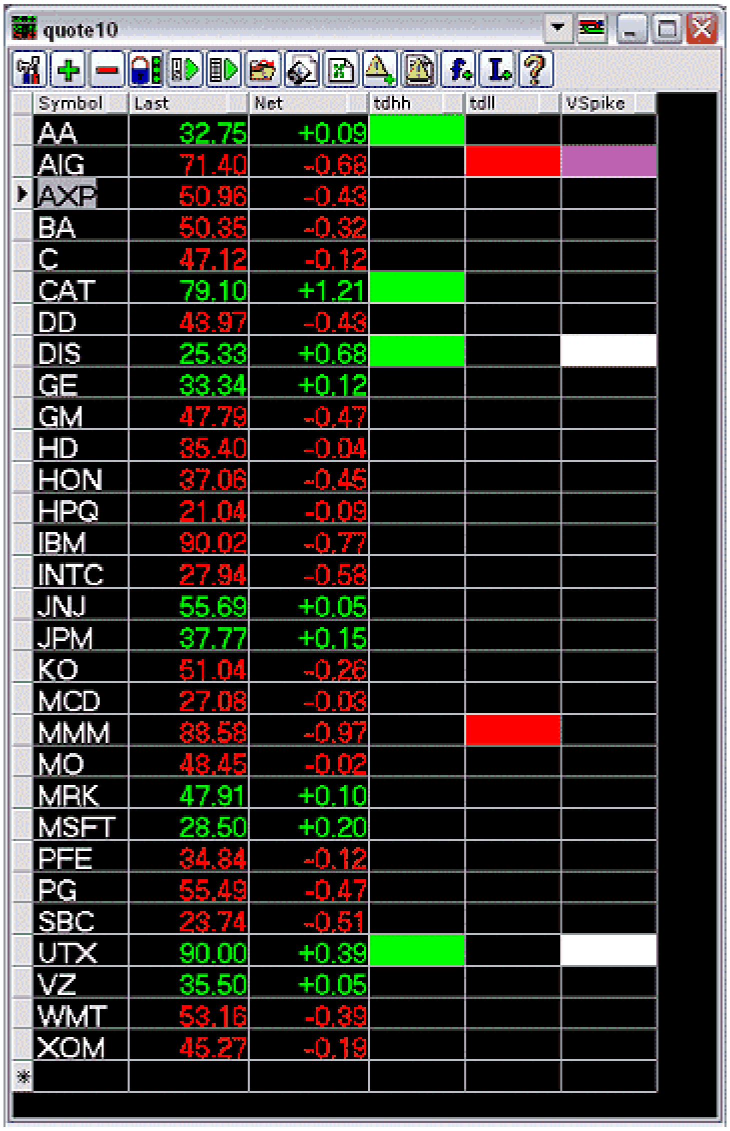
Column Properties (quote10,VSpike)

○ Use Quote Window Colors ● Custom Colors

Rule	Value	FG Color	BG Color	Blink
=	1			No
=	2	🟪	🟪	No
=	0	⬛	⬛	No

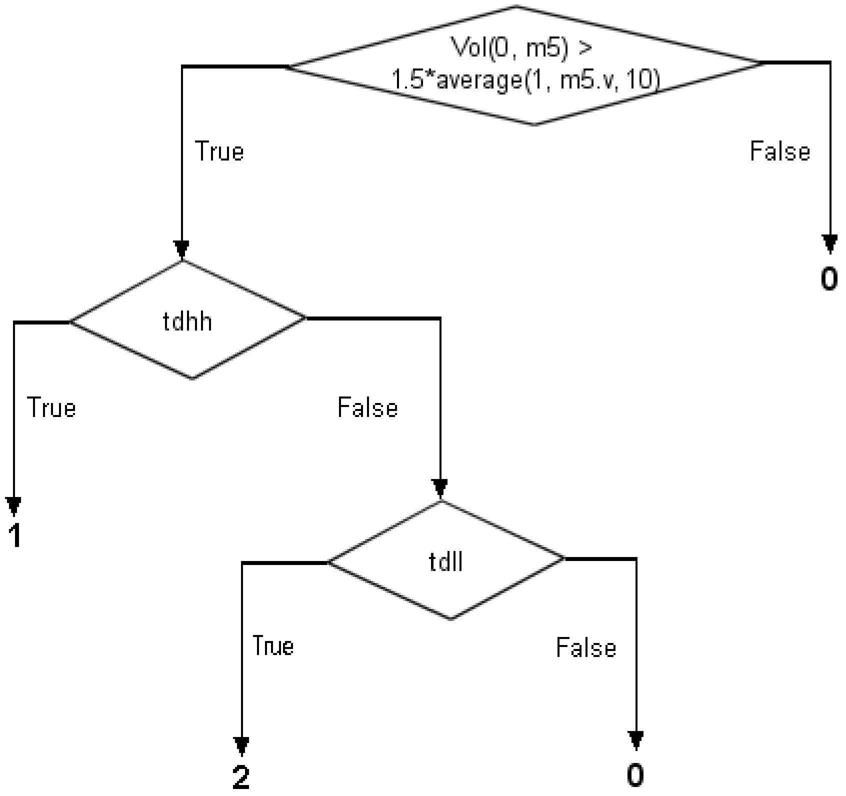
Top rule - considered first (low priority)
Bottom rule - considered last (high priority)

The quote window will look like:



Symbol	Last	Net	tdhh	tdll	VSpike
AA	32.75	+0.09	🟩		
AIG	71.40	-0.68		🟥	🟪
AXP	50.96	-0.43			
BA	50.35	-0.32			
C	47.12	-0.12			
CAT	79.10	+1.21	🟩		
DD	43.97	-0.43			
DIS	25.33	+0.68	🟩		⬜
GE	33.34	+0.12			
GM	47.79	-0.47			
HD	35.40	-0.04			
HON	37.06	-0.45			
HPQ	21.04	-0.09			
IBM	90.02	-0.77			
INTC	27.94	-0.58			
JNJ	55.69	+0.05			
JPM	37.77	+0.15			
KO	51.04	-0.26			
MCD	27.08	-0.03			
MMM	88.58	-0.97		🟥	
MO	48.45	-0.02			
MRK	47.91	+0.10			
MSFT	28.50	+0.20			
PFE	34.84	-0.12			
PG	55.49	-0.47			
SBC	23.74	-0.51			
UTX	90.00	+0.39	🟩		⬜
VZ	35.50	+0.05			
WMT	53.16	-0.39			
XOM	45.27	-0.19			

To understand how the formula works, see the following flow chart breakdown of the formula:



Flow Chart with Formula Logic

Tutorial: RSI in Quote Window Formula

This tutorial will show you how to add an RSI indicator to a quote window. It illustrates the differences between indicator calling in quote window formula and in chart formula. The differences are due to the data available to the formula.

In time chart, the input of the formula can come from different data and indicators. You must specify the data source to the formula, and the time frame of the formula is dictated by the data source. For example, if you specify the input to be 1-minute MSFT, then the formula will have a 1-minute time frame.

In quote window, the input of the formula must be the symbol in the row or an optional symbol. So you do not need to supply a data source. On the other hand, the time frame is unclear. Therefore, you must specify the time frame in a quote window formula.

The good thing is, while you should understand the differences between the two, there is no need for you to memorize the detail syntax as Indicator Wizard will take care of that for you.

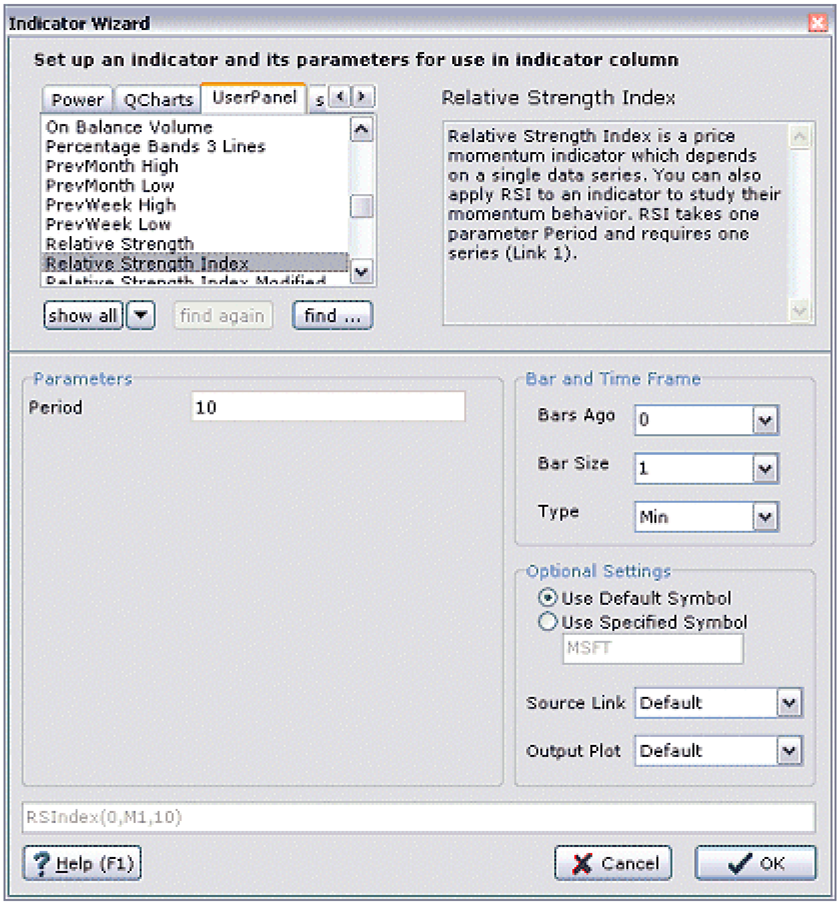
Let's illustrate the differences with a concrete example. The following formula evaluate RSI in a chart:

```
RSIndex(0,data1,10)
```

(The usage of the above formula in a chart can be found in ***Tutorial: Using Multi-Statement Formula to Calculate Indicator on Indicator*** (on page 356))

We will show you how to create an equivalent formula in quote window. First, create a quote window and insert some symbols to it. Press the add indicator column button. This button is a short cut action that creates a formula column and opens the indicator wizard to let you specify indicator. Later if you want to edit this column, you will edit it just as if you are editing a formula column.

The indicator wizard will be open. Choose **RSI**, and notice that on the right hand side there are options to set time frame. Make this a 1-minute RSI. Indicator Wizard should look like:



Relative Strength Index

Relative Strength Index is a price momentum indicator which depends on a single data series. You can also apply RSI to an indicator to study their momentum behavior. RSI takes one parameter Period and requires one series (Link 1).

Period: 10

Bars Ago: 0

Bar Size: 1

Type: Min

● Use Default Symbol

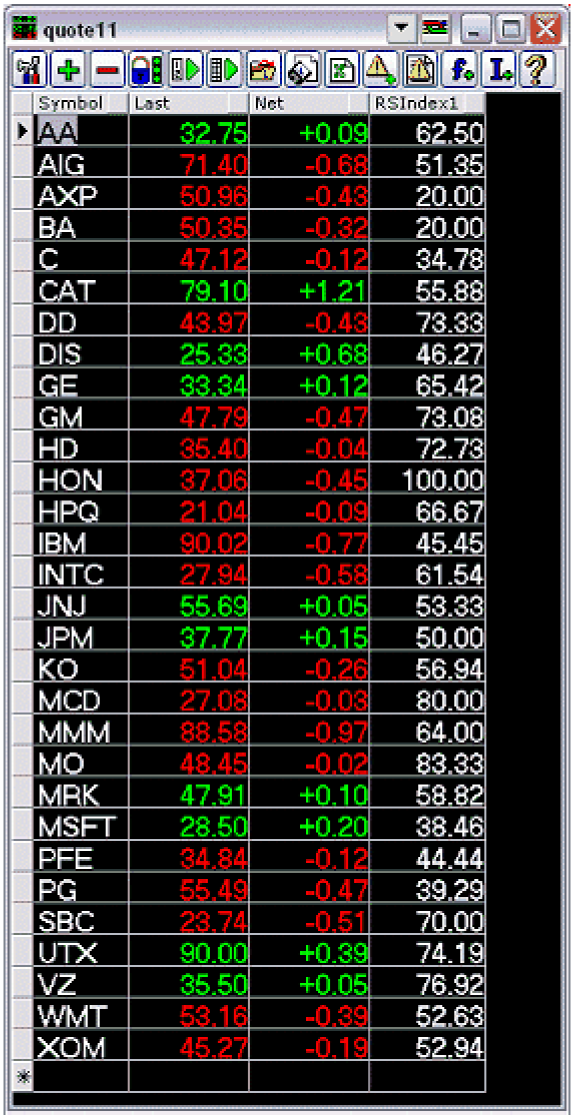
○ Use Specified Symbol: MSFT

Source Link: Default

Output Plot: Default

RSIndex(0,M1,10)

Press the **OK** button. An indicator column that calculates RSI will be created.



Symbol	Last	Net	RSIndex1
AA	32.75	+0.09	62.50
AIG	71.40	-0.68	51.35
AXP	50.96	-0.43	20.00
BA	50.35	-0.32	20.00
C	47.12	-0.12	34.78
CAT	79.10	+1.21	55.88
DD	43.97	-0.43	73.33
DIS	25.33	+0.68	46.27
GE	33.34	+0.12	65.42
GM	47.79	-0.47	73.08
HD	35.40	-0.04	72.73
HON	37.06	-0.45	100.00
HPQ	21.04	-0.09	66.67
IBM	90.02	-0.77	45.45
INTC	27.94	-0.58	61.54
JNJ	55.69	+0.05	53.33
JPM	37.77	+0.15	50.00
KO	51.04	-0.26	56.94
MCD	27.08	-0.03	80.00
MMM	88.58	-0.97	64.00
MO	48.45	-0.02	83.33
MRK	47.91	+0.10	58.82
MSFT	28.50	+0.20	38.46
PFE	34.84	-0.12	44.44
PG	55.49	-0.47	39.29
SBC	23.74	-0.51	70.00
UTX	90.00	+0.39	74.19
VZ	35.50	+0.05	76.92
WMT	53.16	-0.39	52.63
XOM	45.27	-0.19	52.94

If you open Column Properties, you can check the formula for RSI. Notethat RSI is called differently from the way RSI is called in time chart formula. In the place of the source data (`data1`), you now has a time frame information (`M1`, which means 1 minute)

Time chart version:

```
RSIndex(0, data1, 10)
```

Quote window version:

```
RSIndex(0, M1, 10)
```

Tutorial: Catching Errors in Quote Window Formula

Quote window can help you catch errors in formula. When there is an error in a quote window formula, quote window will report the error to you in a short message. Quote window also provides facility to help you further investigate the error.

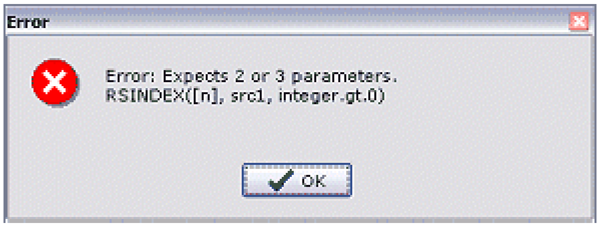
In this example, we will deliberately create an error in a quote window formula to see how error reporting works. First, create a quote window and add some symbols to it. Then create a formula and enter the following formula:

```
fastk(0,M1,5) + RSIndex(0,MA,10,20)
```

This formula has 2 errors:

- There is one more parameter than what `RSIndex` can take
- MA is a bad time frame

If you press **Apply** button, quote window will catch one of the error right away:



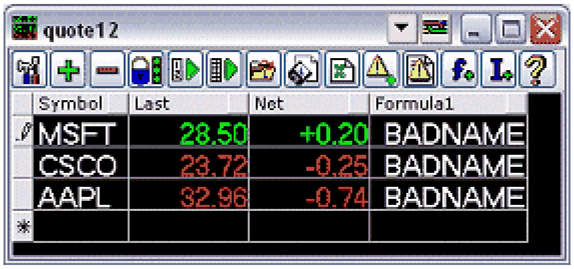
Error: Expects 2 or 3 parameters.
RSINDEX([n], src1, integer.gt.0)

If we change the formula to:

```
fastk(0,M1,5) + RSIndex(0,MA,10)
```

When you press the **Apply** button, quote window will accept this formula. This because while MA is not a valid time frame, it can only be caught in run-time.

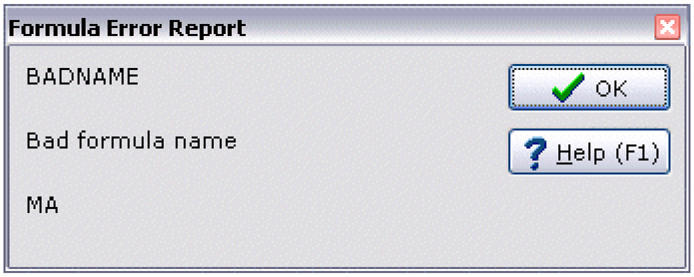
The quote window will show something like:



Symbol	Last	Net	Formula1
MSFT	28.50	+0.20	BADNAME
CSCO	23.72	-0.25	BADNAME
AAPL	32.96	-0.74	BADNAME

Because there is an error in the formula, instead of returning a value, the formula returns an error message. Quote window can give you a more detail error report to help you spot the error. Right click on one of `BADNAME` cell to open the pop up menu, and choose **Error Report**.

Quote window will open an error report window. It makes the best attempt to locate the error in the formula for you:



Formula Error Report

BADNAME

Bad formula name

MA

The error report has identify the error is due to MA being a bad name, and you can correct the formula to:

```
fastk(0,M1,5) + RSIndex(0,M1,10)
```

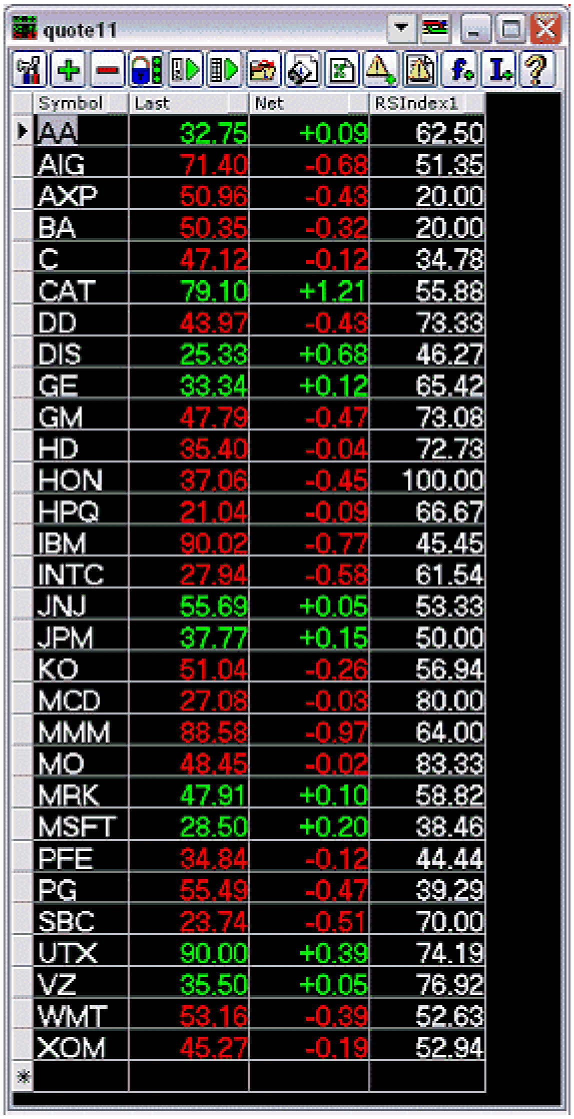
Tutorial: Writing a Three-Answer Evaluation Formula

It is simple to evaluate to true (value 1) or false (value 0) in a formula, but what if you want to evaluate to three answers? For example, if the RSI is between 0-50, returns 1; between 51-75, returns 2; between 76-100, returns 3. This is especially useful if you want to use the value for color rule assignment.

You can use a nested if statement, but the solution is difficult to read and generalize to more than 3 answers.

This tutorial will show you the technique to write such formula. It is relatively easy to generalize the techniques for another multiple answers scenario.

First, create a quote window and add an RSI indicator to it. The column name is `RSIndex1`.



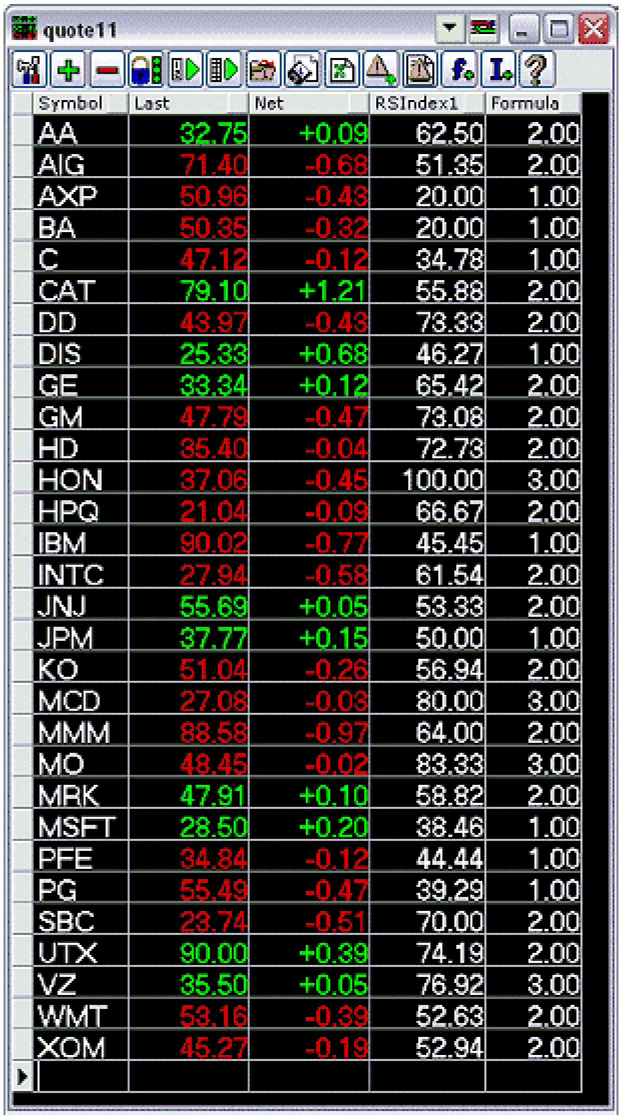
Symbol	Last	Net	RSIndex1
AA	32.75	+0.09	62.50
AIG	71.40	-0.68	51.35
AXP	50.96	-0.43	20.00
BA	50.35	-0.32	20.00
C	47.12	-0.12	34.78
CAT	79.10	+1.21	55.88
DD	43.97	-0.43	73.33
DIS	25.33	+0.68	46.27
GE	33.34	+0.12	65.42
GM	47.79	-0.47	73.08
HD	35.40	-0.04	72.73
HON	37.06	-0.45	100.00
HPQ	21.04	-0.09	66.67
IBM	90.02	-0.77	45.45
INTC	27.94	-0.58	61.54
JNJ	55.69	+0.05	53.33
JPM	37.77	+0.15	50.00
KO	51.04	-0.26	56.94
MCD	27.08	-0.03	80.00
MMM	88.58	-0.97	64.00
MO	48.45	-0.02	83.33
MRK	47.91	+0.10	58.82
MSFT	28.50	+0.20	38.46
PFE	34.84	-0.12	44.44
PG	55.49	-0.47	39.29
SBC	23.74	-0.51	70.00
UTX	90.00	+0.39	74.19
VZ	35.50	+0.05	76.92
WMT	53.16	-0.39	52.63
XOM	45.27	-0.19	52.94

Create a formula column, enter the following formula:

```
choose( (RSIndex1 >= 0) and (RSIndex1 <= 50), 1,
        (RSIndex1 >= 51) and (RSIndex1 <= 75), 2,
        (RSIndex1 >= 76) and (RSIndex1 <= 100), 3,
        -1)
```

The `choose` function returns a value for each matching condition, with the last value being the default if no condition matches. So in the formula above:

- if RSI value is between 0 and 50, the formula returns 1
- if RSI value is between 51 and 75, the formula returns 2
- if RSI value is between 76 and 100, the formula returns 3
- if RSI value is outside of 0 to 100, the formula returns -1



Symbol	Last	Net	RSIndex1	Formula
AA	32.75	+0.09	62.50	2.00
AIG	71.40	-0.68	51.35	2.00
AXP	50.96	-0.43	20.00	1.00
BA	50.35	-0.32	20.00	1.00
C	47.12	-0.12	34.78	1.00
CAT	79.10	+1.21	55.88	2.00
DD	43.97	-0.43	73.33	2.00
DIS	25.33	+0.68	46.27	1.00
GE	33.34	+0.12	65.42	2.00
GM	47.79	-0.47	73.08	2.00
HD	35.40	-0.04	72.73	2.00
HON	37.06	-0.45	100.00	3.00
HPQ	21.04	-0.09	66.67	2.00
IBM	90.02	-0.77	45.45	1.00
INTC	27.94	-0.58	61.54	2.00
JNJ	55.69	+0.05	53.33	2.00
JPM	37.77	+0.15	50.00	1.00
KO	51.04	-0.26	56.94	2.00
MCD	27.08	-0.03	80.00	3.00
MMM	88.58	-0.97	64.00	2.00
MO	48.45	-0.02	83.33	3.00
MRK	47.91	+0.10	58.82	2.00
MSFT	28.50	+0.20	38.46	1.00
PFE	34.84	-0.12	44.44	1.00
PG	55.49	-0.47	39.29	1.00
SBC	23.74	-0.51	70.00	2.00
UTX	90.00	+0.39	74.19	2.00
VZ	35.50	+0.05	76.92	3.00
WMT	53.16	-0.39	52.63	2.00
XOM	45.27	-0.19	52.94	2.00

The choose function returns the value of the first matching condition. With this knowledge, we can simplifies the above formula to:

```
choose( (RSIndex1 >= 76) and (RSIndex1 <= 100), 3,
        RSIndex1 >= 51, 2,
        RSIndex1 >= 0, 1,
        -1)
```

Formula in Charts

Tutorial: Using the Indicator Wizard in Indicators

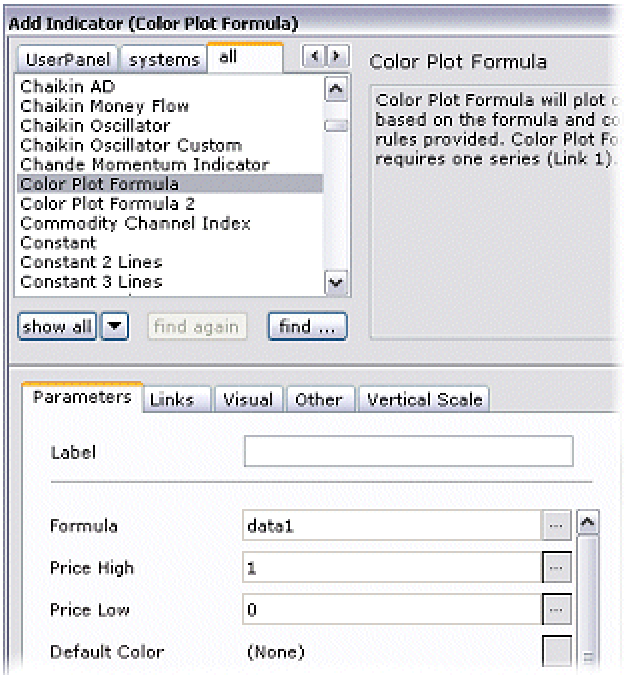
Specifying indicators is a difficult task for many formula writers. There are different parameters and ways of specifying them, making it difficult to specify indicators correctly. NeoTicker® provides an Indicator Wizard to help you specify indicators in formulas. You can simply fill in the parameters and options and Indicator Wizard will specify the indicator for you.

Many indicators accepts formulas as parameter input. For example in the color plot indicator, you can specify a formula for the price high and price low values.

In this example, we will see how to use Indicator Wizard to specify input for an indicator that accepts formula as parameter.

Open a time chart. Add a data series to the time chart. In this example, we use a e-mini futures contract.

Add a Color Plot Formula indicator to the e-mini contract:

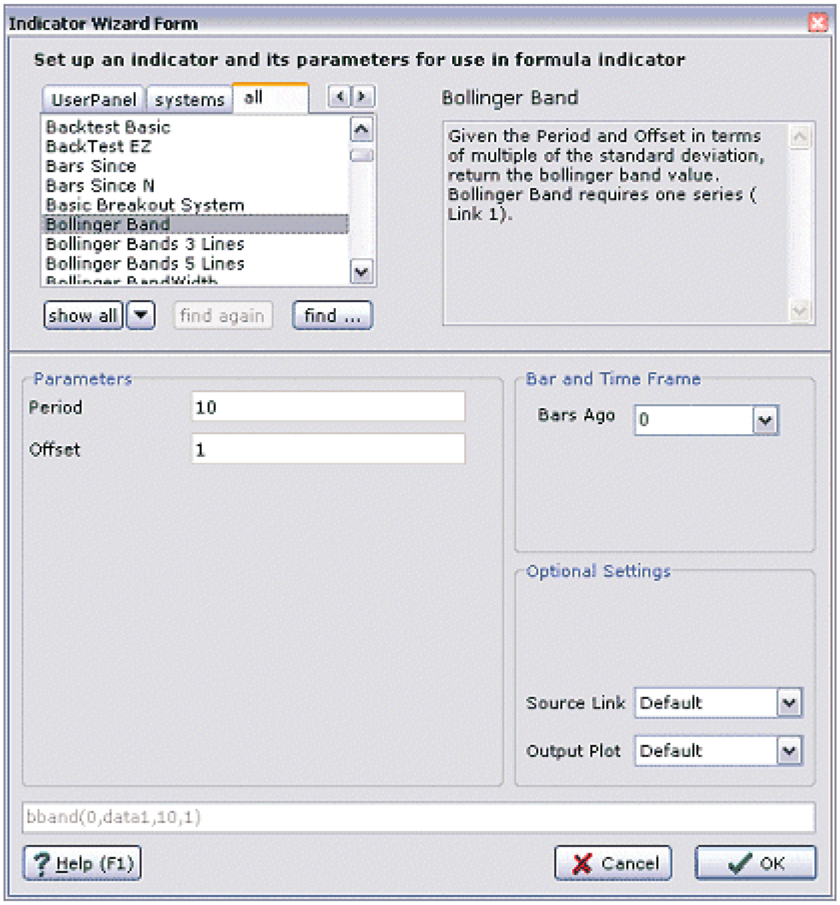


Color Plot Formula will plot c
based on the formula and co
rules provided. Color Plot Fo
requires one series (Link 1).

Label	
Formula	data1
Price High	1
Price Low	0
Default Color	(None)

Press the … button besides the **Price High** parameter. This will let you edit the formula for the **Price High** parameter. Clear the original value of **Price High** in the editor, then press the **Indicator Wizard** button in the editor.

In Indicator Wizard, choose Bollinger Band and specify the **Period** to 10 and **Offset** to 1. Press the **OK** button to accept the change.



Set up an indicator and its parameters for use in formula indicator

Bollinger Band

Given the Period and Offset in terms of multiple of the standard deviation, return the bollinger band value. Bollinger Band requires one series (Link 1).

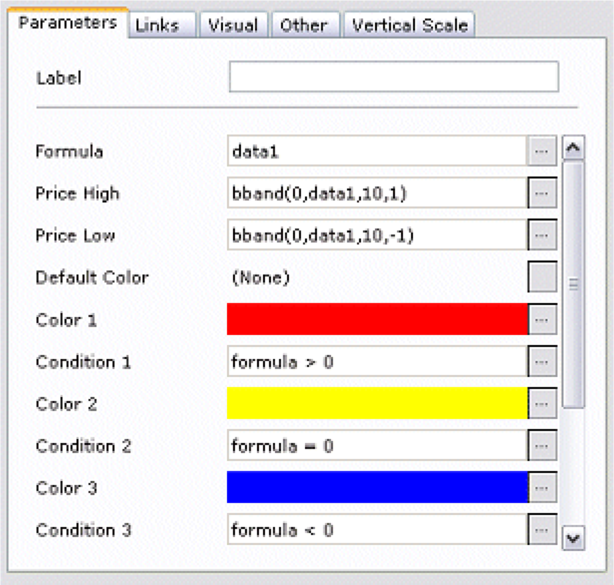
Period	10
Offset	1
Bars Ago	0
Source Link	Default
Output Plot	Default

```
bband(0,data1,10,1)
```

Notice that the formula for **Price High** has been changed to the Bollinger Band indicator.

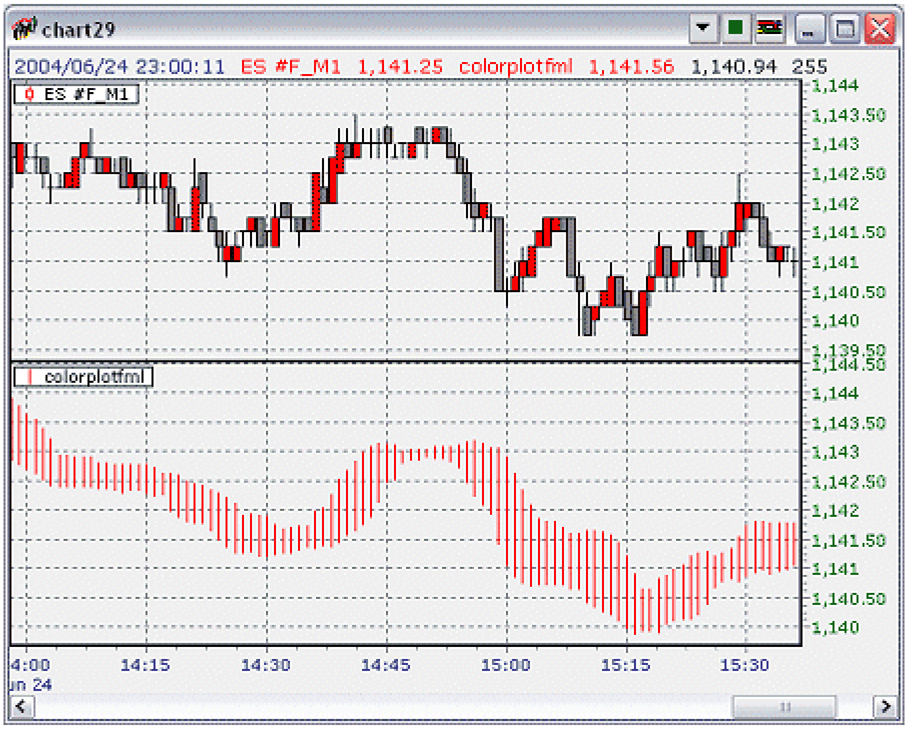
Add a similar Bollinger Band to the **Price Low** parameter. This time, with a **Period** of 10 and **Offset** of -1.

The color plot indicator will look like:



Label	
Formula	data1
Price High	bband(0,data1,10,1)
Price Low	bband(0,data1,10,-1)
Default Color	(None)
Color 1	🟥
Condition 1	formula > 0
Color 2	🟨
Condition 2	formula = 0
Color 3	🟦
Condition 3	formula < 0

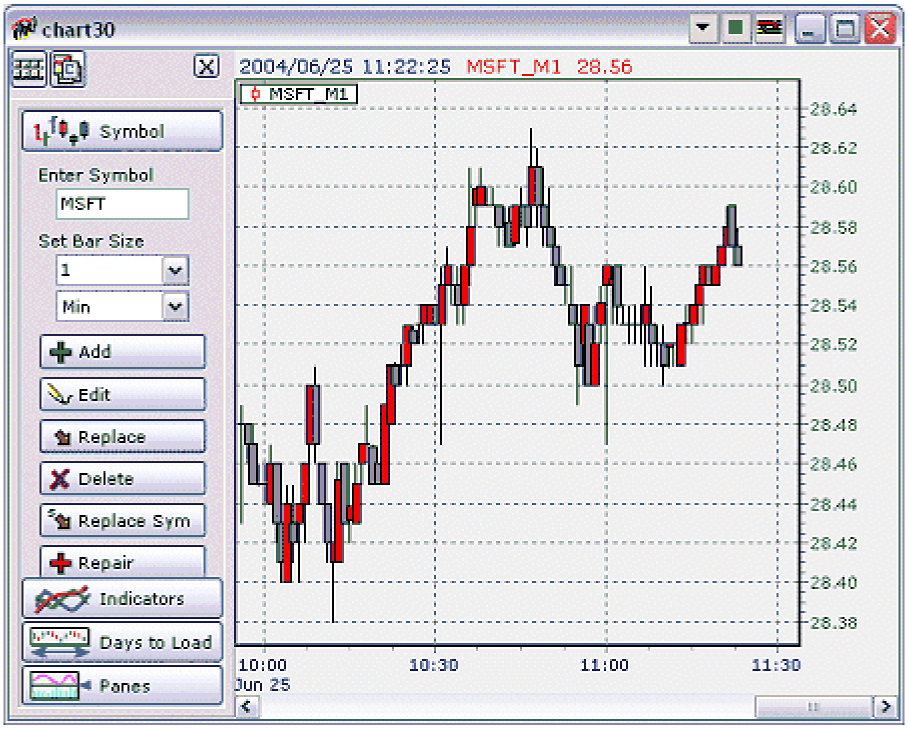
Press the **Apply** button to change the indicator. You have just created a color plot formula using Bollinger Band as high and low.



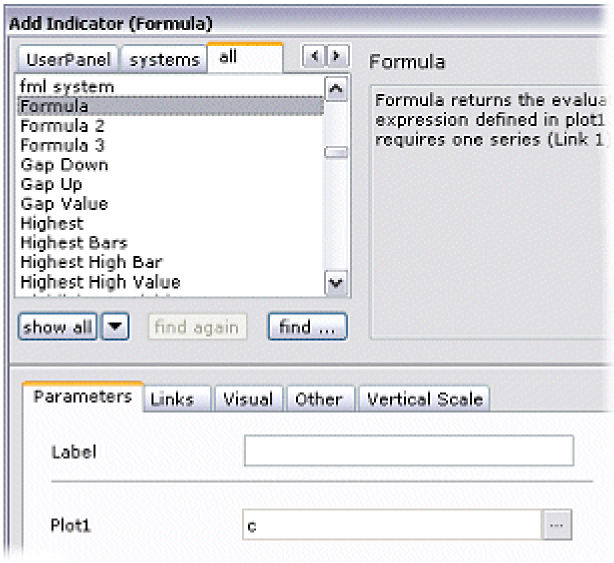
Tutorial: Using Multi-Statement Formula to Calculate Indicator on Indicator

In this example, we will use formula language to do indicator on indicator calculation in time chart. Specifically, we will apply an exponential moving average on RSI. Multi-statement formula will be used to illustrate the concept of assignment in formula language.

First, create a time chart. Add a symbol to it. This example uses MSFT.



Click on the data series to select it, and choose **Add Indicator** from the pop up menu to open the Add Indicator dialog. In the dialog, press the **all** tab. Choose from the list the indicator **Formula**.



Add Indicator (Formula)

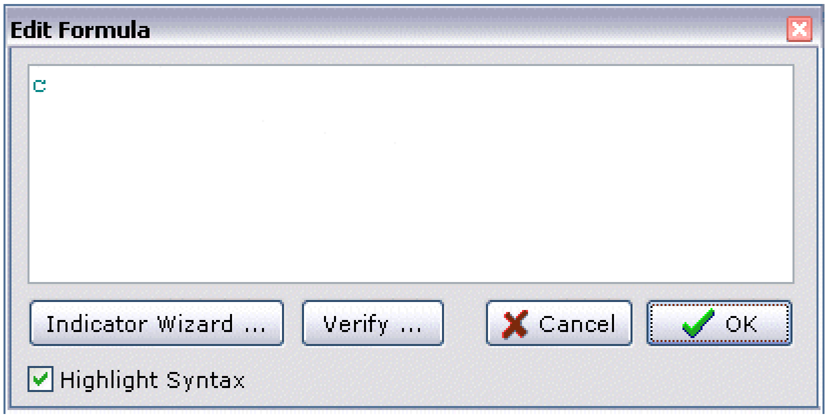
Formula

Formula returns the evalua
expression defined in plot1
requires one series (Link 1

Label:

Plot1: c

The formula indicator lets you use a formula for the indicator plot. It should show c in the parameter for **Plot1**. Press the ... button besides the parameter to edit the formula:



Edit Formula

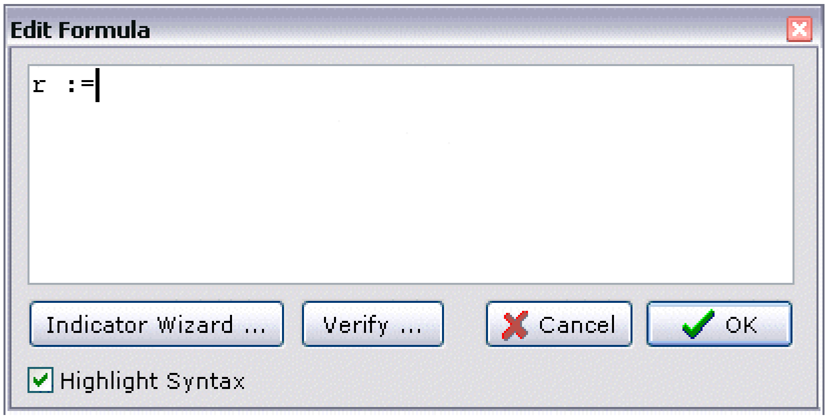
```
c
```

☑ Highlight Syntax

We will create a two-statement formula. Recall that we want to apply exponential moving average on RSI. The first statement in the formula will calculate the RSI and the second statement in the formula will apply an exponential moving average.

Currently, the edit formula currently has a formula in it that we do not want (i.e. the default `c` formula). Press and hold the left mouse button and drag to high light the formula. Press the DELETE key on the keyboard. At this stage, the edit area should be clear.

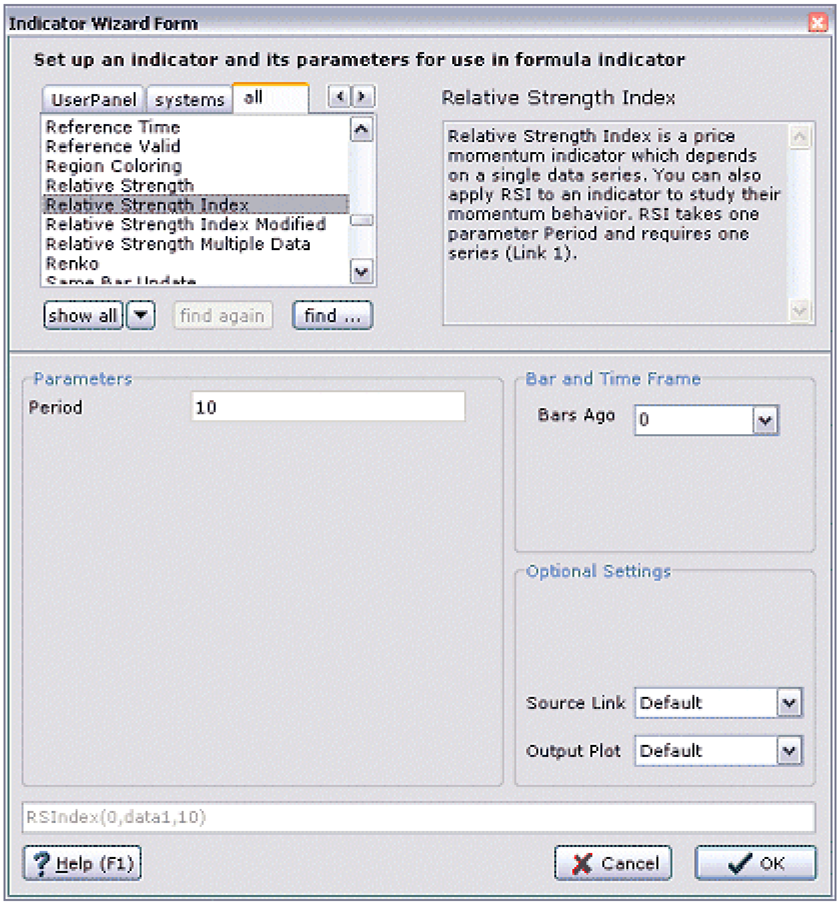
Type `r :=` into the edit area. At this stage, the edit area should look like:



```
r :=
```

Rather than typing in the RSI indicator, we will use the Indicator Wizard to fill in the indicator for us. Press the **Indicator Wizard** button.

The Indicator Wizard is opened. Choose **Relative Strength Index** under the **all** tab, and enter your favorite period for RSI.



Relative Strength Index is a price momentum indicator which depends on a single data series. You can also apply RSI to an indicator to study their momentum behavior. RSI takes one parameter Period and requires one series (Link 1).

Period: 10

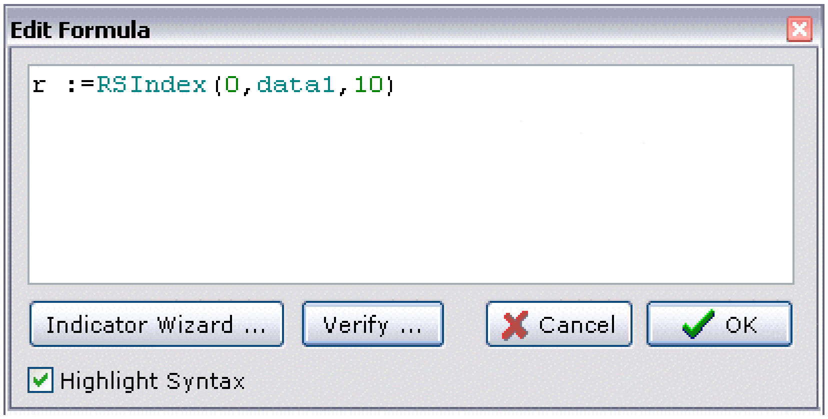
Bars Ago: 0

Source Link: Default

Output Plot: Default

RSIndex(0,data1,10)

Press the **OK** button. Note that the RSI indicator will be added after `r :=` .



```
r :=RSIndex(0,data1,10)
```

☑ Highlight Syntax

This is a statement in the formula language. A statement performs certain operations. What this statement does is to calculate the RSI indicator, and assigns the result of RSI to the variable r.

The symbol := is the assignment operator that relates the variable `r` and the calculation `RSIndex(0,data1,10)`.

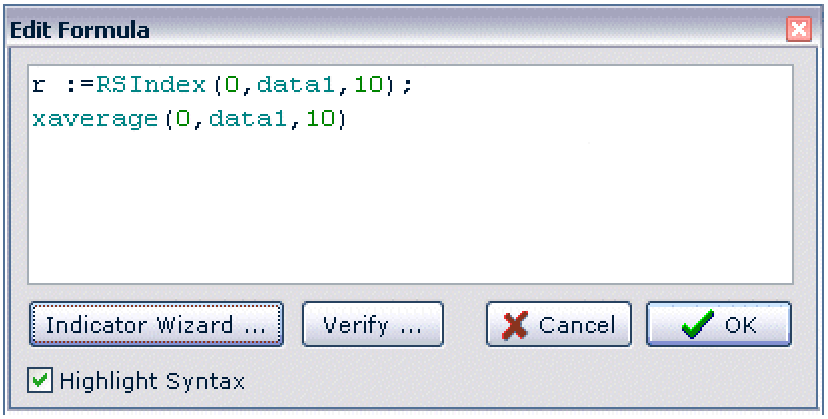
By assigning the result of the RSI calculation to the variable `r`, you can use `r` in other statements.

Note that the RSI indicator is calculating the RSI value on `data1`, the linked data series to the Formula indicator.

Statements in formula language must be separated by a semicolon. So before you enter the next statement, type a semicolon after the statement.

After you add the semicolon, you can optionally press the ENTER key to add a new line in the edit area. Press the **Indicator Wizard** button again. This time, choose the exponential moving average.

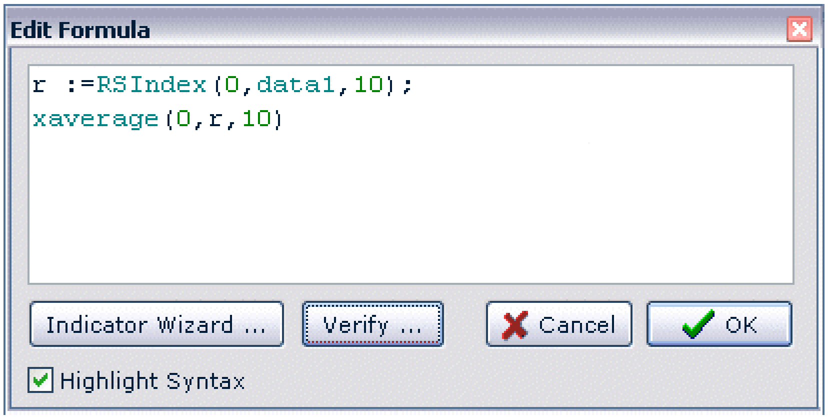
Press the **OK** button. The exponential moving average appears as a statement in the formula.



```
r :=RSIndex(0,data1,10);
xaverage(0,data1,10)
```

The last statement of a formula is for calculating the answer of the formula. So you do not need to perform any assignment.

The last thing to change is to change the input source of the exponential moving average from `data1` to `r`. This will tell the moving average to calculate the moving average using the variable `r`. Recall that `r` stores the RSI calculation result. In effect, you will be applying moving average on RSI.

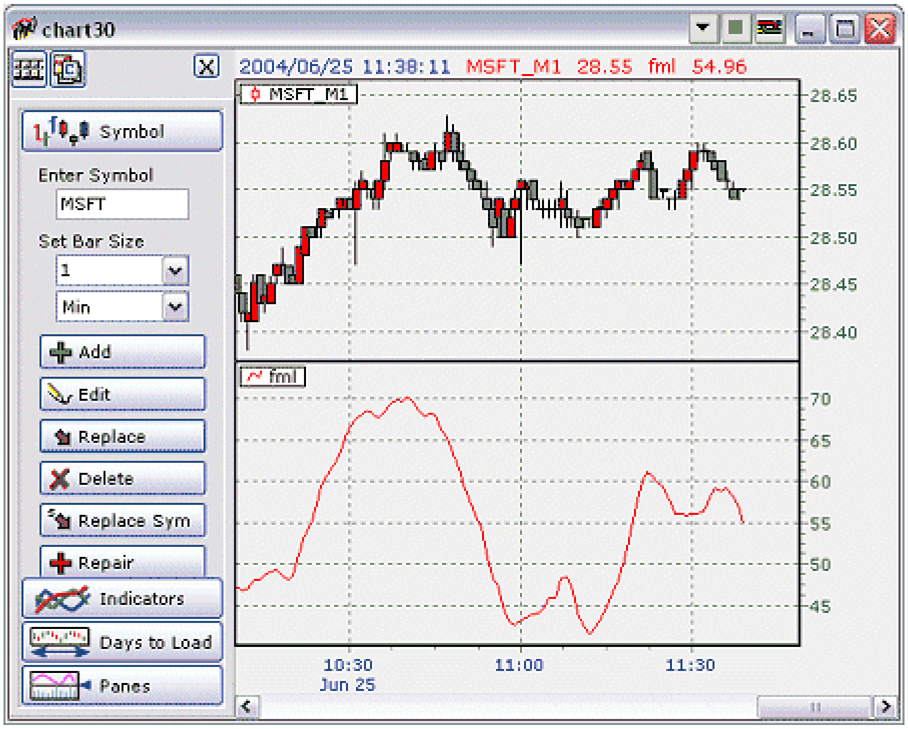


```
r :=RSIndex(0,data1,10);
xaverage(0,r,10)
```

☑ Highlight Syntax

Press the **OK** button in the editor to accept the formula change. Then in the Add Indicator dialog, press the **Apply** button in the indicator setup window to add the indicator to the chart.

The chart will look like:



Tutorial: Quickest Way to Plot Your Custom Expression Using the Formula Indicator

To quickly create a custom plot on a chart is very easy. Lets go through the following example to illustrate the steps required along with some useful tips.

First create a new chart using the symbol DIS, daily data for 250 days.

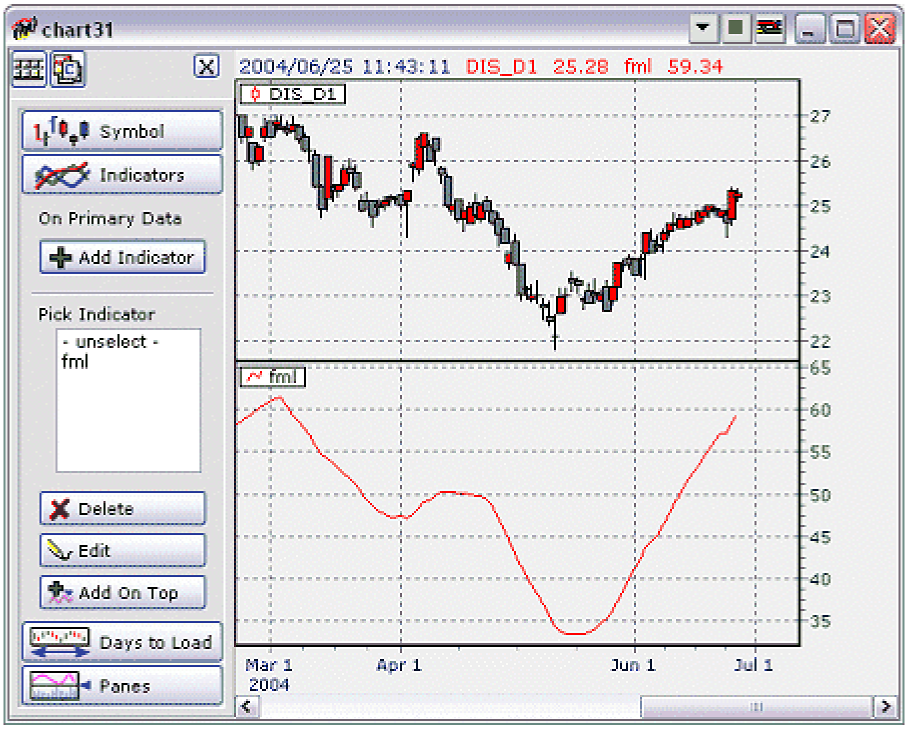
Say you are interested to plot an exponential moving average on top of RSI.

If you just happen to have the RSI already added to your chart, then all you have to do is add an exponential moving average with the Link set to the indicator RSI in the chart. Then you will have the moving average you want.

On the other hand, if you are interested in twisting the combination all together, then you can add the indicator **Formula** to the chart and set its parameter **Plot1** to the following expression,

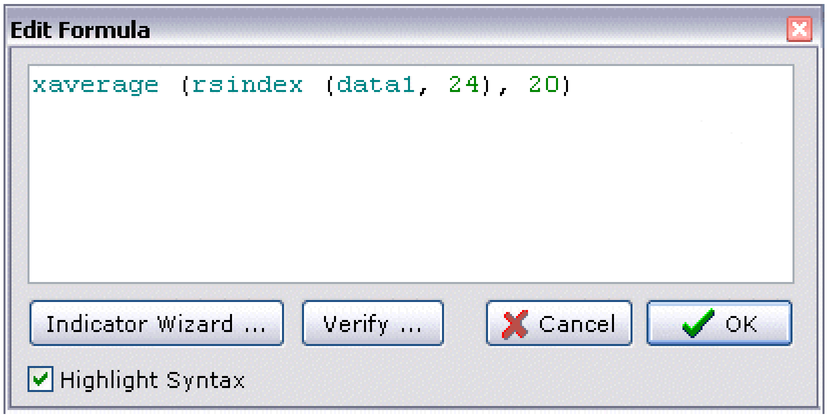
```
xaverage (rsindex (data1, 14), 20)
```

The formula indicator will now plot the expression you just typed in.



Now you are interested in changing the formula from using 14-period to say 24-period for the RSI, all you need to do is to double-click on the **fml** legend in the chart to open the indicator setup window.

Now you can modify the formula in place directly within the Indicator Setup Window. Or, if you find the edit area too small, you can open a formula editor by pressing the ... button next to the **Plot1** edit area.

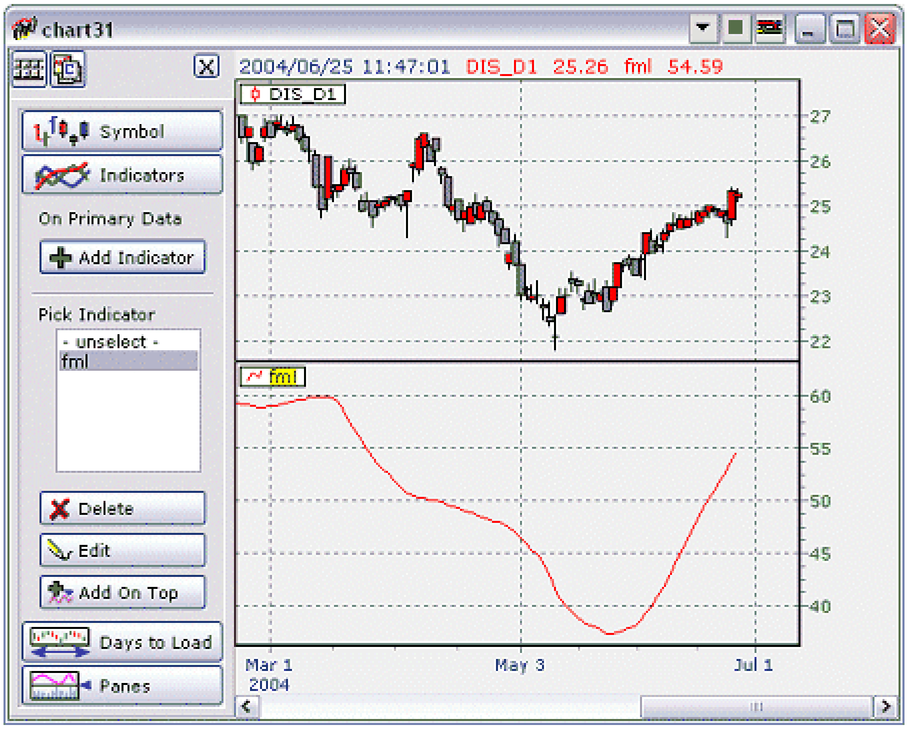


```
xaverage (rsindex (data1, 24), 20)
```

☑ Highlight Syntax

After you are done editing the expression, simply press the **OK** button to finish and return back to the Edit Indicator window.

Now press the **Apply** button to recalculate the indicator based on the new expression.



Tutorial: Making an Indicator with Formula Language, Example 1

Background

For quick calculations, specifying a formula using the formula indicator is perfect. To unleash the full power of NeoTicker®'s indicator, you will want to write indicators with formula language. Writing indicators with formula language offer features such as:

- You can control the number of plots
- You can control the number of data links
- You can specify user parameters to be used inside the indicator
- Reusable - your indicator will be available to other charts, scanner and quote window

This Tutorial

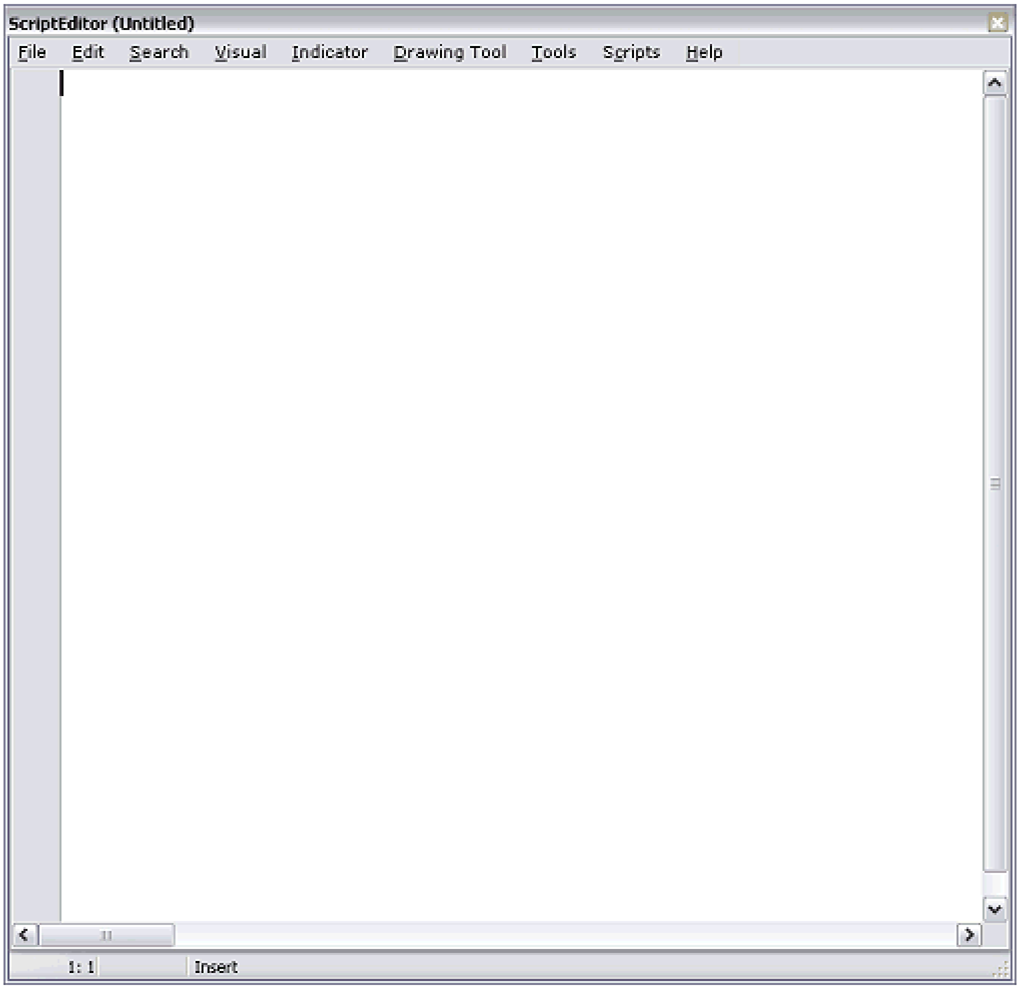
In this tutorial, you will create an reusable indicator using formula language. After you create this indicator, you can easily apply this indicator in other charts, pattern scanner and quote window as if the indicator is a build-in indicator.

This tutorial will make the indicator using the following formula:

```
r := RSIndex(0,data1,10);
mov(0,r,"Exponential",10)
```

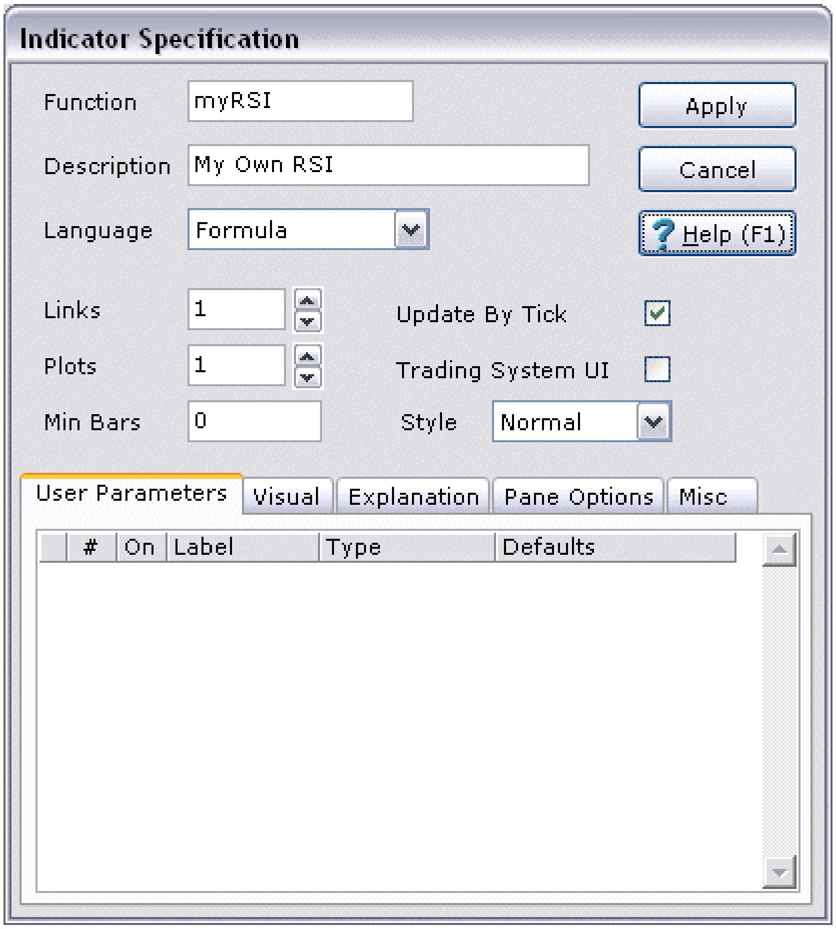
This formula is similar to the one created in ***Tutorial: Using Multi-Statement Formula to Calculate Indicator on Indicator*** (on page 356).

Choose from the main window, **Program>Script Editor>New**. This will open Script Editor for you to create indicator. If this is the first time you open the script editor, you may want to turn off the lower pane by choosing from Script Editor, **Visual>Visual 2** (having two panes are useful if you have a very long formula that covers multiple pages, but in our example, one pane is sufficient).



The first thing you want to do is to set up the indicator as an indicator that uses formula. Choose **Indicator>Setup** from Script Editor's menu.

Enter the parameters as follows.



Indicator Specification

Function: myRSI

Description: My Own RSI

Language: Formula

Links: 1

Plots: 1

Min Bars: 0

Update By Tick ☑

Trading System UI □

Style: Normal

Here is the explanation:

Function

This is the function name (short name) of the indicator. Short name is the name how other indicators reference this indicator. In this example, `myRSI`.

Description

This is the long name of the indicators. Long name are used in places such as Edit Indicator window. In this example `My Own RSI`.

Language

The language of this indicator. In this example, `Formula`.

Plots

Number of plots in this indicator. In this example 1.

Min Bars

Number of bars the input must have before the the indicator will start calculation. In this example, `0`.

Update By Tick

Whether the indicator will be updated by tick by default. In this example, **true**.

Trading System UI

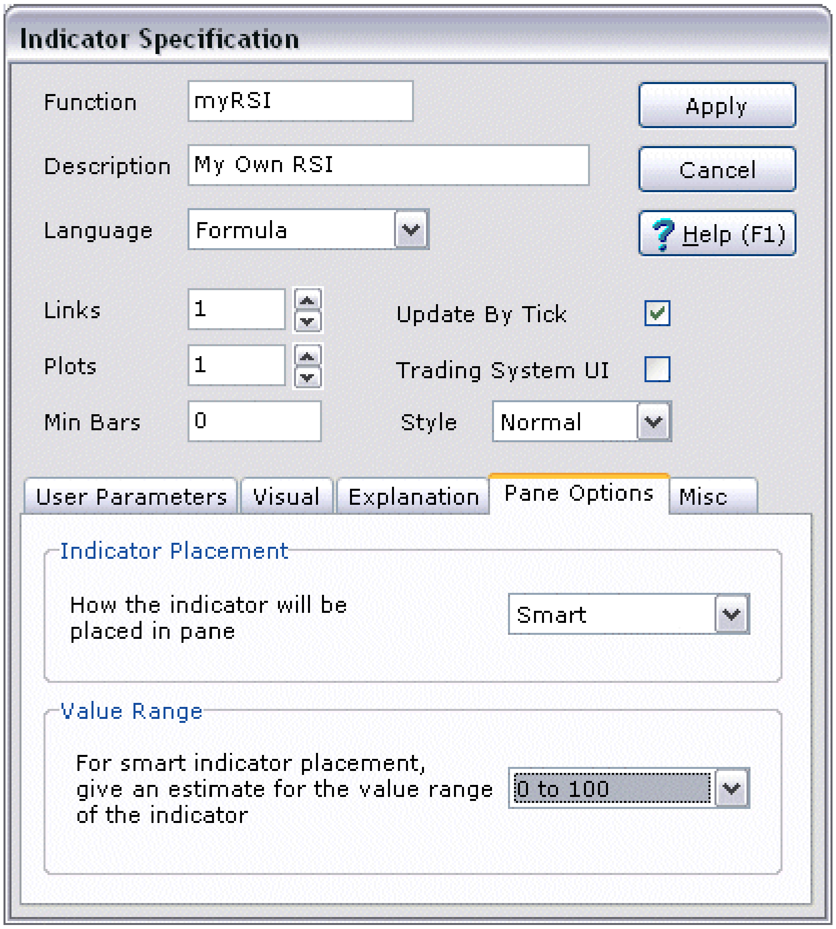
Whether the indicator supports trading system user interface. In this example, **false**.

Style

The drawing style of the indicator. In this example, **normal**.

Press the **Pane Options** tab. Under this tab, you will tell time chart how to place a newly created indicator in a chart.

Set **Indicator Placement** to **Smart** and **Value Range** to **0 to 100**. What this will do is to smartly place myRSI indicator in panes that contain mostly indicators with a range of 0 to 100.



Indicator Specification

Function: myRSI

Description: My Own RSI

Language: Formula

Links: 1

Plots: 1

Min Bars: 0

Update By Tick ☑

Trading System UI □

Style: Normal

Indicator Placement

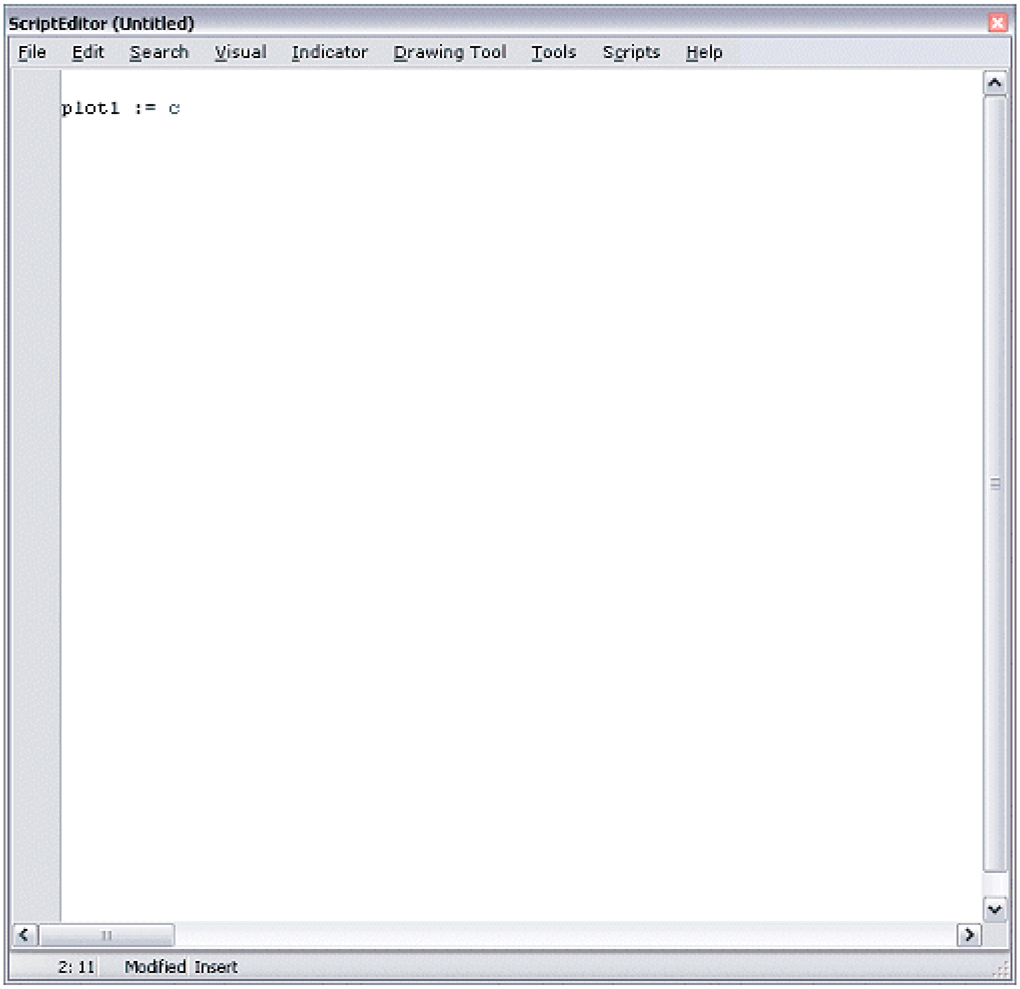
How the indicator will be placed in pane: Smart

Value Range

For smart indicator placement, give an estimate for the value range of the indicator: 0 to 100

Press the **Apply** button.

At this stage, Script Editor should look something like this:

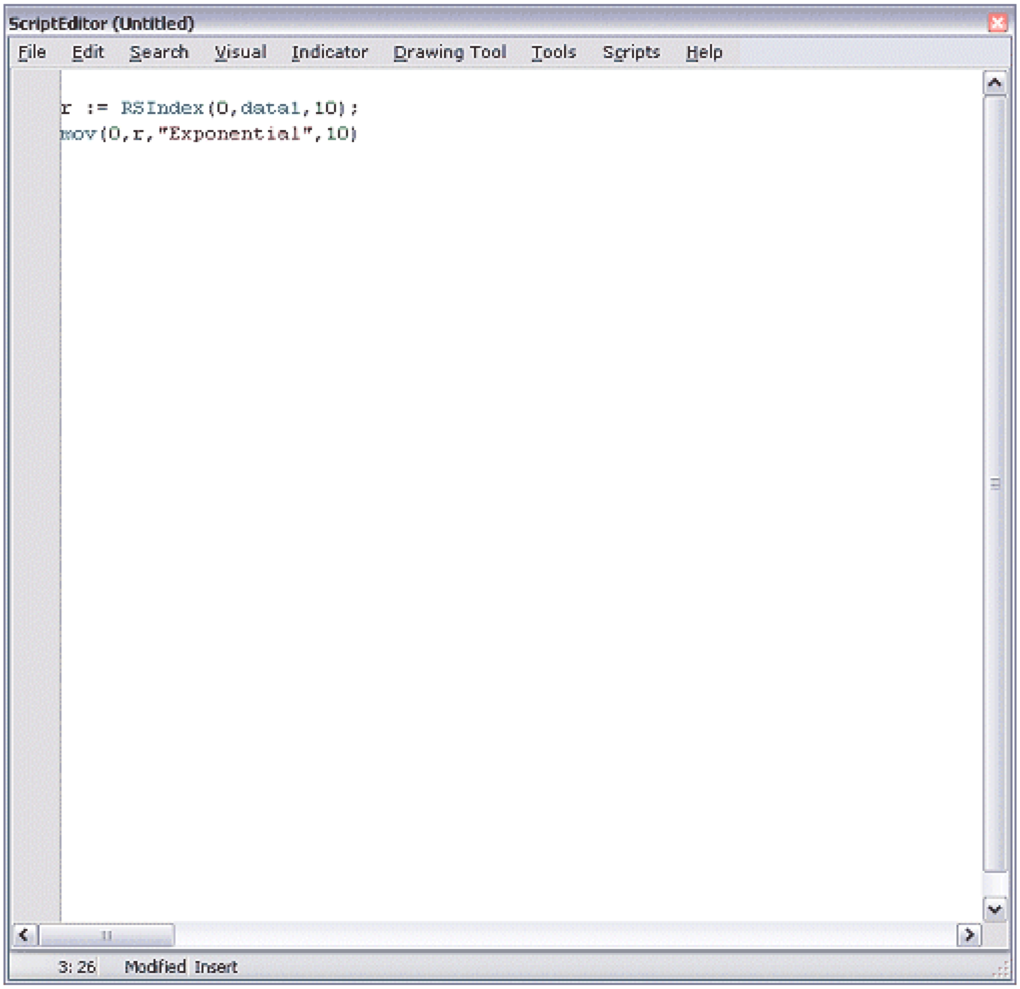

ScriptEditor (Untitled)
File Edit Search Visual Indicator Drawing Tool Tools Scripts Help
2: 11 Modified Insert

```
plot1 := c
```

Script Editor has already created an example formula for you. We are now ready to enter our own formula. Enter the following formula into Script Editor:

```
r := RSIndex(0,data1,10);
```

```
mov(0,r,"Exponential",10)
```



ScriptEditor (Untitled)
File Edit Search Visual Indicator Drawing Tool Tools Scripts Help

```
r := RSIndex(0,data1,10);
mov(0,r,"Exponential",10)
```

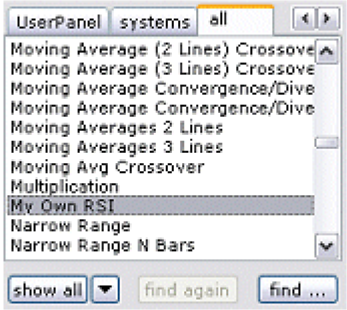
In order to use the indicator, you need to install it. Installing an indicator will make NeoTicker® aware that a new indicator becomes available. Choose **Indicator>Install** from Script Editor. Answer **Yes** when NeoTicker® asks you to save the indicator. Indicators are saved independently from charts so they will be available to NeoTicker® all the time.

A save dialog will open. Give the new indicator a file name, for example, myRSI. Then press the **Save** button.

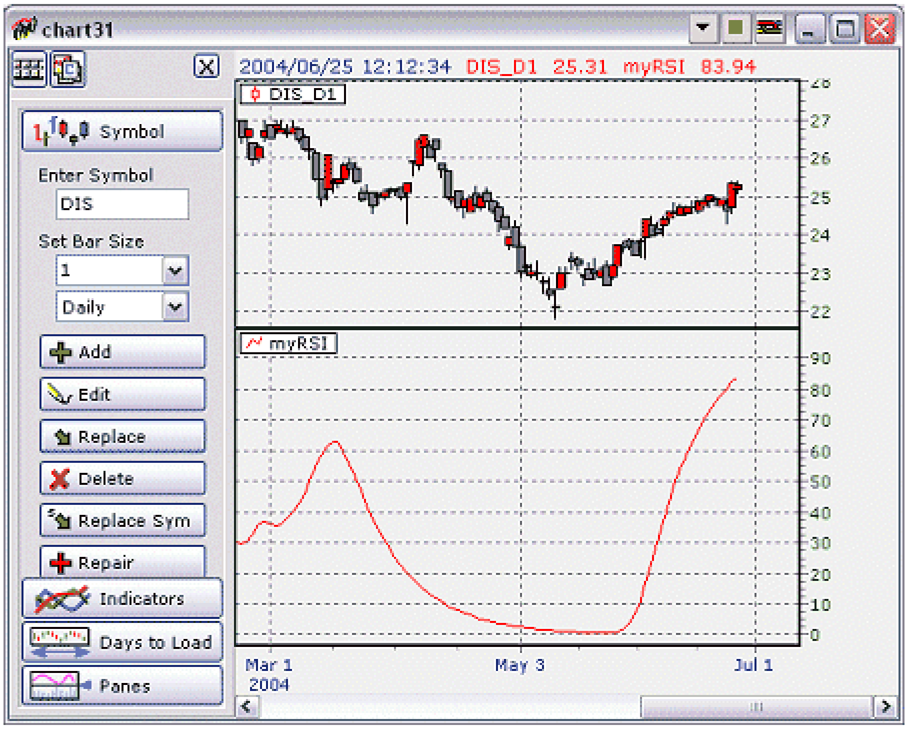
NeoTicker® will report to you that the indicator has been installed successfully. If you have trouble installing the indicator, check and correct any typo when entering the formula, and try installing again.

You can now close Script Editor.

To try out your new indicator, open a chart and **My Own RSI** will be available under the **all** tab when you add an indicator.



Moving Average (2 Lines) Crossove
Moving Average (3 Lines) Crossove
Moving Average Convergence/Dive
Moving Average Convergence/Dive
Moving Averages 2 Lines
Moving Averages 3 Lines
Moving Avg Crossover
Multiplication
My Own RSI
Narrow Range
Narrow Range N Bars

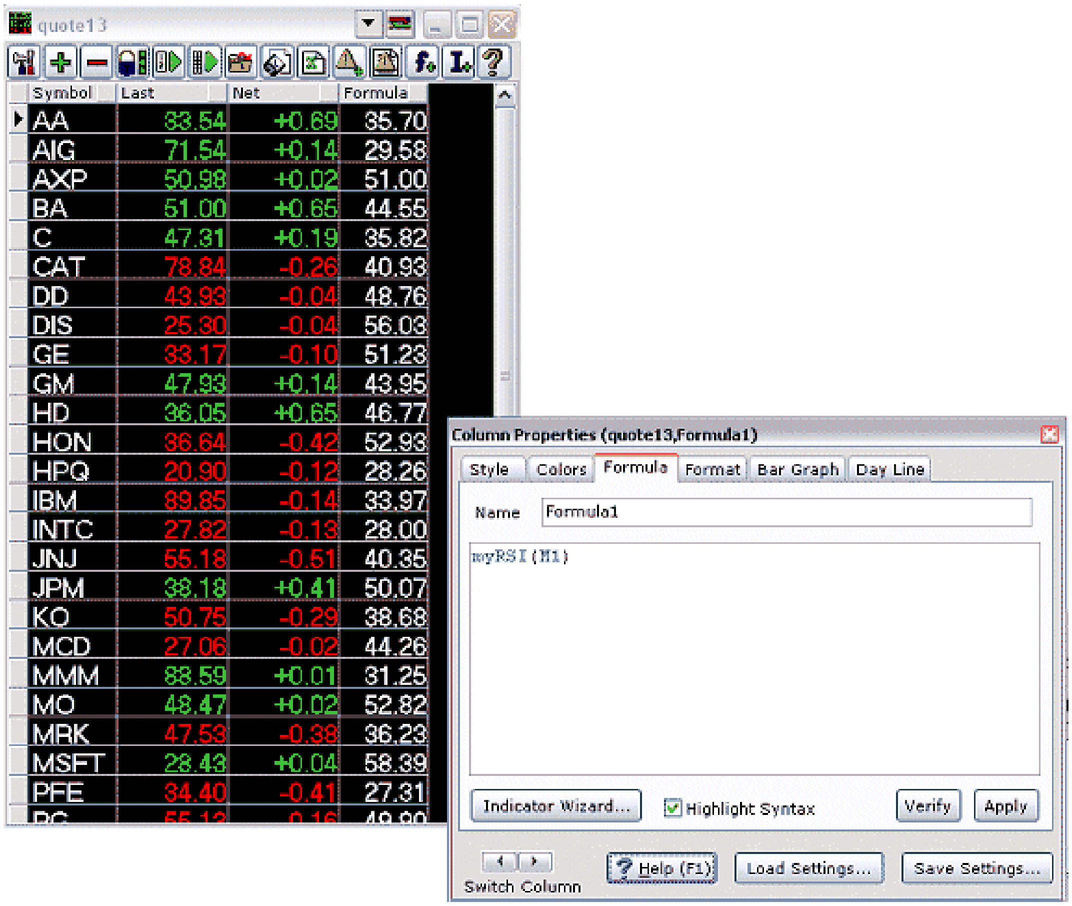


2004/06/25 12:12:34 DIS_D1 25.31 myRSI 83.94

Enter Symbol
DIS

Set Bar Size
1
Daily

You can also use the indicator in quote window by adding a myRSI indicator column.



quote13

Symbol	Last	Net	Formula
AA	33.54	+0.69	35.70
AIG	71.54	+0.14	29.58
AXP	50.98	+0.02	51.00
BA	51.00	+0.65	44.55
C	47.31	+0.19	35.82
CAT	78.84	-0.26	40.93
DD	43.93	-0.04	48.76
DIS	25.30	-0.04	56.03
GE	33.17	-0.10	51.23
GM	47.93	+0.14	43.95
HD	36.05	+0.65	46.77
HON	36.64	-0.42	52.93
HPQ	20.90	-0.12	28.26
IBM	89.85	-0.14	33.97
INTC	27.82	-0.13	28.00
JNJ	55.18	-0.51	40.35
JPM	38.18	+0.41	50.07
KO	50.75	-0.29	38.68
MCD	27.06	-0.02	44.26
MMM	88.59	+0.01	31.25
MO	48.47	+0.02	52.82
MRK	47.53	-0.38	36.23
MSFT	28.43	+0.04	58.39
PFE	34.40	-0.41	27.31

Column Properties (quote13,Formula1)

Name Formula1

```
myRSI(M1)
```

☑ Highlight Syntax

Tutorial: Making an Indicator with Formula Language, Example 2

This is the second example of creating indicator with formula language.

In this tutorial, we will create a more serious indicator called Ergodic Candlestick Oscillator, a popular tool for day traders. This tutorial also illustrates the following formula language features:

- Assignments
- User parameter
- Multiple plots

This tutorial ends with a discussion on the differences between script-based indicator and formula-based indicator.

You can find more information about this indicator in the book, "Momentum, Direction and Divergence" by William Blau.

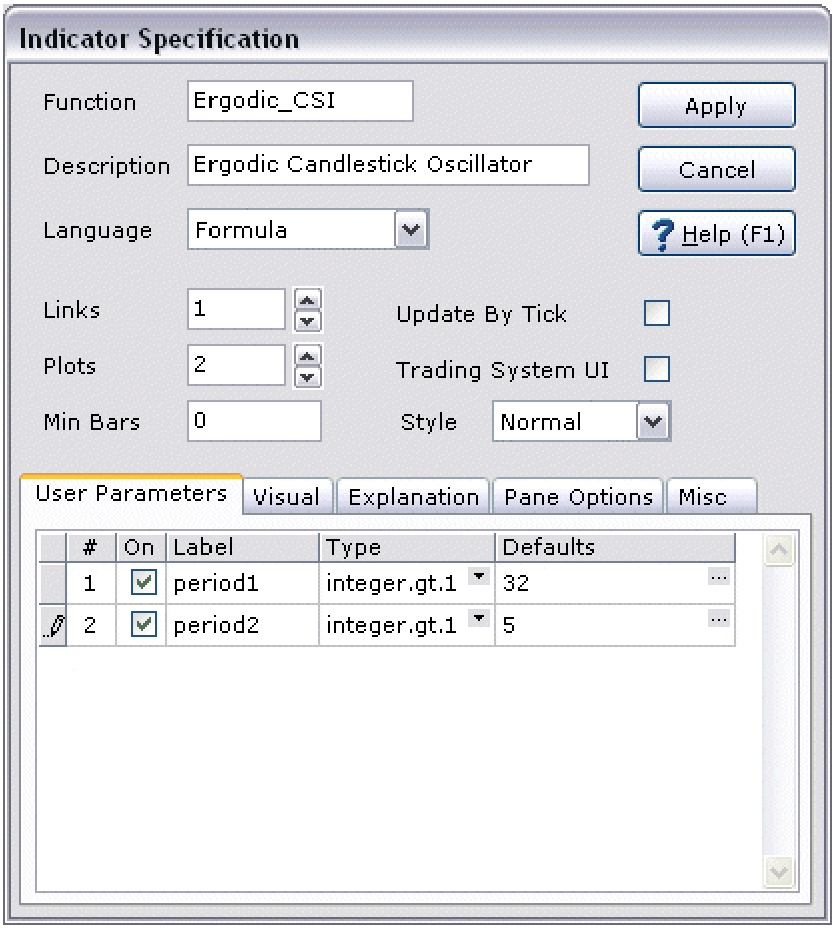
Preparing the Formula

To create a formula indicator open a new script by selecting **Program>Script Editor>New** to open a blank Script Editor.

To set up the indicator, in Script Editor, choose **Indicator>Setup**.

- In the **Function** field type `Ergodic_CSI`
- In the **Description** field type `Erogodic Candlestick Oscillator`
- Select **Formula** as **Language**
- Increase the number of **Plots** from 1 to 2
- Right click on the white space under **User Parameters** add a parameter
- In the **Label** field enter `period1`, change the **Type** to **integer.gt.1**;in the **Defaults** field enter `32`
- Right click to add another parameter; in the **Label** field enter `period2`, change the **Type** to **integer.gt.1**; in the **Defaults** field enter 5
- Click on the **Visual** tab to change the color of the second line to blue

This is what you should have afterward.



Indicator Specification

Function: Ergodic_CSI

Description: Ergodic Candlestick Oscillator

Language: Formula

Links: 1

Plots: 2

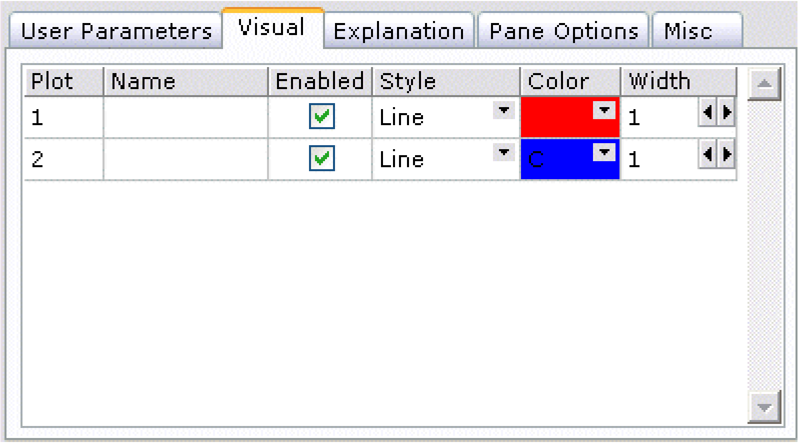
Min Bars: 0

Update By Tick □

Trading System UI □

Style: Normal

#	On	Label	Type	Defaults
1	☑	period1	integer.gt.1	32
2	☑	period2	integer.gt.1	5



Plot	Name	Enabled	Style	Color	Width
1		☑	Line	🟥	1
2		☑	Line	🟦 C	1

Press the **Apply** button to accept the indicator specifications..

Adding a Single Plot

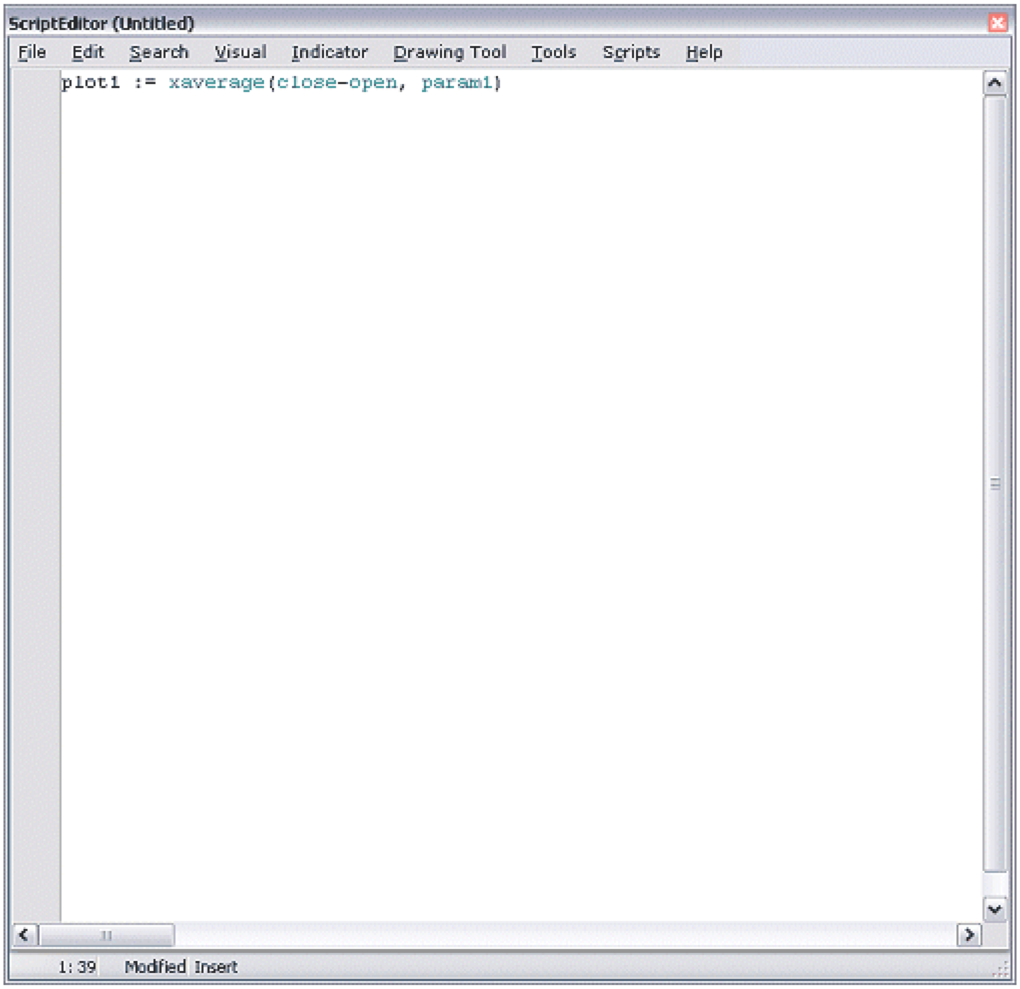
To start constructing the Ergodic candlestick oscillator, we start by writing a difference of open and close with exponential smoothing.

For the indicator to plot anything you will have to assign a formula calculation to a Plot line.

In Script Editor, change the line to:

```
plot1 := xaverage(close-open, param1)
```

The formula language you use here is exactly the same as the formula you would have specified to the formula indicator.



```
plot1 := xaverage(close-open, param1)
```

Lets break down the formula into components to get a better understanding of what this formula means.

Part	Explanation
`plot1`	is a reserve word in formula language. When you assign values to plot1 to plotN, you are telling the indicator to draw those values using that particular plot line.
`xaverage`	is an indicator function, xaverage is a built-in indicator, which takes the result of its first parameter and gives it an exponential smoothing base on the second parameter. You can use any user defined script-based or formula-based indicators
`close - open`	is the close of first link (data1) subtract open of first link
`param1`	is the user parameter, the number of parameters vary among indicators

The next step is to verify the indicator.

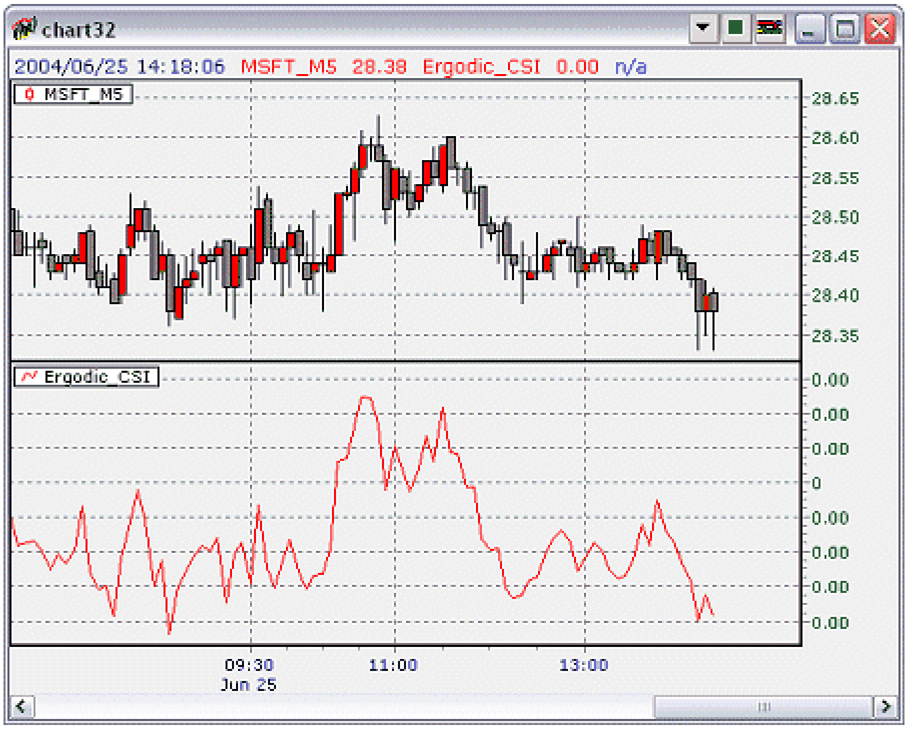
Verifying an indicator means syntactically checking the indicator. Verifying lets you check an indicator before committing it to actual usage. To verify the indicator, choose **Indicator>Verify** from the script editor. The script editor will check if the formula you typed are valid.

If there is no typo in the formula, NeoTicker® will tell you the verification has succeed:

If there is any typo, you can edit the formula and verify the indicator again.

To put the indicator into actual usage, you need to install it. Installing an indicator tells the rest of NeoTicker® that a new indicator that is available for use. Once an indicator is installed, you can access it from windows such as the Add Indicator window just like any other indicators NeoTicker® provides you. To install the indicator, choose **Indicator>Install** from Script Editor.

You can try this formula on a price series, in the following figure we have applied it on Microsoft with 5 days of 5-mintes bar.



You can verify your work by reproducing the result using the indicator on indicator method.

You can manually create an Ergodic candlestick oscillator with indicator-on-indicator to verify the formula indicator you just created is correct:

1 Create the subtraction indicator and select open as the first link and close as the second link, this will give you the difference between open and close of the 5-minutes candlestick.

2 Add Exponential Moving Average and use diff as the link. Set the period of xaverage to 32 and diff to not visible, then you should see an indicator line that looks exactly the same as the one created with your formula.

Completing the Formula

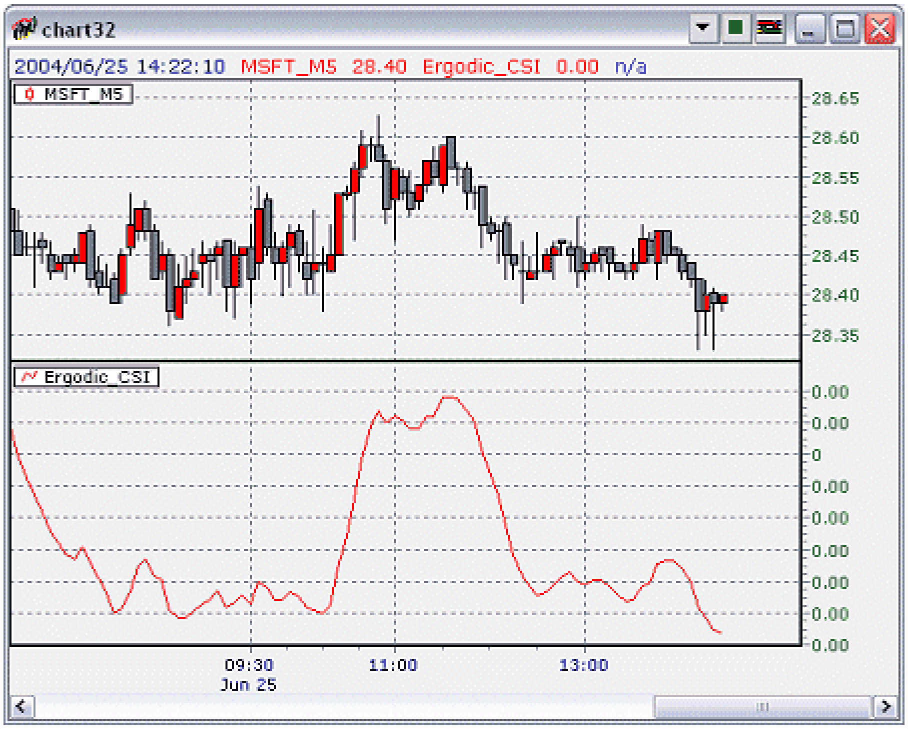
Erogodic CSI is essentially a double exponential smoothing of close subtracting open, then divide by double exponential smoothing of high subtracting low of a candlestick.

To create the Ergodic CSI nominator you need to double smoothing the difference between close and open, so we will have to feed the result of the single smoothing into another exponential smoothing function.

In Script Editor, change the formula to:

```
Plot1 := xaverage(xaverage(close-open, param1), param2)
```

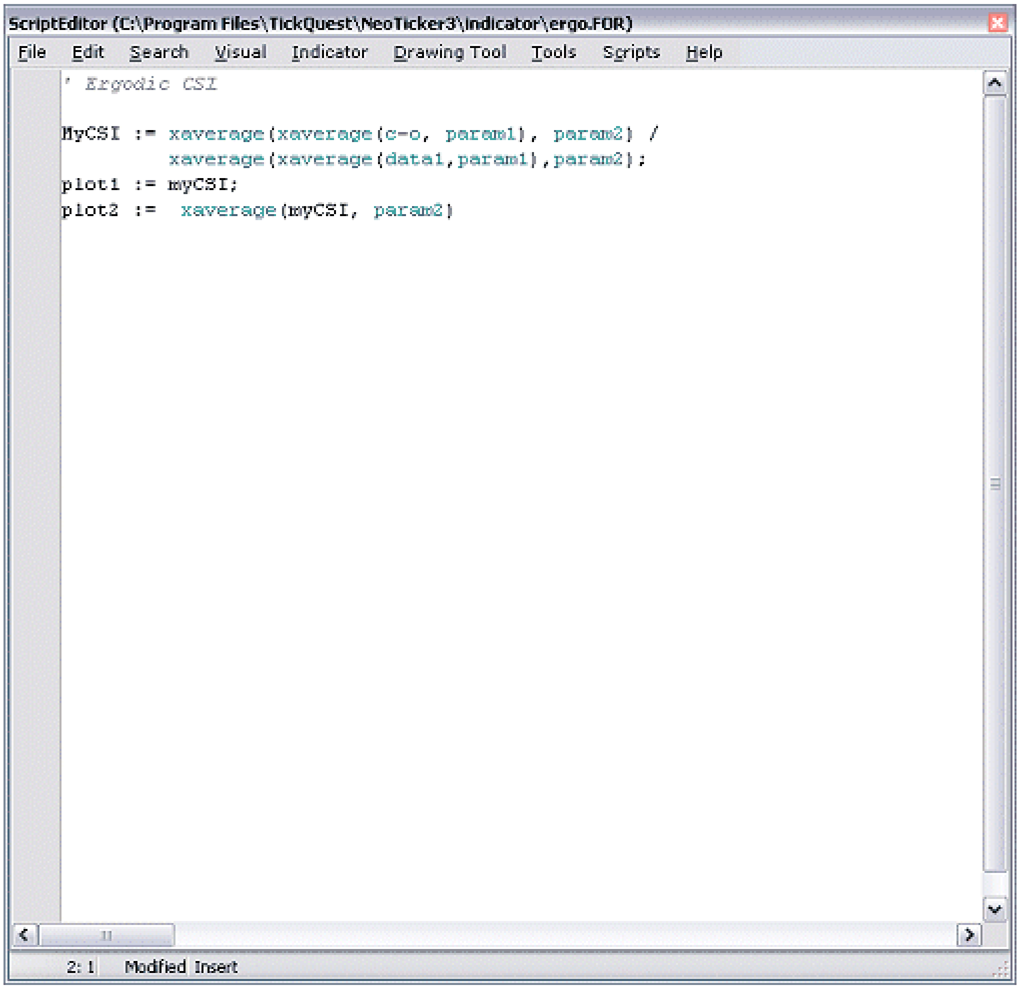
This line we have taken a 5 period smoothing of the result of the first smoothing. Verify and install this new formula. In the chart, press the refresh button to recalculate Erogdodic CSI.



We are now ready to create the final version of Ergodic CSI. In Script Editor, change the formula to:

```
MyCSI := xaverage(xaverage(c-o, param1), param2) /
         xaverage(xaverage(data1,param1),param2);
plot1 := myCSI;
plot2 :=  xaverage(myCSI, param2)
```

You can add comment to your indicator to help clarify the meaning of the formulas. The following is the formula indicator with comment:

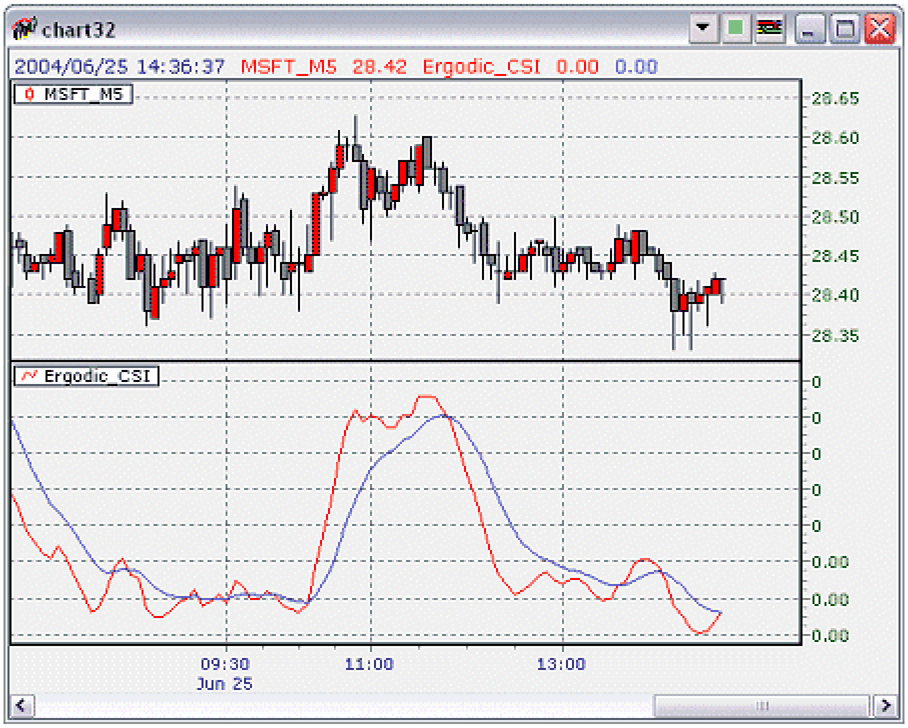


ScriptEditor (C:\Program Files\TickQuest\NeoTicker3\indicator\ergo.FOR)
File Edit Search Visual Indicator Drawing Tool Tools Scripts Help

```
' Ergodic CSI

MyCSI := xaverage(xaverage(c-o, param1), param2) /
         xaverage(xaverage(data1,param1),param2);
plot1 := myCSI;
plot2 :=  xaverage(myCSI, param2)
```

At this point you have completed the formula to construct the indicator Ergodic Candlestick Oscillator.

Install the formula and refresh the chart by pressing the refresh button in the tool bar, the chart will look like:



You must refresh the chart after you change definition of an indicator.

Formula Language vs. Scripting Languages

Now you've learned how to create indicator using formulas. Because you can also create indicators using scripting languages (such as VBScript and Delphi Script), the obvious question is whether you should spend time learning how to write indicators using scripting languages.

The answer depends on your requirements. Formula language is very powerful as it is. You will want to investigate the scripting languages if you have the following requirements:

- Complex data structure and logic such as local functions and variables, loops and arrays
- Using NeoTicker®'s complete mathematical and string library
- Data storage between bar calculations
- Accessing advanced NeoTicker® objects
- Writing user defined drawing tools
- Connectivity to external DLL's and executables
- If you eventually want to port the indicator to an external programming language such as Visual Basic or Delphi because VBScript and Delphi Script syntax is almost identical to the scripting languages.

Tutorial: Fast Way to Mark Your Custom Conditions Using the Indicator Highlight Formula

Background

Do you want to quickly create a marker to highlight a special condition using your favorite indicators and would like to visually examine that on a chart?

Or are you interested to highlight a price pattern you think that has some consistency over time and you would like to examine it further on historical data?

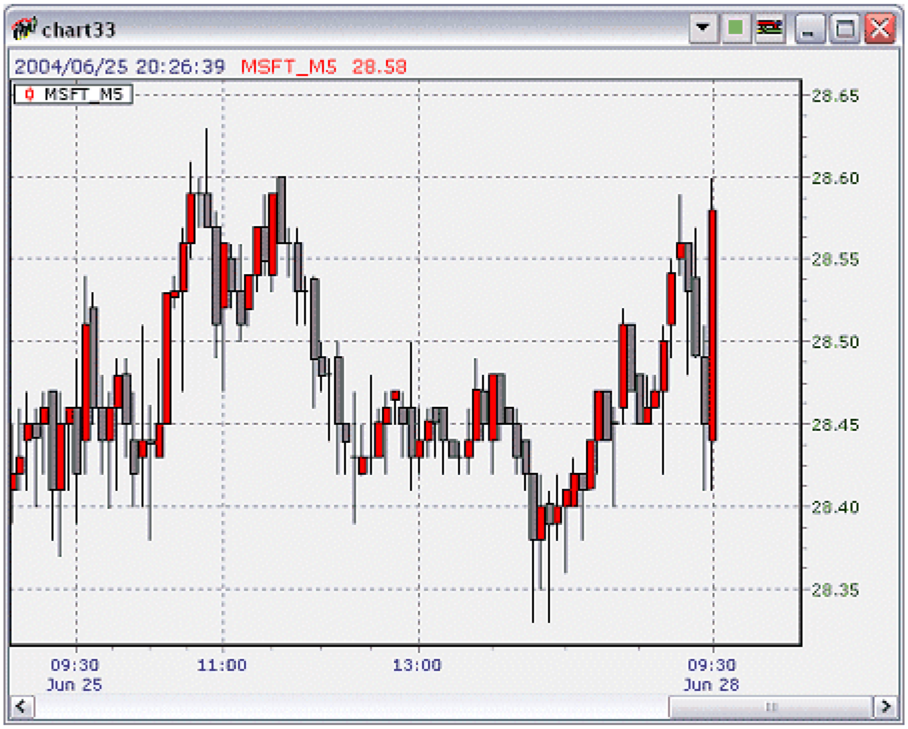
You can successfully accomplish these tasks using the highlight formula indicator to quickly get the results you want. Then if you want to save the result, you can easily convert the result into your own custom indicator.

We will also touch on the subject of assignment and bar validity.

The Tutorial

Say you are interested in locating price bars pulling back to their moving average in real-time to see if they are good buying opportunities. You figure that using a 50-period simple moving average with stochastic bottoming could be what you are looking for.

You can create a new chart with 5-min bars of MSFT.

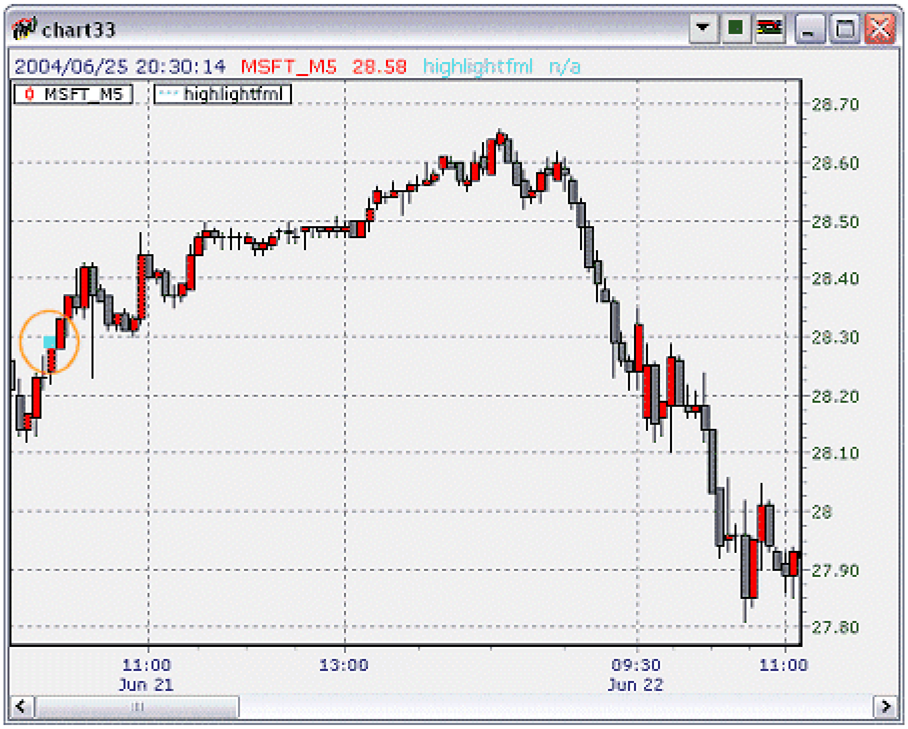


Now apply the Highlight Formula indicator to the chart.

Enter the following formula to the **Condition** parameter:

```
c > Average (data1, 50) and slowd (data1, 5, 3) >= 20 and
slowd (1, data1, 5, 3) < 20
```

And change the **Position** parameter to **High**. Press the **Apply** button. Then move the Highlight Formula to the same pane as the data series. The chart will look like this:



This condition is not very common, so you may want to experiment with a few stocks to spot it.

After visual inspection if you find this custom condition looks useful for real-time monitoring, you can create a formula indicator to track this condition easily.

1 To create a new indicator, choose **Program>Script Editor>New** in main window.

2 Now choose **Indicator>Setup** in Script Editor.

3 Set the indicator description to `My Bottom`.

4 Set the indicator function name to `MyBottom`.

5 Set the number of links to 1, number of plots to 1, meta plot style to Normal.

6 Set the plot 1 style to **Dot**.

In the script editor, type in the following formula lines:

```
MyCond := c > Average (data1, 50) and slowd (data1, 5, 3) >= 20
and slowd(1, data1, 5, 3) < 20;
```

```
Plot1 := H;
Success1 := MyCond;
```

The MyCond assignment directly reuses the condition you have typed in the highlight indicator Condition parameter. Thus you can simply copy and paste that into the script editor without typing.

`Success1` marks the validity of the indicator. Setting `Success1` to 0 or false marks the current bar as invalid (e.g. `Success1 := 0;`). An invalid bar has no value and is not marked on the chart.

Save the indicator. Choose **Indicator>Install** in the script editor to install this new indicator.

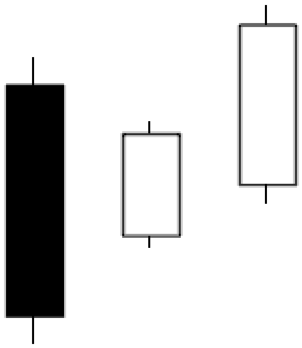
Now you can reuse this condition in anywhere that accepts an indicator.

Tutorial: How to Highlight Complete Bars with Custom Conditions Using the Indicator Highlight Bar Formula

Lets say you want to identify the candlestick pattern Harami Age (Three Inside Up) and draw a bar from high to low onto the price bar itself for easier identification visually.

Description of the Harami Age Pattern can be found in the book "Candlepower" by Gregory L. Morris.

Here is the price pattern Harami Age.



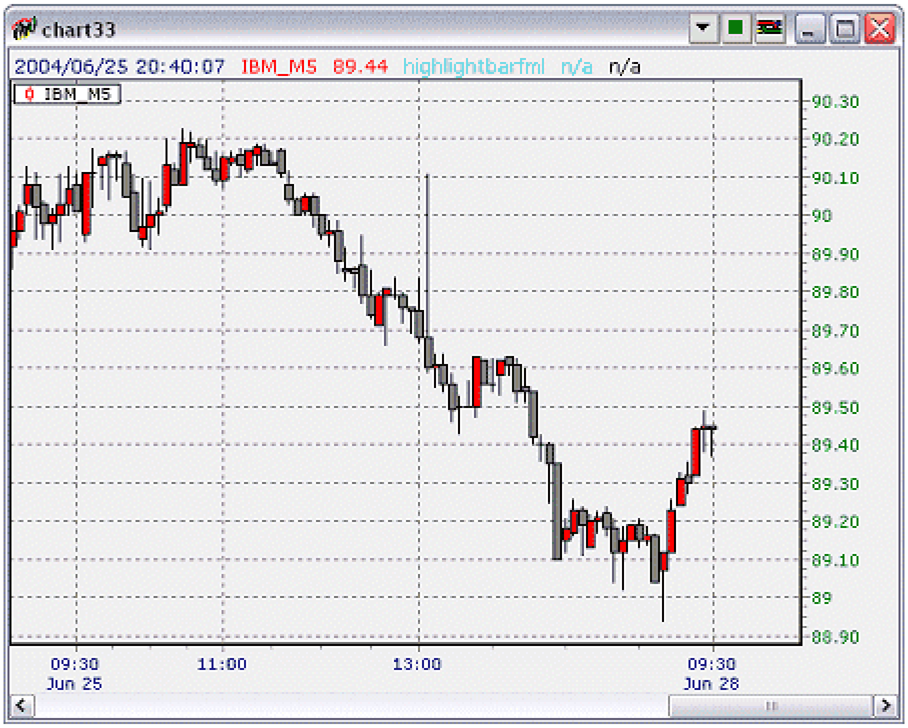
Basically, we can break down the formation to the following rules.

1 1 bar ago the bar is completely inside the range of a down bar 2 bars ago (Harami)

2 1 bar ago is an up bar

3 Current bar has a higher close

Now, create a new chart with the symbol IBM using 5-min bar.

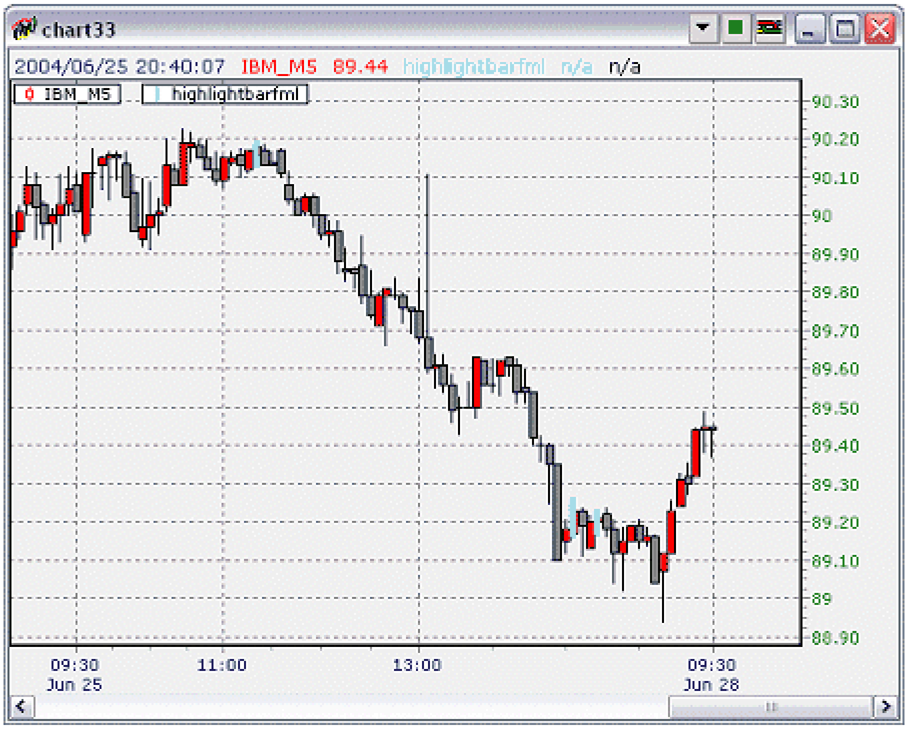


Now apply Highlight Bar Formula indicator to the chart.

Type in the following formula to the **Condition** parameter:

```
H (2) > h (1) and l (2) < l (1) and
c (2) < o (2) and c (1) > o (1) and c > c (1)
```

Under the **Visual** tab, change the meta style of the indicator to **HighLow**. Press the **Apply** button. Then move High Bar Formula to the same pane as the data series.



After visual inspection if you find this condition looks useful for real-time monitoring, you can create a formula indicator to track this condition easily.

1 To create a new indicator, choose **Program>Script Editor>New** in main window.

2 Choose **Indicator>Setup** in the Script Editor.

3 Set the indicator description to `My Harami Age`.

4 Set the indicator function name to `MyHaramiAge`.

5 Set the number of links to 1, number of plots to 2, meta plot style to **HighLow**.

In the script editor, type in the following formula lines:

```
MyCond := H (2) > h (1) and l (2) < l (1)
          and c (2) < o (2) and c (1) > o (1) and c > c (1);
Plot1 := H;
Plot2 := L;
Success1 := MyCond;
Success2 := MyCond;
```

The `MyCond` line is a copy of the condition you have typed in previously, simply copy and paste the original condition to avoid typos.

The main difference in defining a highlight bar indicator from other type is the use of the meta plot style HighLow. When using this meta plot style to define indicator, NeoTicker® draws plot1 and plot2 together as a bar instead of lines within themselves only.

Save the indicator. Choose **Indicator>Install** in the script editor to install this new indicator.

Now you can reuse this condition in time chart, quote window or pattern scanner.

For advanced users who like to change the color of the highlight bar within the indicator depending on different conditions, you can make your indicator to have 3 plots instead of 2. Then plot3 is the color specification and you must specify meta plot style to HighLowColor to get the coloring effect.

Tutorial: Creating Your Own Index using Formula Language

In this tutorial, we will create your own index. An index is a weighted sum of a basket of symbols, usually stocks.

This tutorial demonstrates the unique power of NeoTicker® of constructing data from multiple plot line indicators. You can tap into this power by using formula.

It also illustrates the usage of user parameters in indicators using formula language.

You should already know something about script editor and creating a indicator using formula language. For more information, consult ***Tutorial: Making an Indicator with Formula Language, Example 1*** (on page 364).

First, we will create a 4-plot indicator. The four plots represent the open, high, low, close of your index. Create a new Script Editor and choose **Indicator>Set up**.

Enter the following information into the setup window under the **User Parameters** tab.

Function

Enter `myIndex.`

Description

Enter `My Own Index.`

Language

Choose **Formula.**

Links

We will create a 3-component index. So set **Links** to 3.

Plots

Set to 4. The plots will represent open, high, low, close values.

Min Bars

Set to 0.

Update By Tick

Set to true.

Trading System UI

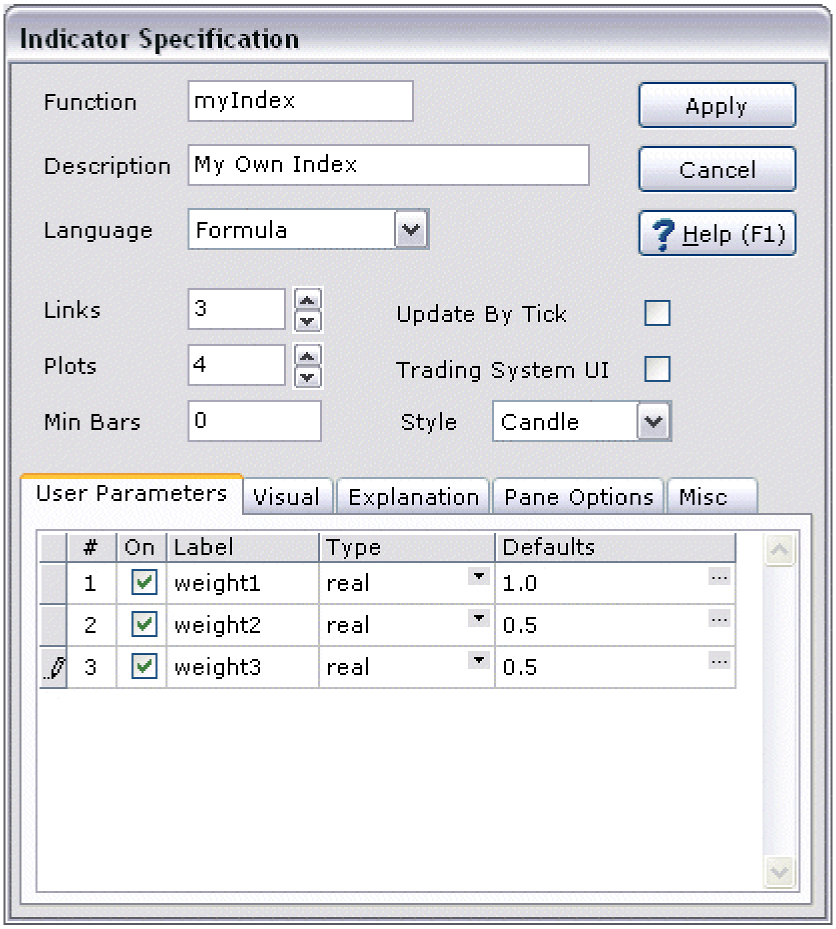
Set to false.

Style

Set to **Candle**. This tells the indicator to behave as candlesticks, with the 4 plots representing open, high, low, close.

Under **User Parameters** tab, right click on the blank area to open the pop up menu, and choose **Add** to add user parameter. We will use the parameters to assign weights to the components in the index. Add three parameters to the user parameters. Enter `weight1`, `weight2`, `weight3` as labels, set the type to **real** and assign `1`, `0.5`, `0.5` for the default values.

The window should look like:



Indicator Specification

Function: myIndex

Description: My Own Index

Language: Formula

Links: 3

Plots: 4

Min Bars: 0

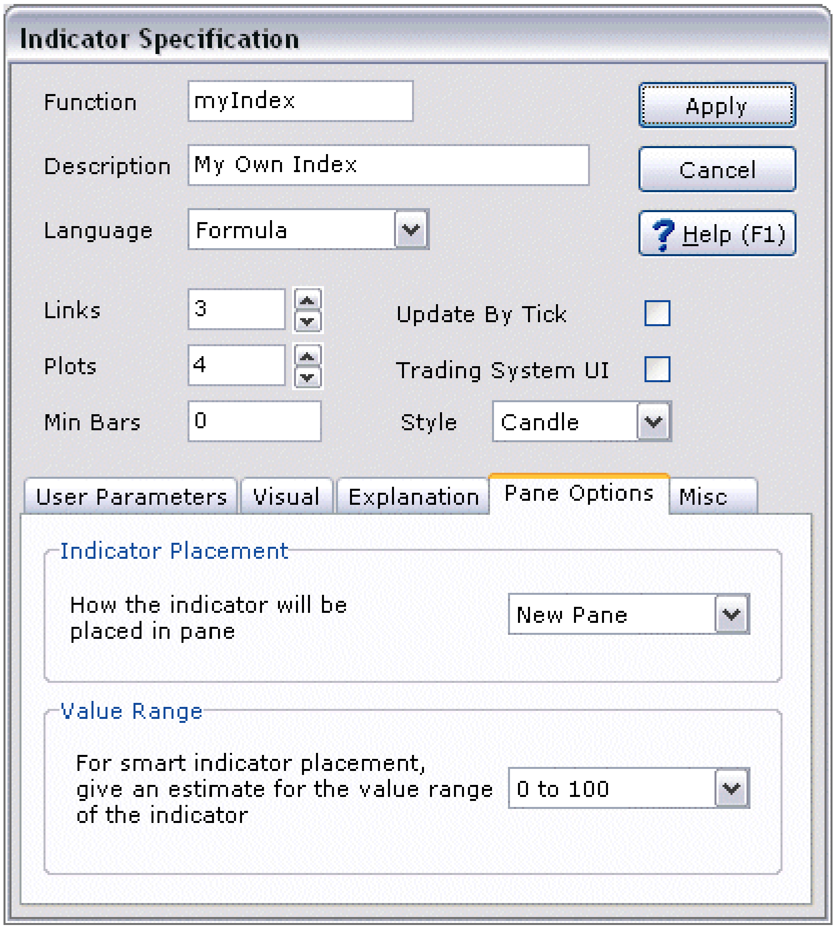
Update By Tick ☐

Trading System UI ☐

Style: Candle

#	On	Label	Type	Defaults
1	☑	weight1	real	1.0
2	☑	weight2	real	0.5
3	☑	weight3	real	0.5

We want this indicator to appear in a new pane when it is added to a time chart, so press the **Pane Options** tab, and choose **New Pane** for **Indicator Placement**. Our guess is the index value won't be too large for 3 components. So assign **0 to 100** for **Value Range**.



Indicator Specification

Function: myIndex

Description: My Own Index

Language: Formula

Links: 3

Plots: 4

Min Bars: 0

Update By Tick □

Trading System UI □

Style: Candle

Indicator Placement

How the indicator will be placed in pane: New Pane

Value Range

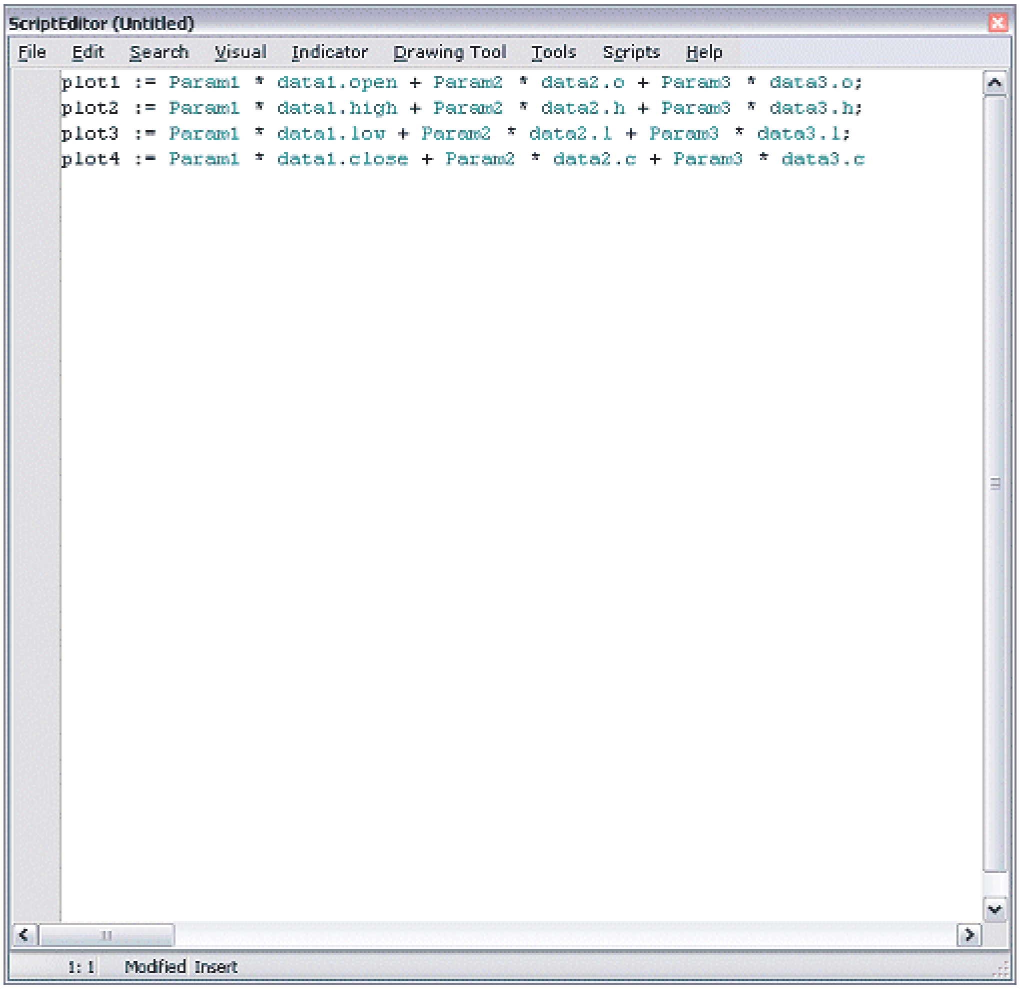
For smart indicator placement, give an estimate for the value range of the indicator: 0 to 100

Press the **Apply** button to finish setting up the indicator.

Replace the default formula in the script editor with the following formula:

```
plot1 := Param1 * data1.open + Param2 * data2.o + Param3 *
data3.o;
plot2 := Param1 * data1.high + Param2 * data2.h + Param3 *
data3.h;
plot3 := Param1 * data1.low + Param2 * data2.l + Param3 *
data3.l;
plot4 := Param1 * data1.close + Param2 * data2.c + Param3 *
data3.c
```

Script Editor should look like:



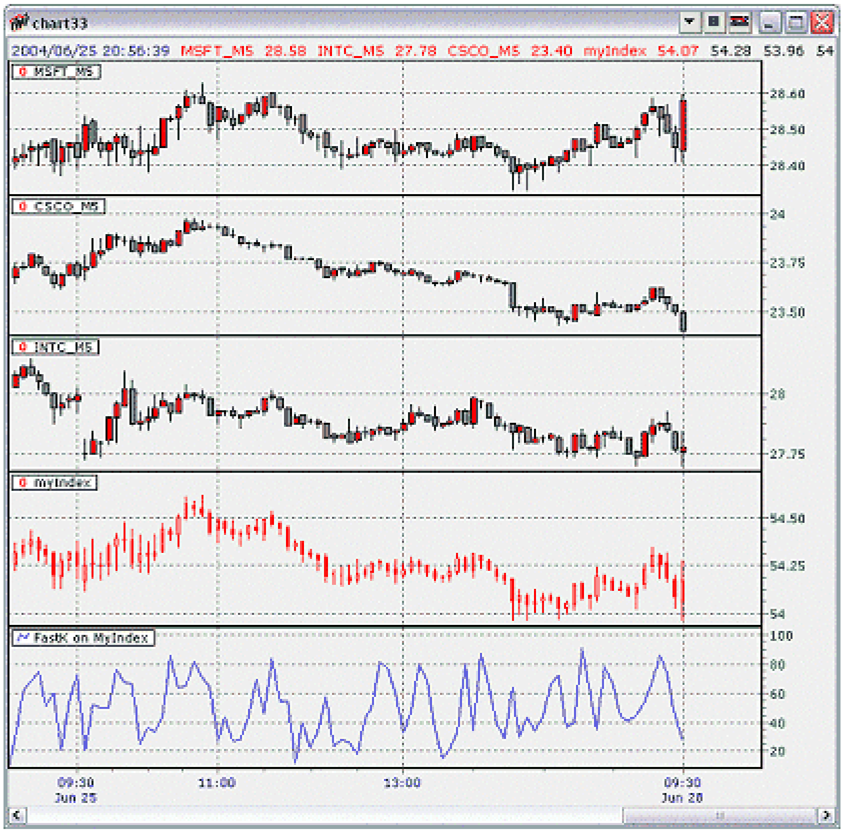
ScriptEditor (Untitled)
File Edit Search Visual Indicator Drawing Tool Tools Scripts Help
1: 1 Modified Insert

```
plot1 := Param1 * data1.open + Param2 * data2.o + Param3 * data3.o;
plot2 := Param1 * data1.high + Param2 * data2.h + Param3 * data3.h;
plot3 := Param1 * data1.low + Param2 * data2.l + Param3 * data3.l;
plot4 := Param1 * data1.close + Param2 * data2.c + Param3 * data3.c
```

Install the indicator.

Now you want to test the indicator. Open a chart, add 3 data series to the chart, and apply the **My Own Index** indicator to the chart. Make sure you link **My Own Index** to the 3 data series in the chart.

Because **My Own Index** is displaying as candlesticks, you can even apply indicators to **My Own Index** and expect **My Own Index** to behave as if it is a data series. For example, you can add a **Stochastic FastK** indicator to **My Own Index** and expect the OHLC values to be picked up correctly.



CHAPTER 11

Operation Guide

In This Chapter

Shortcut Summary

Context Sensitive Help

Press F1

Mouse shortcut - general

Right-click On any function window, bring up its general operation menu. If an item is selected in the function window, then the related special object menu will be shown instead.

Left-click Select the item directly underneath the cursor within a function window.
Un-select the current selected item if click on blank area within the same function window.

Keyboard shortcut - general

F1	Context sensitive help
F2	Buy order (refer to ***Order Interface Manager*** (see "Order Interface" on page 735) for more detail)
F3	Sell order (refer to ***Order Interface Manager*** (see "Order Interface" on page 735) for more detail)
F4	Show/Hide the Chart Tool bar
F5	Toggle main window
F6	Show/Hide Data View window
F8	Replace symbol in chained function windows
F9	Replace symbol in active window
F12	Auto repair data for all windows
CTRL-F9	Minimize window
CTRL-F10	Maximize/restore window

CTRL-J	Take screen shot
CTRL-S	Save all opened groups
CTRL-P	Open print dialog
ENTER	Similar to press the OK or Apply button if one is there in the current dialog
ESC	Close the current setup or edit dialog like pressing a regular Cancel button
SPACE	Launch the setup or edit dialog of the current function window or current selected object
PAGE UP	Switch to the previous opened group
PAGE DOWN	Switch to the next opened group
CTRL-TAB	Rotate to next function window in active group

ALT-M	Use keyboard to move a window
ALT-S	Use keyboard to resize a window
CTRL-ALT-N	For quote.com only to switch to next available server

Main Window

CTRL-O	Open previously saved session
CTRL-S	Save all groups
Double-clicks	On windows list, minimize / normal size a function window

Symbol List Manager

DELETE	In the symbol listing area, remove the current selected symbol

Time Chart

F11	Repair intraday data if your data vendor provides historical data
DELETE	Delete the currently selected object
HOME	Jump to earliest part of chart
END	Jump to latest part of chart
SPACE	Launch the setup or edit dialog of the currently selected object
+	Zoom In
SHIFT+	Zoom In (less strength)
-	Zoom Out
SHIFT-	Zoom Out (less strength)

TAB	Rotate through all the available cursor mode. If you are using a drawing tool, it will terminate the drawing mode and switch back to the first basic cursor
CTRL-0 to CTRL-9	Change Time Frame and Days To Load based on settings in the Shortcut Manager
Left-click-drag	Move a selected object to another pane. If the selected object is moved to the edge of a pane, a new pane will be automatically created to hold the object. If left-click on an edge of a pane, will resize the pane on-the-fly
Double-click	On data legend will show data editing window On indicator legend will show indicator editing window On pane price scale to edit pane scale and setting

Quote Window

CTRL-C	Copy rows.
CTRL-V	Paste rows. When pasting to a quote window, only the symbols are pasted. When pasting to an external program such as Excel, all the cell values are pasted.
ENTER	Rotate send to chain

Double-click	On symbol field equivalent to sending symbol to chain On a position related field will pop the edit position dialog On on notes field will pop up edit notes dialog On formula column will pop up edit formula edit window On any other column will pop up edit column properties window

Pattern Scanner

ENTER	Rotate send to chain
Double-click	Equivalent to sending symbol to chain

Alert Window

Double-click	Open the news message for news alert

Chart Manager

Double-click	On a data, indicator, or object item will launch its edit dialog
DELETE	Delete the selected item in the current active listing

Script Editor

CTRL-N	New script
CTRL-O	Open script
CTRL-S	Save script
CTRL-ALT-S	Save As
CTRL-Z	Undo
CTRL-X	Cut

CTRL-C	Copy
CTRL-V	Paste
CTRL-Del	Delete
CTRL-A	Select All
CTRL-F	Find
CTRL-L	Replace
CTRL-H	Find Again
CTRL-G	Goto Line

Script Error Log

double-click	The script that caused the error will be opened with the cursor moved to the appropriate line. If the script is already opened, it will be activated instead

Indicator Manager

double-click	To launch script editor and editor the indicator you've clicked on

Activating Products from Third Party Developers

There are two steps to activate a product from a third party developer.

Sending Information About NeoTicker® to the Developer

This step provides information to the third party developer so that they can create a key to activate their product for your copy of NeoTicker®.

The third party developer will instruct you on how to send the information to them. Usually, you will need to choose from main window, **Help>User Info** to open a dialog, and send the information in the dialog to the third party developer.

Authorizing the Product

Once the third party developer receives the information, they can generate an activation key file and send it to you.

The activation key file is a text file. After you receive the activation key file from the third party developer:

1. In main window, choose **Tool>Activation Tool** to open Activation Tool.
2. In Activation Tool, press **Install Activation File** tab.
3. You can drag the activation key file to the area labelled **Drag and Drop Activation File Here**. Alternatively, you can press the **Browse For Activation File** button to locate the activation file yourself.

Alert Tracking

Basic Concept of Alerts

There are two types of alerts in NeoTicker®:

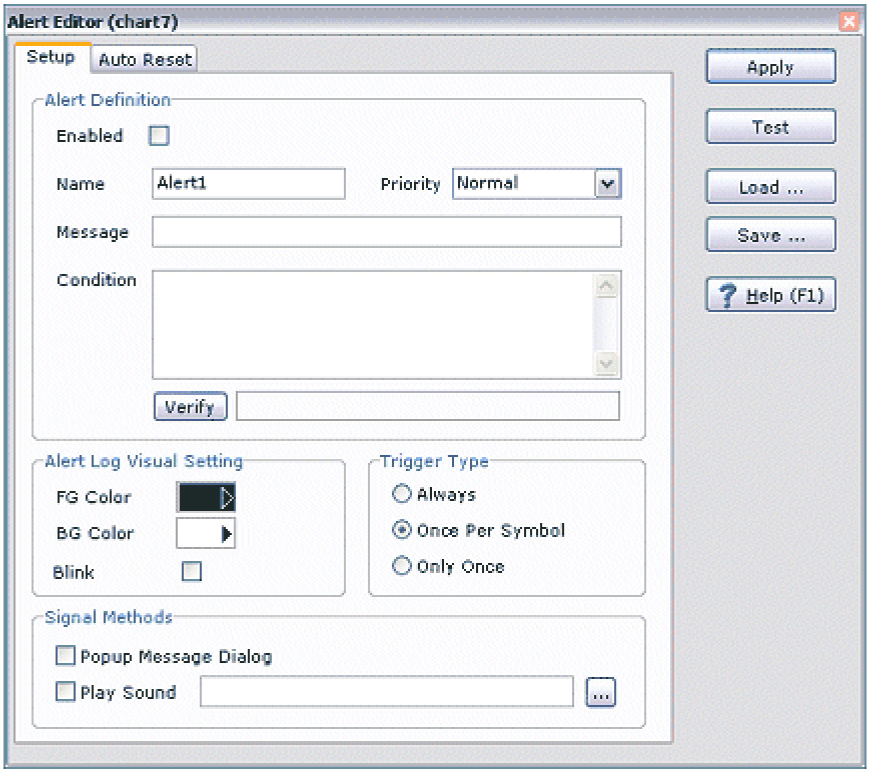
- Standalone alert windows. Alert window are discussed in ***Alert Windows Operation Guide*** (on page 419).
- Centralized alert tracking that tracks alerts in different windows, the topic of this section.

NeoTicker®'s alert tracking model is built on flexible alert object design. Each function window can host an alert list which contains one or more alerts at the same time. These alerts all have the same capability in generating alert signals to you. You can easily manage, edit and add alerts to each function window. A unified interface is provided such that you do not need to relearn how to add alerts from one function window to the next.

To add an alert to a function window, look for the **Add Alert** pop up menu item in the pop up menu. For example, if you are interested in adding alert to a Time Chart, then un-select any active object (by clicking in a blank area within the window) then right-click to bring up the general pop up menu. You can then add an alert by selecting the **Add Alert** item.

To edit an alert or manage the list of alerts, you can choose the **Alert List** item from within the same pop up menu.

Lets take a look at the Alert Editor:



Enabled □

Name Alert1

Priority Normal

Message

Condition

FG Color ■

BG Color □

Blink □

○ Always

● Once Per Symbol

○ Only Once

□ Popup Message Dialog

□ Play Sound

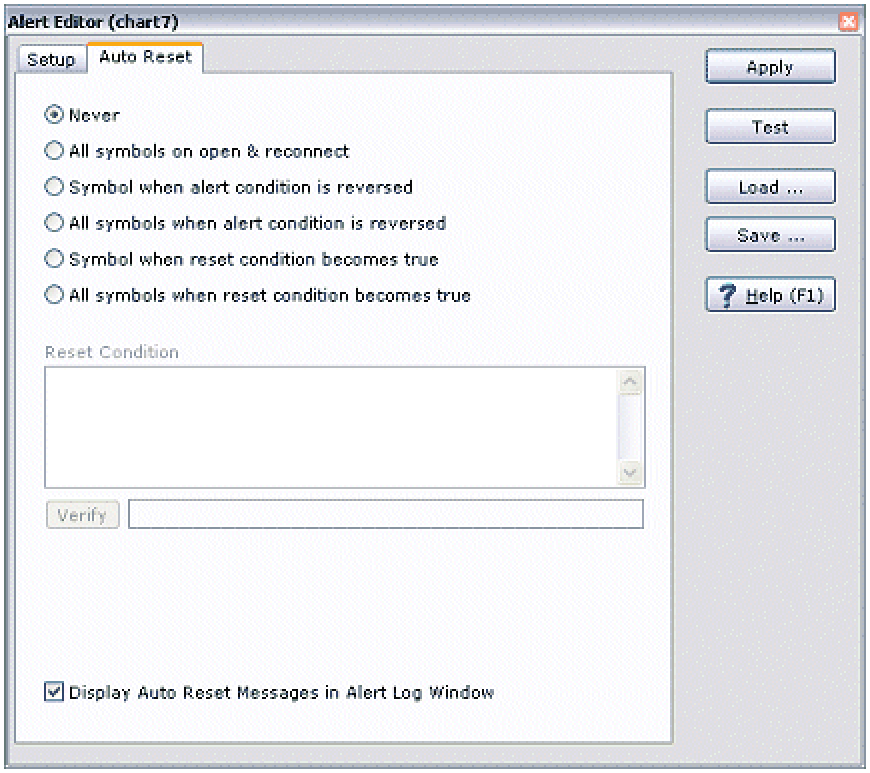
You will define the alert under **Alert Definition**. To define an alert:

- Provide a unique **Name** for the alert. An auto generated name is provided for you but you can always use a more descriptive name.
- Set the **Priority** of the alert so when the alert is triggered, it can remind you how important this alert is.
- Provide a descriptive **Message** for the alert in case you need some detail information about the alert. You can leave this blank.
- **Condition** is a simple conditional expression or formula describing the scenario that should trigger the alert. The condition definition can be as simple as something like a price comparison within a Quote Window (e.g. `Last > 50`), or as complex as a price pattern with many indicator values filtered within a Time Chart (e.g. `c(0) > c(1) and l(0) < l(1)` and `c(0) > I1.plot2(0)`).

  To learn more about how to define different conditions for different function windows, refer to the individual sections on this topic.

- Fill in the visual settings under **Alert Log Visual Setting**. You can leave this unchanged.
- Specify a **Signal Method**. This is how the alert notify you when it is triggered. You can choose to play a sound of your and/or show a pop up window to notify you.
- You have a choice on how the alert will behave when its condition is matched multiple times. You can set it to trigger all the time (the option **Always**), just once on a per symbol basis (the option **Once Per Symbol**), and just once only (the option **Only Once**).

Under the **Auto Reset** tab:



Alert Editor (chart7)

Setup | Auto Reset

● Never

○ All symbols on open & reconnect

○ Symbol when alert condition is reversed

○ All symbols when alert condition is reversed

○ Symbol when reset condition becomes true

○ All symbols when reset condition becomes true

Reset Condition

Verify

☑ Display Auto Reset Messages in Alert Log Window

You also have the option to choose whether the alert will reset itself on open and restart which is the default. If you choose the option **Never** then the state of the alert will be saved when you close the function window hosting it. And the state of the alert will be restored next time you open the same function window.

Refer to the ***Auto Resetting Alerts*** (on page 417) for more details.

To test the signal behavior, you can simply press the **Test** button to check if the Alert Log is displaying the alert the way you want. Change the signal options until it is doing exactly what you want.

You can save the alerts as templates and reuse them later through the **Load** and **Save** buttons.

As defining an alert requires some interactivity of edit and modification, the Edit Alert Window does not close on pressing the **Apply** button. You can close the window by clicking on the Window Close button at the upper right corner of the window, or, simply press the ESC key on the keyboard to close.

Signal Methods

You have two choices for how an alert signals you:

- **Popup Message Dialog** - A pop up window is opened to signal you the alert condition is met.
- **Play Sound** - A sound file is played.

If you have multiple monitors and want to use **Popup Message Dialog**, consider carefully where you want the pop up window to appear.

- **Monitor Center of Parent Form** - Pop up window will appear in center of the screen where the window that triggers the alert is located.
- **Primary Monitor Center** - Pop up window will appear in the center of the screen of your primary monitor.
- **Primary Monitor Lower Right Corner** - Pop up window will appear in the lower right corner of the screen of your primary monitor
- **Parent Form Center** - Pop up window will appear in the center of the window that triggers the alert.

Adding Alerts to Time Chart Window

To add an alert to Time Chart, choose **Add Alert** from the general pop up menu.

Now specify the name, general settings, and signal options. (If you do not know what are the general settings and signal options, refer to the previous section ***Basic Concept of Alerts*** (on page 407))

For the condition, Time Chart has the following items available for you to specify your alert condition.

Data

Data series can be referenced with the data object, followed by a dot and a field name. An option number in bracket refer to the value of certain bars ago.

The following fields are available to the data object:

Field (Long Form)	Field (Short Form)
Open	O
High	H
Low	L
Close	C
Volume	V
Tick	T
OpenInterest	OI

Formula language is not case sensitive thus you do not need to worry about typing the token in the wrong case.

So, to specify that you want to trigger the alert on close of first data series greater than 50, use the following expression.

```
Close > 50
```

the above is exactly the same as any one of the following lines

```
Close (0) > 50
C > 50
c (0) > 50
```

When you want to refer to the other data series, you can add the prefix of `Data#` or `D#`, where # is the order of the data series. Thus the following lines are the same

```
c (0) > 50
d1.c (0) > 50
data1.c (0) > 50
data1.close (0) > 50
```

Thus when you want to refer to the second data series, you can do as follows.

```
data2.close (0) > 50
d2.c > 50
data2.c > 50
```

Now say we want to specify that we want the first data series to have its close greater than the previous close while the second data series is having a lower high, then the condition looks like the following.

```
c(0) > c(1) and d2.h(0) < d2.h(1)
```

Indicator

To refer to indicator values, you will address them with `Indicator#` or `I#`, where # is the order of the indicator. For example, to refer to the last value of the first plot (or the only plot) of the first indicator defined, it looks like any of the following.

```
Indicator1 (0)
I1 (0)
I1.plot1 (0)
Indicator1.p1 (0)
Indicator1.plot1 (0)
```

The `plot1` or `p1` from the examples above are the suffix specification for multiple plot values. If you need to refer to the second plot of an indicator, use `plot2`.

Now lets look at this example. If you have defined the Moving Averages 3 Lines indicator as the first indicator in your chart, and you want to create an alert of the crossover of the first 2 moving averages, then the condition will look like this,

```
I1 (1) < I1.plot2 (1) and I1(0) > I1.plot2 (0)
```

Drawing Tools

To access the value of a trend line, you can use the token `Trendline` or `TL`. Since time chart is capable of handling multiple data series and indicators, by default using the trend line tokens on its own it will inherit the time frame of the first data series, thus you can check for a TL break out from below as follows.

```
C (1) < TL (1) and C (0) > TL (0)
```

If you need a quick and dirty comparison which you know the trend line is above the prices you are tracking, then it can be as simple as follows.

```
C(0) > TL (0)
```

Now if you want to compare the trend line against another data series, then you have to add the data series identity as the suffix to `TL` such that the time frame of the trend line can be correctly matched to the data series.

For example, if you would like to refer to the third data series against the second trendline, you can do as follows.

```
d3.close (0) > TL2.d3 (0)
```

If you would like to compare a trend line against an indicator, then you can do as follows.

```
Trendline3.I2 (0) > I2.plot4 (0)
```

To access the value of a horizontal line, you can use the token `HorizontalLine` or `HL`. Its formula usage is exactly same the `Trendline`.

Adding Alerts to Quote Window

To add an alert to Quote Window, choose **Add Alert** from the pop up menu.

Now specify the name, general settings, and signal options. (If you do not know what are the general settings and signal options, refer to the previous section ***Basic Concept of Alerts*** (on page 407))

For the condition, Quote Window has all its columns available for your usage. When the column name has a percentage sign (%) in its name, it is replaced with Pct instead. For example, the field **range%** is available in the formula as `RangePct`.

For example you want to check if your position for a specific symbol has reached its target price, then you can specify the following condition.

```
PosSize <> 0 and PosFromTarget <= 0
```

`PosSize` is a field you can enter data to the quote window through the Edit Position Window.

`PosFromTarget` is a pre-defined field that is calculated from the last price and the `PosTarget` price that you have entered using the Edit Position Window.

You can use formula columns in an alert condition. For example, if you have defined a formula column called **MovDiff**, then you can use this column in your condition.

```
MovDiff > 0
```

Remember that the alerts defined in a quote window will be applied to all the symbols in the quote window and trigger its alert according to the specific condition of each symbol individually. Thus if you set the alert to trigger only once, many signals after the first generated by same or other symbols will not be reported. You may want to use **Once Per Symbol** trigger type for quote window alerts.

Managing Alerts

To manage alerts, you can choose **Alert List** from the pop up menu of the function window.

The Alert List Window will be displayed with the latest status of each alert reported.

To quickly enable or disable an alert, click the check box.

To edit an alert, simply double click on the alert item.

To delete an alert, select the alert from the list, then press the **Delete** button.

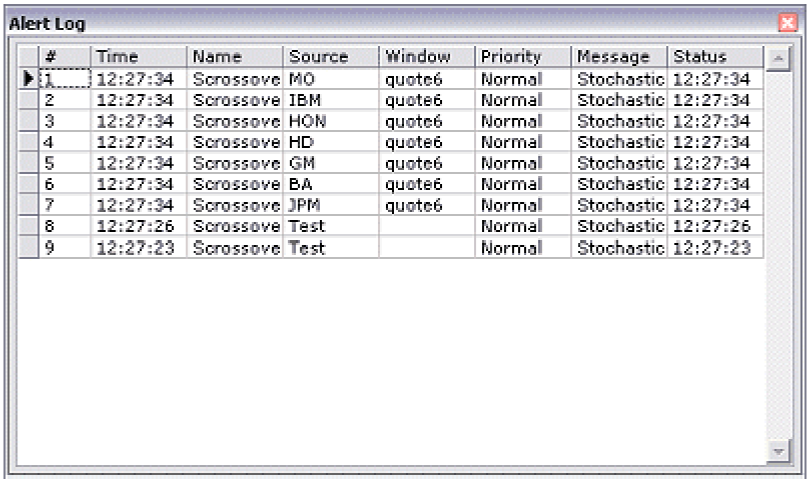
To add a new alert from the alert list, press the **Add** button.

To reset an alert, select the alert from the list, then press the **Reset** button. The alert will be set to the state of not triggered.

To reset all the alerts, press the **Reset All** button.

Alert Log Window

Alert Log window is the centralized alert display. You can open the Alert Log window by choosing **Program>Alert Log** from main window.



	#	Time	Name	Source	Window	Priority	Message	Status
▶	1	12:27:34	Scrossove	MO	quote6	Normal	Stochastic	12:27:34
	2	12:27:34	Scrossove	IBM	quote6	Normal	Stochastic	12:27:34
	3	12:27:34	Scrossove	HON	quote6	Normal	Stochastic	12:27:34
	4	12:27:34	Scrossove	HD	quote6	Normal	Stochastic	12:27:34
	5	12:27:34	Scrossove	GM	quote6	Normal	Stochastic	12:27:34
	6	12:27:34	Scrossove	BA	quote6	Normal	Stochastic	12:27:34
	7	12:27:34	Scrossove	JPM	quote6	Normal	Stochastic	12:27:34
	8	12:27:26	Scrossove	Test		Normal	Stochastic	12:27:26
	9	12:27:23	Scrossove	Test		Normal	Stochastic	12:27:23

Alert Log window can save its display position, visible state and all related display settings when you exit NeoTicker®. Thus you can position it on the screen for tracking alerts where no function windows will block your view.

You can easily modify the look and feel of the alert log window by choosing the columns you want to display through the pop up menu.

You can adjust the Font size and style using **Font** from the pop up menu.

You can also re-arrange the column order by drag and drop of the columns to the desired position.

When an alert event happen, it will be logged down by the alert log window. A summary of the alert is also saved to log files for the day in the `AlertLog` directory.

If the alert is traceable back to a particular function window, you can double click on the alert entry to show that particular function window at once.

If you no longer need the alert log window on your screen, just close the window. The alert events will still be tracked for you and you can re-open it any time to inspect your current alert entries.

Double Click Action

When you double click on an alert, it will open the window that triggers the alert. You can customize this behavior by choosing **Double Click** from the pop up menu. The following choices are available:

- **Open Alert Window** - Default action, opens the window that triggers the alert.
- **Maximize Alert Window** - Opens the window that triggers the alert and maximizes the window.
- **Send to Chain** - Sends the symbol that triggers the alert to chain.
- **Launch Macro** - Launches a macro. Macro is user defined script that carries out certain actions.

Double Click Macro

If you set double click action to **Launch Macro**, when you double click an alert, the alert window will carry out the following actions:

1. Makes the window that triggers the alert the active window.
2. Launch the macro. Information about the alert are sent to the macro as parameters.

A macro is simply a script file that carries out certain actions. It is an OLE automation script (see ***OLE Automation*** (on page 653)) that has additional parameters passed to it that are alert related.

You can easily write macros yourself. To tell alert log window to use an macro, choose **Specify Macro** from the pop up menu, and select the script file.

The following is a VBScript that implements restore action. When an alert is double clicked, it will restores the window that triggers the alert to its normal size if the window is currently maximized or minimized.

```
' ALERT LOG Macro Example
' Command Line Parameters from Alert Log
' 0 - datetime
' 1 - name
' 2 - source
' 3 - window
' 4 - priority
' 5 - mesg
' 6 - status

set nt = createobject ("NeoTicker.App")

set fw = nt.FunctionWindowInActiveGroup (WScript.Arguments (3))

fw.restore

set fw = nothing
set nt = nothing
```

The script first calls `createobject` to create an instant of application, and assign the object to the variable `nt`.

The next step calls `nt.FunctionWindowInActiveGroup` to retrieve the function window that triggers the alert by the window name (third argument), and assigns the function window object to the variable `fw`.

The next step calls `fw.restore` to restore the function window to its normal size.

Finally, the variables `fw` and `nt` are assigned with nothing to free any used memory.

Auto Resetting Alerts

You can set up reset conditions for alerts such that when the reset condition is met, an alert is reset and can be triggered later.

You do not have to worry about all symbol vs. single symbol for chart alert. The all symbol vs. single symbol reset condition is for quote window when you want to make distinction between alerts for individual symbols and alerts for all symbols. For charts, you can simple use the single symbol version.

All Symbols on Open & Re-connect

The alert is reset when the window is opened or when a re-connection to the server has happened.

Symbol vs. All

The idea is the alert can be triggered by multiple symbols. So auto reset can either reset a particular symbol, or all the symbols for the alert.

After the alert is reset, it can be re-triggered.

Condition Reversed vs. Reset Condition Becomes True

Condition Reversed means the alert condition changes from true to false.

If the condition is very sensitive to price movement, condition reversed can happen too frequently, resetting and triggering the alert a lot.

Reset Condition can solve this problem. Reset condition is another formula. The alert is reset only when the alert condition becomes true.

For example, if the alert is `last > 50`, if you use one of the condition reversed reset options, the alert can get triggered a lot if the price keep moving below and above 50. Instead, it is more desirable to have some lag before the alert is reset. For example, you can specify such that the alert is reset only when the last price goes below 49.75. The reset condition would then be `last < 49.75`.

Display Auto Reset Message in Alert Log Window

If checked, when an auto reset happens, the alert log window will display the auto reset message.

Example

For example, suppose your alert has been triggered by MSFT and IBM. Here is the behavior:

If **Symbol on Condition Reversed** is checked, suppose the alert condition changed from true to false for IBM, then the alert is reset for IBM. If the alert condition is true again for IBM, the alert will be triggered again for IBM. MSFT will not be triggered.

If **All symbols on Condition Reversed** is checked, suppose the alert condition changed from true to false for IBM, then the alert is reset for IBM and MSFT. IBM and MSFT will both be triggered when the alert condition becomes true again.

If **Symbol on Reset Condition True** is checked, suppose the reset condition becomes true for IBM, then the alert is reset for IBM. MSFT will not be triggered.

If **All symbols on Reset Condition** True is checked, suppose the reset condition becomes true for IBM, then the alert is reset for IBM and MSFT.

Alert Windows Operation Guide

An alert window gives you an audio warning when a specified situation happens.

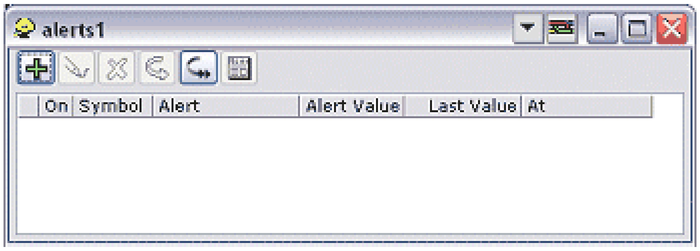
Alert window offers standalone price alert services. NeoTicker® also has centralized functional window based alerts. For these types of alerts, refer to ***Alert Tracking Tutorial*** (see "Alert Tracking" on page 407).

Creating an Alert Window

To create an alert window:

- Press the alert window button 💡 to create an alert, or
- Choose **Window>New>Alerts** from the main window.

A new alert window is blank and does not have alert in it.



At the top of the window are six buttons:

➕	Adds a new alert
✎	Edits a selected alert
✖	Deletes a selected alert
↻	Resets a selected alert
↻»	Resets all alerts
📰	Shows the news of a streaming news alert

The alert window has a pop up menu. You can use the pop up menu to invoke the functions mentioned above. You can also send a symbol to a new window, template, chain or another window.

Following are the descriptions of the columns in the alert window:

On Whether the alert is enabled. An alert can go off only when it is enabled

Symbol The symbol that the alert is monitoring

Alert The type of the alert

Alert Value The value of the alert. If the alert is a price alert such as price above or price below, then the value is a price. If the alert is a volume above alert, then the value is the volume. If the alert is a tick pattern alert, then the value is the tick pattern. If the alert is a percentage alert such as up tick percentage or down tick percentage, then the value is the percentage. If the alerts is news, then the value is not defined

Last Value The last value of the security. What the value represents depends on the alert type

At At what time the alert happens. The At field also shows status of some alerts. For buy stop and sell stop, the At field also displays whether the stop has activated. For up tick and down tick percentage, the At field also displays whether the alert is ready

Adding an Alert

To add an alert, press the ➕ button on the alert window.

After you fill up the alert parameter, press the **OK** button. The alert will show up in the alert window. You can add multiple alerts of different symbols.

Press the **Quote** button to quickly fill in some fields.

When an alert happens, you will hear a sound and the alert will change to yellow.

Many alert types are available, see ***Alert Types*** (on page 423) for explanation.

Editing Alert

You can edit an alert directly in the alert window. Simply click on a cell and edit. Alternatively, you can edit an alert using the alert editor. To edit an alert using the alert editor, select the alert and press the ✎ button, or choose **Edit Alert** from the pop up menu.

You cannot edit an alert that has gone off. You must reset it first. If an alert has gone off you can only enable or disable the alert.

Deleting Alert

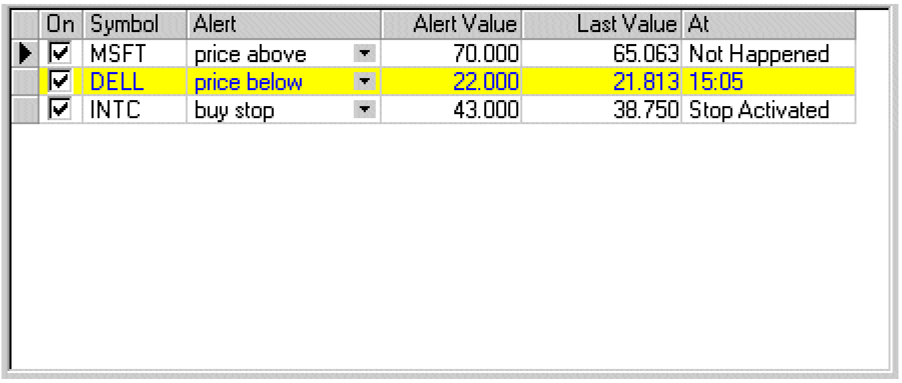
You can delete an alert by selecting the alert (clicking in the left most box) and press the ✖ button. Alternatively, you can delete an alert using the pop up menu.

Resetting Alert

After an alert happens, you can reset the alert such that it will warn you the next time. To reset an alert, press the ↶ button. If you want to reset all alerts, press the ↶ button.

When Alert Goes Off

When an alert goes off, an audio sound is played and the alert will change to yellow. In the figure below, the alert for DELL has gone off.



	On	Symbol	Alert	Alert Value	Last Value	At
▶	☑	MSFT	price above	70.000	65.063	Not Happened
	☑	DELL	price below	22.000	21.813	15:05
	☑	INTC	buy stop	43.000	38.750	Stop Activated

Once an alert has gone off, you cannot edit it until you reset the alert. You can only disable or enable an alert that has gone off. For news alerts, you can double click the alert, press the ▤ button or choose **Show News** from the pop up menu to read the news.

Alert Types

Alert type is set in the alert editor. You can select an alert and press the button to change the alert type.

The alert window supports many different types of alerts. In general, an alert can react to price, volume, tick pattern, tick percentage, or news. Here is the explanation of the supported alert types:

Alert Type	Meaning
Price Above	A price above alert goes off when the last traded price of a security is above the specified price.
Price Below	A price below alert goes off when the last traded price of a security is below the specified price.
Buy Stop	A buy stop alert goes off when a security trades pass and above the specified price. For example, if a security has a buy stop at $70, the alert will go off when a trade happens below or at $70, then followed by a trade happens above $70. When a trade happens below or at the specified price, the buy stop is said to be activated. A *stop activated* message is shown in the **At** field in the alert window.
Sell Stop	A sell stop alert goes off when a security trades pass and below the specified price. For example, if a security has a sell stop at $70, the alert will go off when a trade happens above or at $70, then followed by a trade happens below $70. When a trade happens above or at the specified price, the sell stop is said to be activated. A *stop activated* message is shown in the **At** field in the alert window.
Volume Above	Alert goes off when volume is above the specified value.
Tick Pattern	Tick pattern alert. See section below.
Up Tick Percentage	An up tick percentage alert goes off when the percentage of up ticks on the current trading day goes above the specified percentage.
Down Tick Percentage	An down tick percentage alert goes off when the percentage of down ticks on the current trading day goes above the specified percentage.
News Alert	A news alert goes off when an incoming news headline contains the security symbol that you have previously specified.

Tick Pattern

Alert window keeps track of each tick of all securities and constructs tick patterns for these securities. The tick pattern goes off when the specified tick pattern matches the tick pattern of the security.

Alert window supports two types of tick patterns: Up/Down/Side ticks and Up/Down ticks. In an Up/Down/Side tick pattern, the plus sign (+) represents an up tick, the minus sign(-) represents a down tick and the equal sign(=) represents a side tick. In an Up/Down tick pattern, there is no side tick. When the price of a trade is equal to the previous trade, the sign of the tick is the same as the previous tick.

Tick pattern alert supports both type of tick patterns. When you specify an Up/Down/Side tick alert, you will use all three signs, e.g. +++=-. When you specify an Up/Down tick alert, you will use only the plus and minus signs, e.g. +++---.

Up Tick and Down Tick Percentage Caution

Up tick percentage and down tick percentage depends on ticks that have already happened. If you start NeoTicker® in the middle of a trading day, the up/down tick percentage alerts are not ready until NeoTicker® finishes going through all the ticks that have happened. If an up tick percentage alert is not ready, a "Not Ready" message is shown in the **At** field in the alert window.

News Alert

A news alert goes off when an incoming news headline contains the security symbol that you have previously specified. When a news alert goes off, you can bring up a news window and read the news by double clicking on the alert. Alternatively, you can bring up the news window by pressing the ▤ button or using the pop up menu while the news alert is selected.

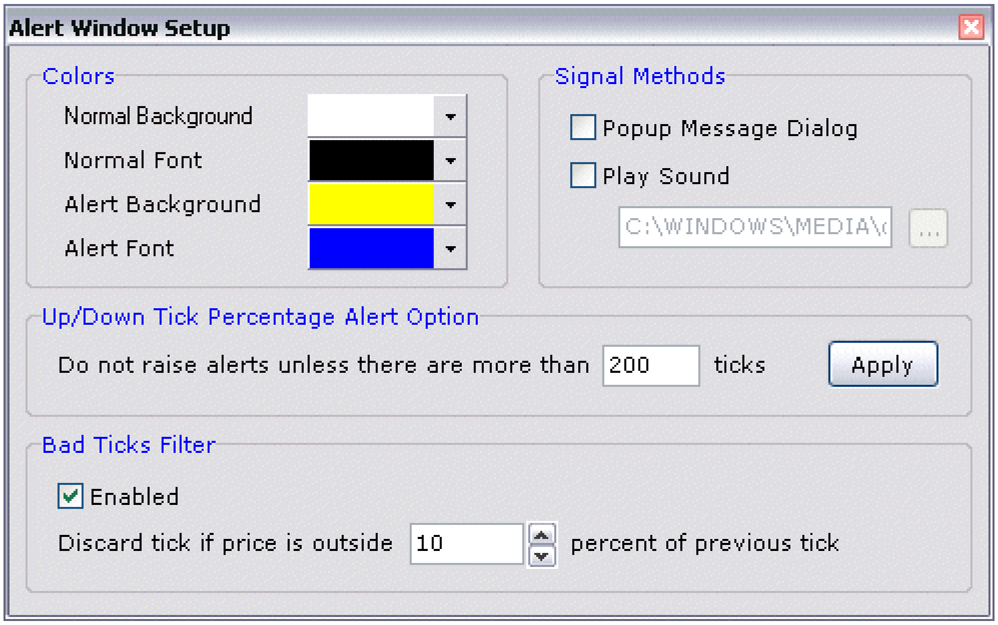
Even after a news alert goes off, the alert continues to receive news and the news window will be updated accordingly.

News articles in an alert does not save with the alert window. A news alert is reset after it has been loaded from file.

Your data vendor must support streaming news and you must have news service turned on before this alert type can work.

Alert Window Setup

To set up alert window, choose **Setup** from pop up menu.



Alert Window Setup

Colors

Normal Background ⬜

Normal Font ⬛

Alert Background 🟨

Alert Font 🟦

Signal Methods

□ Popup Message Dialog

□ Play Sound

C:\WINDOWS\MEDIA\(

Up/Down Tick Percentage Alert Option

Do not raise alerts unless there are more than 200 ticks

Bad Ticks Filter

☑ Enabled

Discard tick if price is outside 10 percent of previous tick

Alert Sound

Alert window uses the sound files in the standard MS Windows installation as well as the sound files in the `Media` folder in NeoTicker®'s installation (e.g. `C:\Program Files\TickQuest\NeoTicker3\Media`). You can install your own sound files in the `Media` folder to have more sound choices available. The sound files must be in .wav format

Up/Down Tick Percentage

The minimum number of ticks cumulated in a day for a single symbol before tick percentage alerts become ready. This minimum value prevents problems caused by the instability due to low trading volume.

Bad Tick Filter

Alert window implements its own bad tick filter separated from the main bad tick filter, because alerts are more sensitive to bad ticks than other function windows. By default, alert window's bad tick filter discards all ticks that have prices that are 10% outside of previous price. It is possible for the bad tick filter to malfunction in a gap up or gap down situation at the beginning of a trading day. In this case, you can reset the bad tick filter by resetting all the alerts.

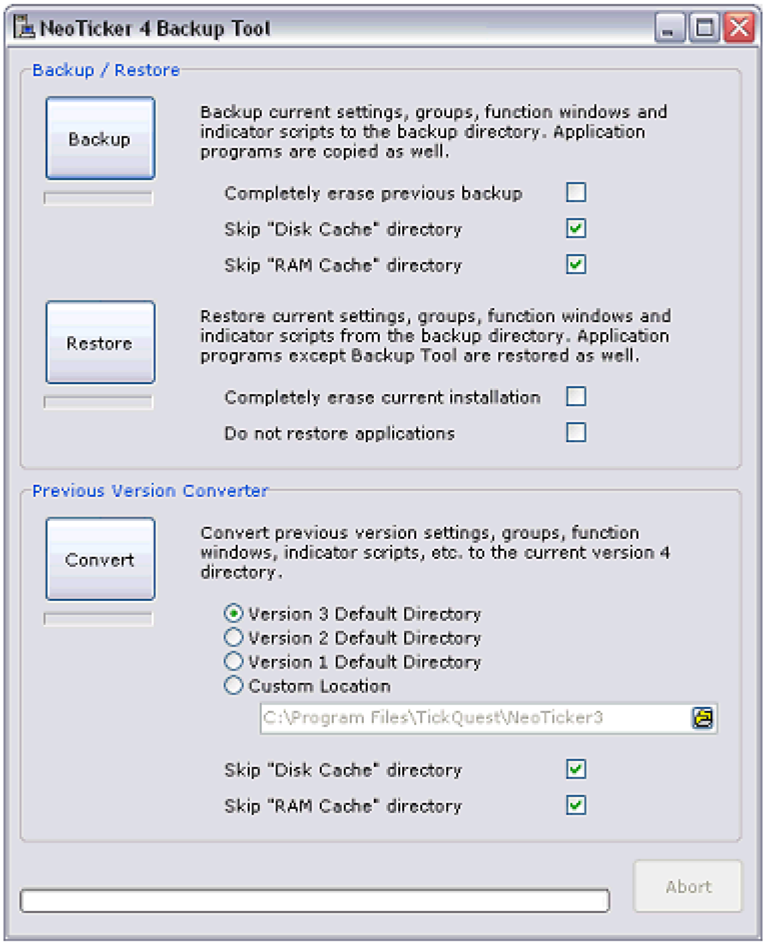
Auxiliary Tools

Auxiliary tools are tools that are designed to work with NeoTicker® but is not part of NeoTicker® itself. Some auxiliary tools are come with NeoTicker® (e.g. Scan Workshop). Some requires installation (e.g. Equity Monaco).

In general, auxiliary tools can be launched from the **Tool** menu in main window.

Backup Tool

NeoTicker® comes with a backup tool that helps you manage NeoTicker® files. The backup tool is a separate utility program in the NeoTicker® installation directory. To open the backup tool, choose **Tool>Backup Tool** from main window, or launch the program `NTBackupTool` in the NeoTicker® installation directory.



NeoTicker 4 Backup Tool

Backup / Restore

Backup

Backup current settings, groups, function windows and indicator scripts to the backup directory. Application programs are copied as well.

Completely erase previous backup □

Skip "Disk Cache" directory ☑

Skip "RAM Cache" directory ☑

Restore

Restore current settings, groups, function windows and indicator scripts from the backup directory. Application programs except Backup Tool are restored as well.

Completely erase current installation □

Do not restore applications □

Previous Version Converter

Convert

Convert previous version settings, groups, function windows, indicator scripts, etc. to the current version 4 directory.

● Version 3 Default Directory
○ Version 2 Default Directory
○ Version 1 Default Directory
○ Custom Location

C:\Program Files\TickQuest\NeoTicker3

Skip "Disk Cache" directory ☑

Skip "RAM Cache" directory ☑

Abort

Backing Up

To backup NeoTicker®, press the **Back up** button. The backup tool creates a directory call `Backup` and copy everything in the NeoTicker® installation directory to the backup directory.

Because all the files are backed up, including the executables, backup should be performed before you make any major change to NeoTicker®, such as applying a patch, etc.

Restoring

To restore from a backup, press the **Restore** button. The backup tool will copy everything in the `Backup` directory to the current NeoTicker® installation. NeoTicker® installation is not erased before restoring. This means if you have created a new file since last backup, the new file will not be erased.

This is the desirable behavior when the new file does not affect the operation of NeoTicker®.

Optionally, you can check the **Completely erase current installation** option before pressing the **Restore** button. All files will be erased before the restoring operation begins.

If you check **Do not restore applications**, then only data and settings are restored, not programs. This is useful when you want to restore data but do not want to revert back to an old version of NeoTicker®.

Previous Version Converter

Use this button only if you are converting NeoTicker® version 1, 2, 3 files to version 4.

Different versions of NeoTicker® are installed under different directories. So if you are upgrading from V3 to V4 for example, you will want to copy your V3 settings to V4.

Backup Tool can detect the default installation directory of previous NeoTicker® versions (e.g. C:\Program Files\TickQuest\NeoTicker3 for version 3). However, if you installed previous version of NeoTicker® in a different directory, you must choose the **Custom Location** option and specify the location manually.

If you use Internet feed with historical data, or IDS with satellite feed, you should skip the Disk/RAM Cache copy because NeoTicker® will retrieve fresh copy of historical data from your data vendor/IDS. If you collect historical data yourself, you should copy Disk/RAM Cache.

Procedure to transfer files from V4:

1 Open backup tool.

2 Specify the version of NeoTicker® you are converting from, or a custom location if you did not install previous version of NeoTicker® in default directory.

3 Choose the skip options. For Internet feed/IDS, skip copying. Otherwise, perform copying.

4 Press **Convert** button.

Data Tool

NeoTicker® Data Tool is a companion application that comes with NeoTicker®.

This tool is intended for data feed users who want to take an active role in maintaining the disk cache.

NeoTicker® data tool provides the following services:

- Archiving disk cache data
- Renaming symbols in the disk cache (useful for ticker changed and for handling futures data symbology differences).
- Batch removal of data (useful to repair large scale data corruption probably due to power failure).
- Installing data from CD.

What is the Disk Cache

Disk cache stores data from an Internet data vendor locally on your hard disk. Because disk cache stores the data locally, the disk cache significantly reduces the amount of historical data requests from the data vendor and speeds up data access a lot.

Also, the disk cache can store much longer period of historical data than what your data vendor is willing to provide (most data vendors only provide less than two months of minute data and a few days of tick data). Thus allowing you to work on data that is long expired on your data vendors server.

Disk cache is automatically enabled in NeoTicker®. You won't notice it.

Starting the NeoTicker® Data Tool

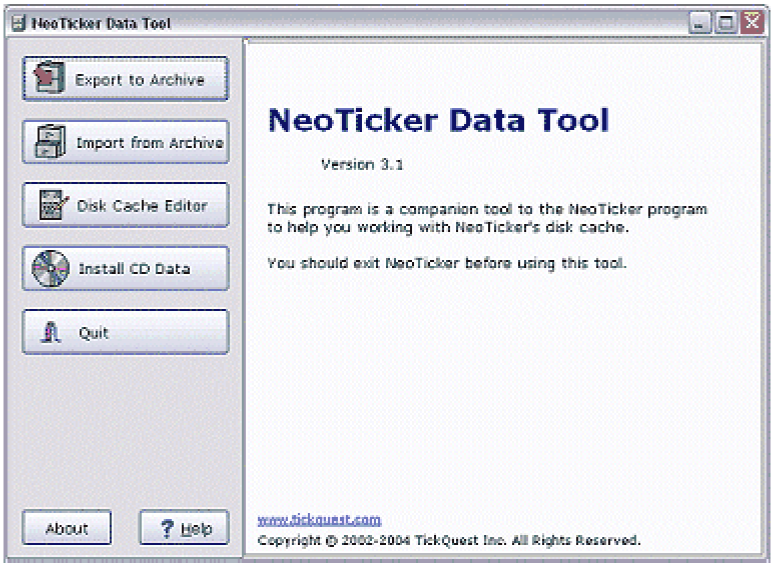
NeoTicker® Data Tool is a separate program. You can start the Data Tool from the Windows start menu.

You need to exit NeoTicker® before starting Data Tool to avoid disk cache corruption.

To start the Data Tool, from Windows **Start** button, choose **Start>Programs>TickQuest>DataTool**.

Using the Data Tool

Data Tool is mainly wizard driven. To use a function, simply press the button on the Data Tool window. In most cases, a wizard will be launched to help you. Follow the direction the wizard gives you to complete the task.



NeoTicker Data Tool

Version 3.1

This program is a companion tool to the NeoTicker program to help you working with NeoTicker's disk cache.

You should exit NeoTicker before using this tool.

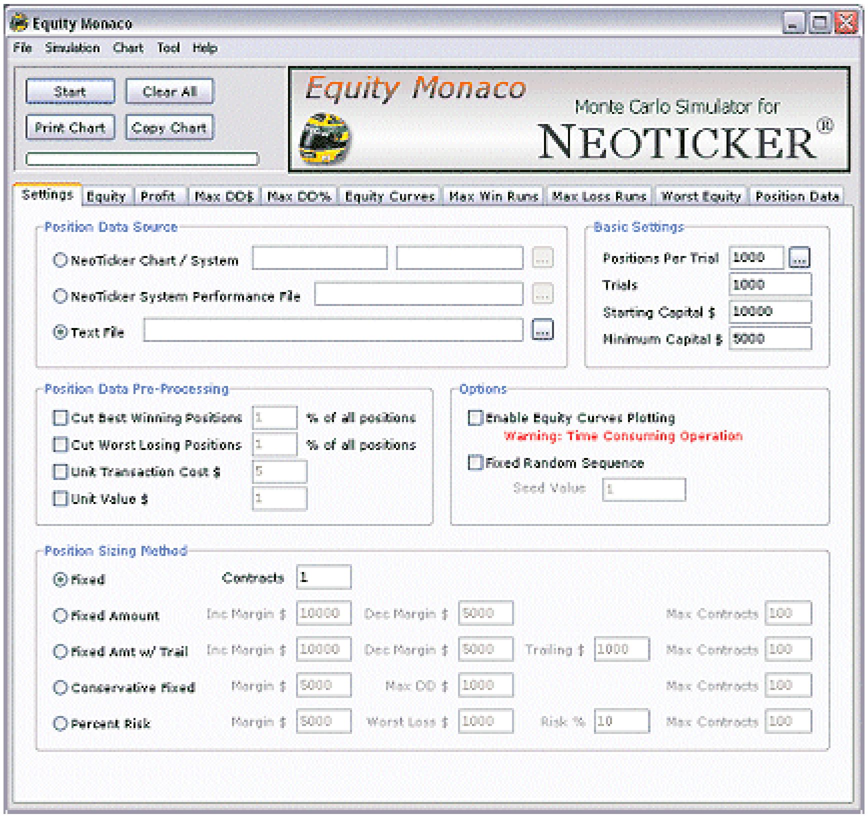
www.tickquest.com

Copyright © 2002-2004 TickQuest Inc. All Rights Reserved.

The data tool comes with its own set of online documentation. Should you need any help, simply press the **Help** button or the F1 key.

Equity Monaco

Equity Monaco is TickQuest's Monte Carlo Simulation tool that seamlessly integrates with NeoTicker®. Equity Monaco is a free tool, you can simply download it from ***here*** http://www.tickquest.com/EquityMonaco.



Position Data Source

○ NeoTicker Chart / System

○ NeoTicker System Performance File

● Text File

Basic Settings

Positions Per Trial	1000
Trials	1000
Starting Capital $	10000
Minimum Capital $	5000

Position Data Pre-Processing

□ Cut Best Winning Positions	1	% of all positions
□ Cut Worst Losing Positions	1	% of all positions
□ Unit Transaction Cost $	5	
□ Unit Value $	1	

Options

□ Enable Equity Curves Plotting

Warning: Time Consuming Operation

□ Fixed Random Sequence

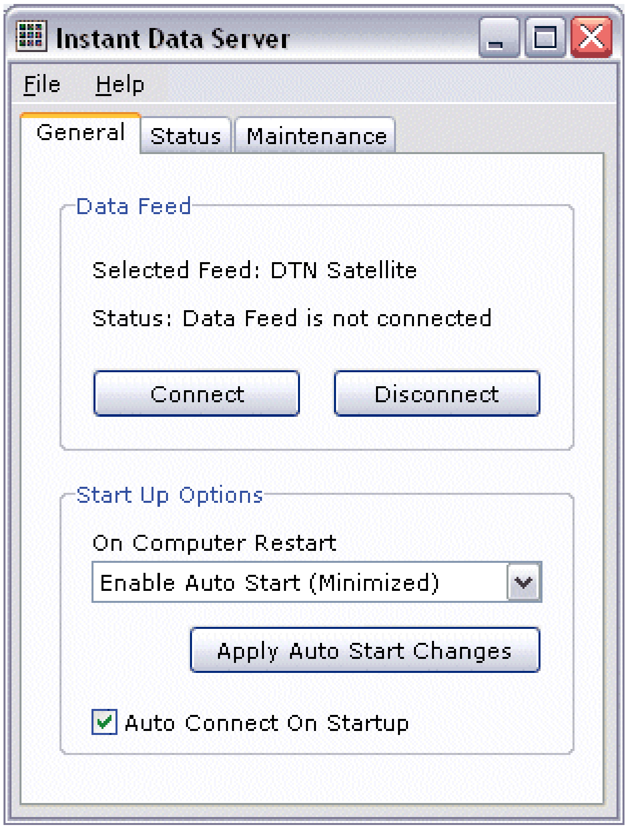
Seed Value 1

Position Sizing Method

● Fixed	Contracts	1						
○ Fixed Amount	Inc Margin $	10000	Dec Margin $	5000			Max Contracts	100
○ Fixed Amt w/ Trail	Inc Margin $	10000	Dec Margin $	5000	Trailing $	1000	Max Contracts	100
○ Conservative Fixed	Margin $	5000	Max DD $	1000			Max Contracts	100
○ Percent Risk	Margin $	5000	Worst Loss $	1000	Risk %	10	Max Contracts	100

Instant Data Server

Instant Data Server (IDS) for connecting NeoTicker® to satellite data feeds. For more information, refer to IDS's own documentation.



Selected Feed: DTN Satellite

Status: Data Feed is not connected

On Computer Restart: Enable Auto Start (Minimized)

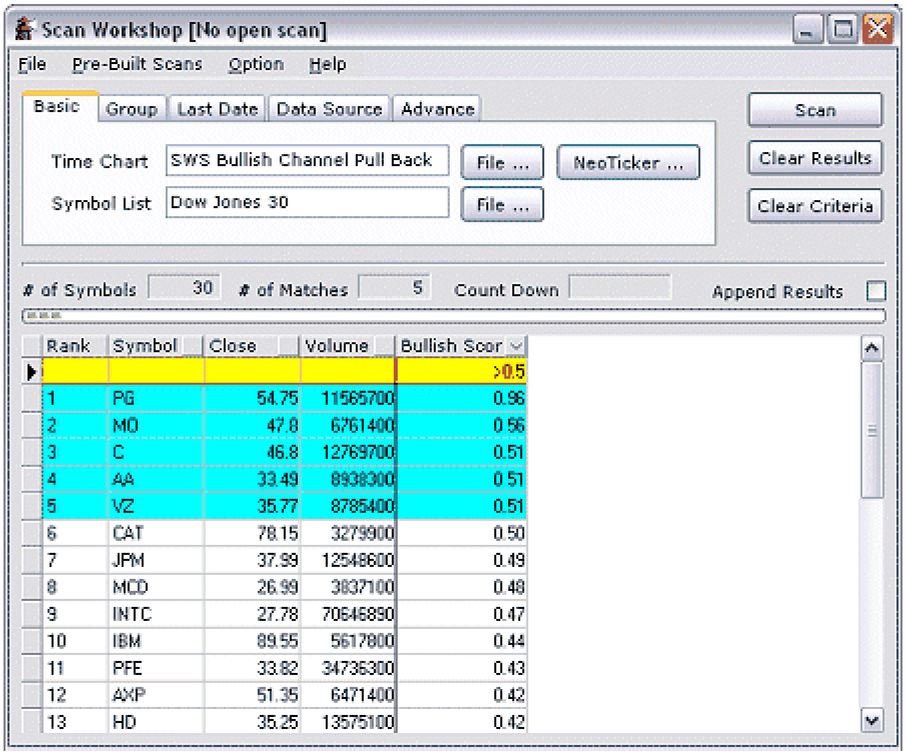
☑ Auto Connect On Startup

Scan Workshop

Scan Workshop is a complete solution for data mining that uses NeoTicker® as the engine. Because Scan Workshop is based on NeoTicker®, it allows you to scan the really useful market formations such as channel pullbacks and wedge breakouts. You can further refine the result using custom query.

This level of scanning is simply not possible with web-based scanning or scanner that does not have a sophisticate engine like NeoTicker®.

To find out how to use Scan Workshop, refer to Scan Workshop's own tutorial and documentation.



Time Chart: SWS Bullish Channel Pull Back

Symbol List: Dow Jones 30

# of Symbols: 30

# of Matches: 5

Rank	Symbol	Close	Volume	Bullish Scor
				>0.5
1	PG	54.75	11565700	0.96
2	MO	47.8	6761400	0.56
3	C	46.8	12769700	0.51
4	AA	33.49	8938300	0.51
5	VZ	35.77	8785400	0.51
6	CAT	78.15	3279900	0.50
7	JPM	37.99	12548600	0.49
8	MCD	26.99	3837100	0.48
9	INTC	27.78	70646890	0.47
10	IBM	89.55	5617800	0.44
11	PFE	33.82	34736300	0.43
12	AXP	51.35	6471400	0.42
13	HD	35.25	13575100	0.42

Bid and Ask Support

Introduction

NeoTicker® supports bid and ask analysis. You can use bid and ask information as if they are normal prices. For example, bid and ask charting is possible. You can use bid and ask data in charts, quotes, scanners, quote window formula, etc.

You can construct bid and ask symbols by adding !A and !B to a regular symbol. For example, the symbol for MSFT bid is `MSFT!B` and the symbol for MSFT ask is `MSFT!A`.

Data Vendor Considerations

Bid and ask data are not normal data. There are some issues you need to be aware of when you use bid and ask data.

Bid and Ask Data are Tick Data

Data vendors do not offer historical bid and ask data at minute level. Therefore, all bid and ask data are constructed from ticks. This is true for all time frames. You should use bid and ask prices with the same caution as tick data. If you use an Internet data vendor, it will take time to download the bid and ask ticks from the data vendor.

Not all Data Vendor Supports Historical Bid and Ask

The following data vendors provide historical bid and ask data:

- Quote.com
- NAQ
- eSignal
- IQFeed
- DTN Satellite

The following data vendors do not provide historical bid and ask data:

- Interactive Brokers
- PFG Best Direct
- myTrack
- All DDE feed

For Internet feed, if your data vendor does not provide historical bid and ask data, you can still chart bid and ask, but the bars will need to build up over time. NeoTicker® will auto flush the bid ask data to your local disk so they will be available later.

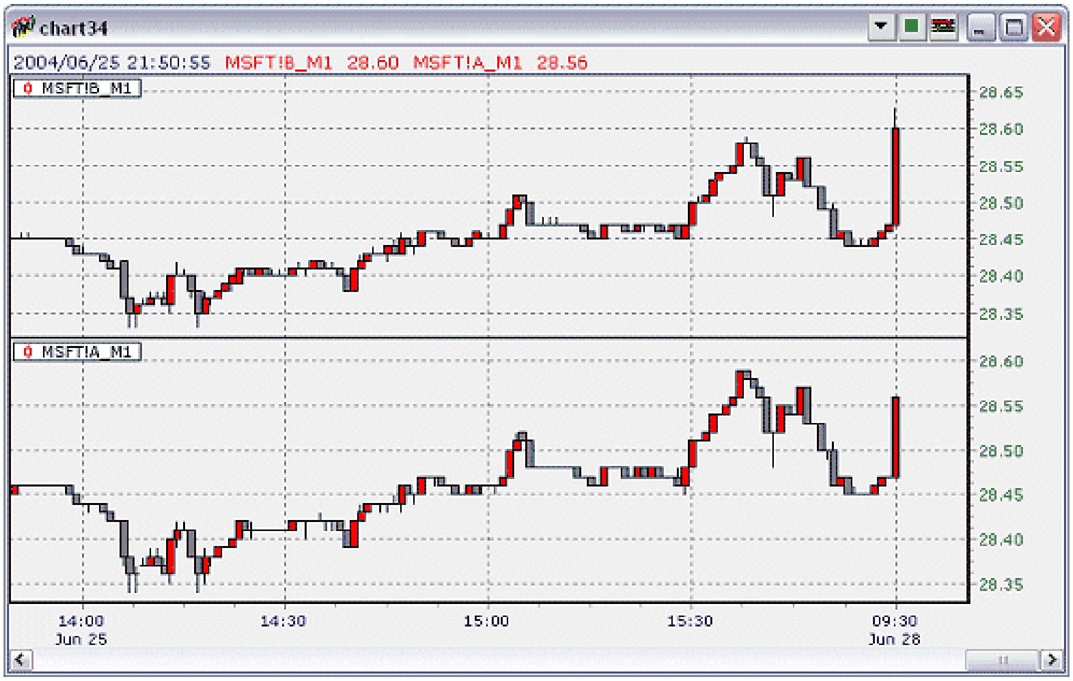
Example: Bid and Ask Charts

Open a chart. Add a data series. Type `MSFT!B` for the symbol and choose a 1-minute time frame.

This will add a data series to the chart with 1-minute bid data. The 1-minute bid data is constructed using the bid ticks. So it may take some time to construct the chart.

Add another data series. Type `MSFT!A` for the symbol and choose a 1-minute time frame.

This will add a data series to the chart with 1-minute ask data.



Cache Manager - Introduction

NeoTicker® uses cache to improve data access performance. The cache are seamlessly integrated with the application. You only have to concern with the cache occasionally. There are two level of cache in NeoTicker®: RAM Cache and Disk Cache. RAM Cache stores data in RAM for ultra fast data access. Symbols that are in the RAM Cache can be accessed almost instantaneously. Data stored in RAM Cache is not permanent. An auto saving feature is provided to flush RAM Cache data into Disk Cache for permanent storage.

Disk Cache stores data in your local hard drive. Retrieving data from the Disk Cache is much faster than from the historical server of an Internet feed. Disk Cache also provides unlimited historical data storage.

Cache Manager is the tool to manage cache data. You can access Cache Manager by choosing from the main window, **Manager>Cache**.

For information about disk cache, see ***Cache Manager - Disk Cache*** (on page 443).

For information about RAM Cache, see ***Cache Manager - RAM Cache*** (on page 461).

Cache Manager - Disk Cache

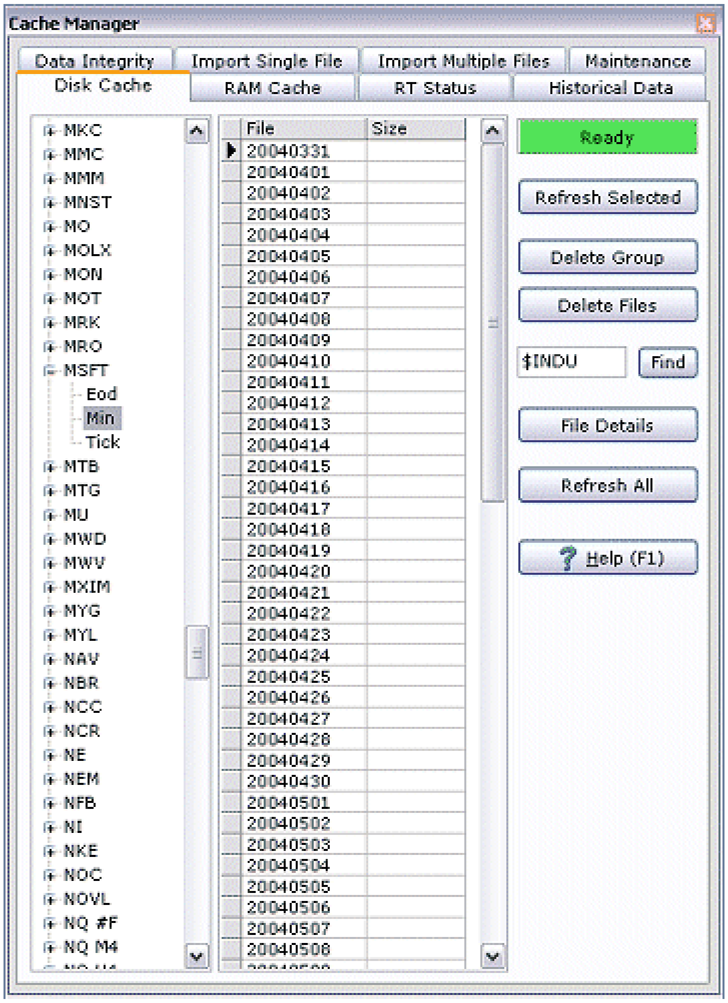
Disk Cache provides storage for longer term data to enhance performance. For lengthy data request, Disk Cache is consulted first before request is made to the data vendor.

Basic Operations

To view Disk Cache:

1. Choose **Manager>Cache** from the main window.
2. Press the **Disk Cache** tab.
3. It can take a few second for the cache manager to check your disk cache, a **Refreshing** label will be showing.
4. Once the **Refreshing** label becomes **Ready**, double click on **Symbol** on the left hand side.

5 Symbols that have disk cache data will be shown. You can press the **+** button to see the cache data.



- MKC
- MMC
- MMM
- MNST
- MO
- MOLX
- MON
- MOT
- MRK
- MRO
- MSFT
  - Eod
  - Min
  - Tick
- MTB
- MTG
- MU
- MWD
- MWV
- MXIM
- MYG
- MYL
- NAV
- NBR
- NCC
- NCR
- NE
- NEM
- NFB
- NI
- NKE
- NOC
- NOVL
- NQ #F
- NQ M4

File	Size
20040331	
20040401	
20040402	
20040403	
20040404	
20040405	
20040406	
20040407	
20040408	
20040409	
20040410	
20040411	
20040412	
20040413	
20040414	
20040415	
20040416	
20040417	
20040418	
20040419	
20040420	
20040421	
20040422	
20040423	
20040424	
20040425	
20040426	
20040427	
20040428	
20040429	
20040430	
20040501	
20040502	
20040503	
20040504	
20040505	
20040506	
20040507	
20040508	

Disk Cache Organization

Disk cache is organized by symbols. Each symbol stores three types of data: EOD, Min, Tick. To view a type of data, expand the symbol in the left hand side by pressing the **+** button, and choose the data type.

EOD data is responsible for daily data and above. EOD data is organized by year. So all dates of a year are stored in a single file.

Min data is responsible for minute data. Min data are organized by date. So all minute bars of a day are stored in a single file.

Tick data is responsible for tick and second data. Tick data are organized by date. So all ticks of a day are stored in a single file.

Disk cache maps to the file under the `Cache` folder in your NeoTicker® installation (e.g. `C:\Program Files\TickQuest\NeoTicker3\Cache`). The file organization structure is exactly as described above. So you can backup and manage the disk cache files by using Windows file operations.

Make sure you exit NeoTicker® before directly working on disk cache files.

Querying Disk Cache

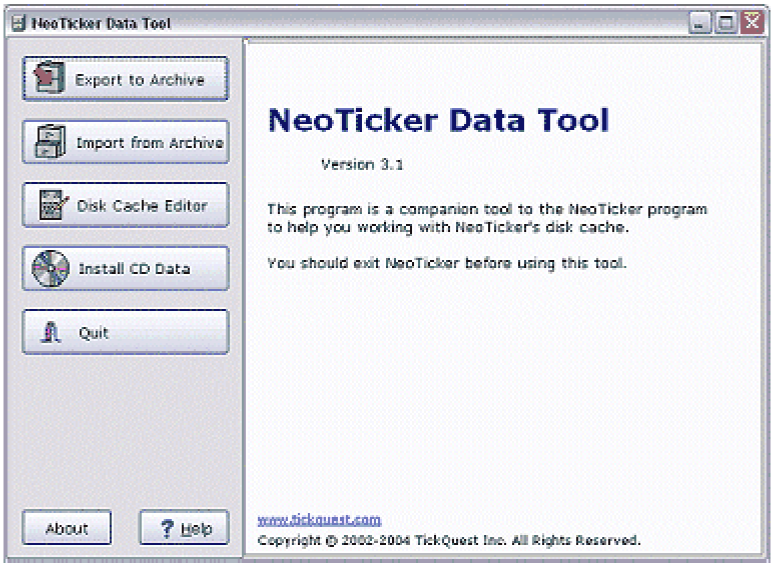
Due to performance reason, disk cache information displayed in cache manager is static, they do not change automatically to reflect changes in the disk cache. For instance, you may have disk cache files created by NeoTicker® for a new symbol, but the symbol will not show up automatically in Disk Cache.

To refresh the display, press one of the refresh buttons.

Archiving Disk Cache Data

NeoTicker® comes with a companion application called Data Tool. Data Tool provided archiving and high level editing of disk cache files. You can launch the data tool by choosing from Windows Start button, **Start>Programs>TickQuest>Data Tool**.

Refer to Data Tool's own manual on usage information.



NeoTicker Data Tool

Version 3.1

This program is a companion tool to the NeoTicker program to help you working with NeoTicker's disk cache.

You should exit NeoTicker before using this tool.

www.tickquest.com

Copyright © 2002-2004 TickQuest Inc. All Rights Reserved.

Cache Data Editor

You can edit data in disk cache using Cache Data Editor.

It is possible for data vendor to provide "bad ticks". Bad ticks can cause problems in charting and trading system analysis. Cache Data Editor is for correcting the data to solve these problems.

To open Cache Data Editor.

1 Open Cache Manager.

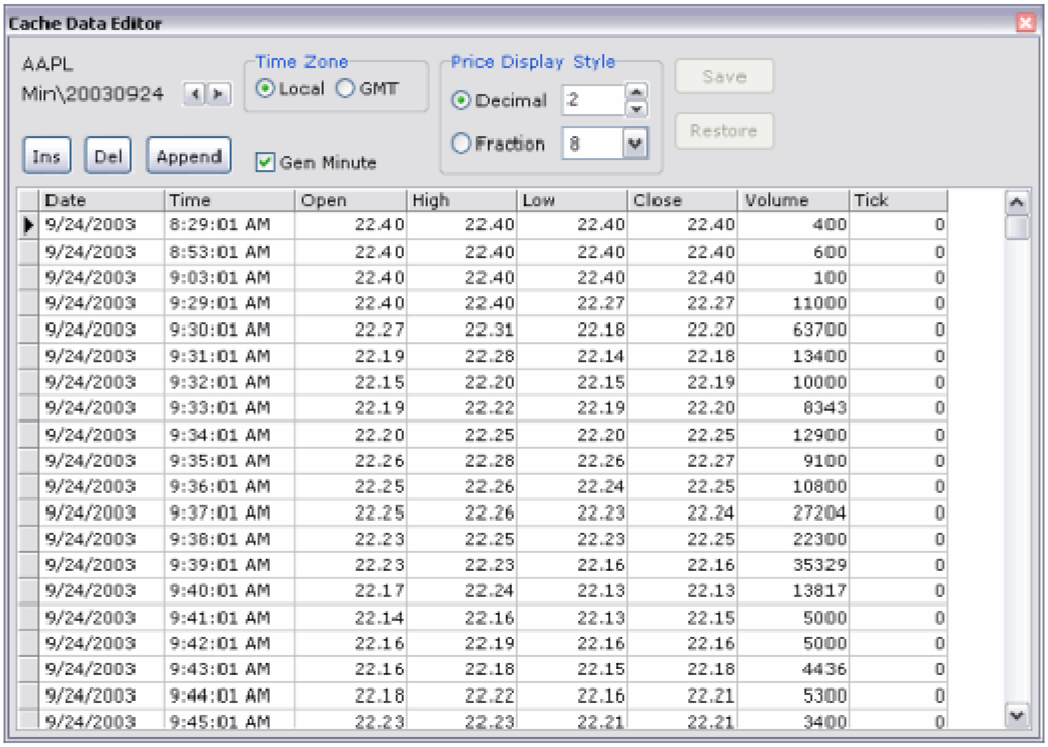
2 Press **Disk Cache** tab.

3 Open the folder for the symbol you want to edit data (you may need to double click on **Symbols** to see all the symbols).

4 Select the type of data to edit, **Eod** for daily data, **Min** for minute data, **Tick** for tick data.

5 In the right hand side, select the data file to edit. For minute and tick data, each file contains one day of data. For daily data, each file contains one year of data.

6 Press **Edit Data** to open Cache Data Editor.



AAPL
Min\20030924

Time Zone: ● Local ○ GMT

Price Display Style: ● Decimal 2 ○ Fraction 8

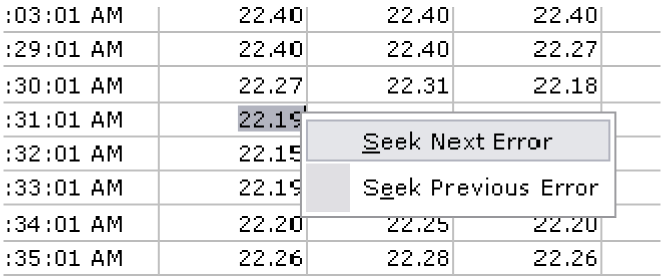
☑ Gen Minute

Date	Time	Open	High	Low	Close	Volume	Tick
9/24/2003	8:29:01 AM	22.40	22.40	22.40	22.40	400	0
9/24/2003	8:53:01 AM	22.40	22.40	22.40	22.40	600	0
9/24/2003	9:03:01 AM	22.40	22.40	22.40	22.40	100	0
9/24/2003	9:29:01 AM	22.40	22.40	22.27	22.27	11000	0
9/24/2003	9:30:01 AM	22.27	22.31	22.18	22.20	63700	0
9/24/2003	9:31:01 AM	22.19	22.28	22.14	22.18	13400	0
9/24/2003	9:32:01 AM	22.15	22.20	22.15	22.19	10000	0
9/24/2003	9:33:01 AM	22.19	22.22	22.19	22.20	8343	0
9/24/2003	9:34:01 AM	22.20	22.25	22.20	22.25	12900	0
9/24/2003	9:35:01 AM	22.26	22.28	22.26	22.27	9100	0
9/24/2003	9:36:01 AM	22.25	22.26	22.24	22.25	10800	0
9/24/2003	9:37:01 AM	22.25	22.26	22.23	22.24	27204	0
9/24/2003	9:38:01 AM	22.23	22.25	22.23	22.25	22300	0
9/24/2003	9:39:01 AM	22.23	22.23	22.16	22.16	35329	0
9/24/2003	9:40:01 AM	22.17	22.24	22.13	22.13	13817	0
9/24/2003	9:41:01 AM	22.14	22.16	22.13	22.15	5000	0
9/24/2003	9:42:01 AM	22.16	22.19	22.16	22.16	5000	0
9/24/2003	9:43:01 AM	22.16	22.18	22.15	22.18	4436	0
9/24/2003	9:44:01 AM	22.18	22.22	22.16	22.21	5300	0
9/24/2003	9:45:01 AM	22.23	22.23	22.21	22.21	3400	0

Editing data is straight forward. Just click on the cell you want to edit data and type in the new value. After you finish editing data, you will need to press the **Save** button to save changes you made.

Automatically Seek For Possible Data Error

Cache Data Editor can help you find possible data error. Simply right click on the data to open pop up menu and choose **Seek Next Error** or **Seek Previous Error**.



:03:01 AM	22.40	22.40	22.40
:29:01 AM	22.40	22.40	22.27
:30:01 AM	22.27	22.31	22.18
:31:01 AM	22.19		
:32:01 AM	22.15		
:33:01 AM	22.19		
:34:01 AM	22.20	22.25	22.20
:35:01 AM	22.26	22.28	22.26

Gen Minute

This option is applicable only when you are editing tick data. Enabling this option (default) will force the corresponding minute data to be adjusted using the edited tick data.

Changing Disk Cache Location

By default, disk cache is saved under the `cache` directory in the installation, e.g. `C:\Program Files\TickQuest\NeoTicker4`.

You can change the disk cache location. This is useful if you want to store disk cache files in a separate location, possibly a larger disk drive.

1 Choose **Program>User Preference** from main window.

2 Under the **Directories** tab, change the settings under **Disk Cache Parent Directory Location**.

Note that this is the directory that contains the `Cache` directory, so you do not need to include `Cache` in the setting. For example, if you change the directory to `D:\`, disk cache files will be saved to the folder `D:\cache`.

You must exit and restart NeoTicker® for the changes to take effect.

Checking File Size

Disk cache file size tells a lot of about the quality of a disk cache file. If you compare the file size of a disk cache with its peer, it should not deviate too much. A file size that deviates too much from its peer is a good indication of missing or corrupted data.

Rules of thumb when you inspect file size.

- For EOD data, except for the current year, all file size should be very close, almost identical.
- For Min data, except for the current trading day and holidays, all file size should be very close.
- For Tick data, size can vary a bit because the trading volume is different every day, and compression is used to handle tick data. Still, file size should be similar to peer except for current trading day and holidays.

To check file size of all files.

1 Make sure you are under **Disk Cache** tab in cache manager.

2 Click on the **+** sign besides the symbol.

3 Click on **EOD**, **Min**, or **Tick**.

4 Press the **File Details** button.

To check/update the file size of a specific cache file.

1 Make sure you are under **Disk Cache** tab in cache manager.

2 Click on the **+** sign besides the symbol.

3 Click on **EOD**, **Min**, or **Tick**.

4 Click the cell under the **Size** column for the specific file.

File size are not updated automatically for performance reason. You must use one of the method to query for updated file size.

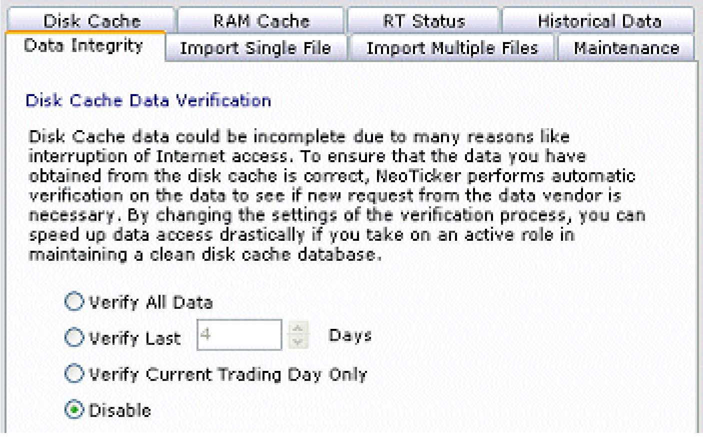
Data Verification between Disk Cache and Data Feed

To setup disk cache verification:

1 Choose **Manager>Cache** in the main window to open cache manager.

2 Press the **Data Integrity** tab.

3 Choose between **Verify All Data**, **Verify Last, Verify Current Trading Day Only, Disable**.



Disk Cache Data Verification

Disk Cache data could be incomplete due to many reasons like interruption of Internet access. To ensure that the data you have obtained from the disk cache is correct, NeoTicker performs automatic verification on the data to see if new request from the data vendor is necessary. By changing the settings of the verification process, you can speed up data access drastically if you take on an active role in maintaining a clean disk cache database.

○ Verify All Data

○ Verify Last 4 Days

○ Verify Current Trading Day Only

● Disable

What is Data Verification

When you chart a data, NeoTicker® needs to make a decision on how to handle data lies within the trading hours (excluding holidays). Normally, data can be retrieved from the disk cache without accessing your Internet data vendor's data server over the Internet.

From time to time, there will be gaps in the disk cache. The gaps are caused by different reasons:

- No trading happens during that time period due to holiday, lack of trading activities, trading halted.
- It just happens that the data is not stored in the disk cache, but the data is available in the Internet data vendor's server

Obviously in the former case, the data gap is normal and nothing has to be done. In the later case, NeoTicker® should retrieve the data from the data vendor and fill the gap.

From NeoTicker®'s perspective, it cannot tell the difference between the two types of data gap and certain assumptions have to be made on how to verify whether additional data can be retrieved from the data vendor to fill the data gap. The verification assumption is a trade off between speed and data integrity.

The less NeoTicker® trusts the data in the disk cache, the more verification it needs to make. This will improve data quality but at the expense of performance as extra time is needed to query a data server, possibly over the Internet.

The more NeoTicker® trusts the data in the disk cache, the less verification it needs to make. This will improve performance but at the expensive of data quality as data gaps may not be filled properly.

Verify All Data

Do not trust the data in Disk Cache. Verify the data whenever there is a gap in the data.

Verify Last

Only verify the data when the gap happens within the specified days from today. For example, if this value is set to 1, Disk Cache will verify last trading day's data plus the current trading day's data.

Verify Current Trading Day Only

Only verify the data of current trading day.

Disable

Trust the data in the disk cache. Never verify.

How you set the options depends on your trading style. If you are concern with historical data accuracy, you will want to use the first option, and settle with less performance. If you are less concern with historical data accuracy, you will want to use the latter two options and have better data loading performance.

Deleting Disk Cache Files

If you suspect a data problem is due to problems with disk cache, you can delete the cache file. After the disk file is deleted, when you request the data, it will be retrieved from the data feed.

Data feed have a limited history of data. If you have used NeoTicker® for several months, there is a good change your disk cache has more historical data than your data feed. In this case, deleting the disk cache is not recommended. You should consult your data vendor for the length of historical data available.

To delete disk cache files, make sure you are under the **Disk Cache** tab in cache manager.

To delete all disk cache files of a symbol:

1 Click on the symbol.

2 Press the **Delete Group** button.

To delete all EOD disk cache files of a symbol:

1 Click on the **+** sign besides the symbol.

2 Click on **EOD**.

3 Press the **Delete Group** button.

To delete all Min disk cache files of a symbol:

1 Click on the **+** sign besides the symbol.

2 Click on **Min**.

3 Press the **Delete Group** button.

To delete all Tick disk cache files of a symbol:

1 Click on the **+** sign besides the symbol.

2 Click on **Tick**.

3 Press the **Delete Group** button.

To delete a specific disk cache file.

1 Click on the **+** sign besides the symbol.

2 Click on **EOD**, **Min**, or **Tick**.

3 Click on the file in the right hand side list.

4 Press the **Delete File** button.

Discovering New Sample Data

Each release can come with new sample data. When you run NeoTicker® after an upgrade, these new sample is found and NeoTicker® will ask you if you want to merge the new sample data into your disk cache.

You have the following choices:

Merge Data

Sample data are merged to disk cache. If there is a date collision, you will be prompt how to handle the collision.

Cancel, I will merge later

Sample data are not merged to disk cache. You can merge later by:

1 Open cache manager.

2 Click on **Historical Data** tab.

3 Press the **Check Sample Data** button.

Cancel, I will not merge later

Sample data are not merged to disk cache and will be deleted.

File Locking

By default, Disk Cache files are not locked. This means a chart can overwrite the data by data loading, repair etc. Sometimes you may want to protect Disk Cache files by locking them. Usually you want to do this after editing a file manually to correct bad data from data vendor.

To lock/unlock a file:

1 Open Cache Manager.

2 Press **Disk Cache** tab.

3 Open the folder for the symbol you want to edit data (you may need to double click on **Symbols** to see all the symbols).

4 Select the type of data to edit, **Eod** for daily data, **Min** for minute data, **Tick** for tick data.

5 In the right hand side, select the data file(s) to lock/unlock.

6 Press the **Lock Files / Unlock Files** buttons to lock/unlock the file(s).

Importing Text File to Disk Cache

When to Import to Disk Cache

Import the text file to disk cache if you want the data to integrate with real-time server. Typically usage is import additional historical data that your data feed does not provide.

Importing a Single Text File

1 Choose **Manager>Cache** in the main window to open cache manager.

2 Press **Import Single File** tab in cache manager.

3 In the **Source** group, choose the file format, enter directory and file name, choose data type.

4 In the **Target** group, enter **Symbol** (this is symbol which data will be imported to) and choose other options.

5 Press **Import Now** button.

Importing Multiple Text Files

1 Choose Manager>Cache in the main window to open cache manager.

2 Press **Import Multiple Files** tab.

3 In the **Source** group, choose the file format, enter directory, choose data type.

4 Choose options in the **Target** group.

5 Press the **Read Dir** button. The files in the directory will be read into the source file/target symbol list.

6 You can choose which file to import, and the target file name for each text file.

7 Press the **Import Now** button.

Settings for importing multiple files can be saved to a file. You can later re-use the same settings by loading them from a file. To save/load settings, press the **Save / Load** button.

Option

The following options under **Option** group are available when importing text files into disk cache:

Option	Usage
Adjust Time Stamp	Offset the time stamp by the specified number of seconds. This option is for importing text files that have a different time stamp convention from NeoTicker®'s. NeoTicker® expects bars to be stamped at end of a time period.
Adjust Volume	Multiple the volume by a multiple. This is for text files that has a volume reported in lots. NeoTicker® expects volumes in number of contracts/stocks.
Construct Minute Bars From Imported Tick Data	When importing tick data, construct minute bars based on the imported tick data.

Compatible Text File Formats

See ***Compatible Data File Formats*** (on page 530).

Locating Symbol

To locate a symbol in the disk cache.

1. Make sure you are under the **Disk Cache** tab in cache manager.
2. Enter the symbol in the field besides the **Find** button.
3. Press the **Find** button.

Read-Only Mode (Simulation Server)

When Simulation Server is the real-time data feed, it is not possible to modify Disk Cache data and Cache Manager will go into read-only mode. Read-only mode is marked on the caption of Cache Manager and buttons that cannot be used will be disabled.

To get out of read-only mode, temporarily configure real-time server to offline:

1. Choose **Program>Server Setup** from main window.
2. Under **Data Feed** tab, choose **No Feed**.
3. Press **OK** button.

4 Restart NeoTicker®.

5 Perform the Disk Cache operations as required.

6 Use Server Setup to reconfigure data feed to **Simulation**.

7 Restart NeoTicker®.

Refreshing Disk Cache

Disk cache information in the cache manager does not automatically change to reflect disk cache changes.

First, make sure you are under the **Disk Cache** tab in cache manager.

To refresh the display for all symbols, press the **Refresh All** button.

To refresh the display for a particular symbol, click the symbol on the left hand side, and press the **Refresh Selected** button.

To refresh a type of data for a particular symbol, click the data type on the left hand side, and press the **Refresh Selected** button.

Retrieving a Range of Data to Fill Disk Cache

1 Choose **Manager>Cache** from the main window

2 Press the **Historical Data** tab

3 Specify a **From** and **To** range.

4 Choose an option under **Symbols to Download**.

5 Press the **Request Data** button.

This feature is useful when you need to fill in the disk cache for the first time or if you have data gaps in the disk cache due to switch of data vendors or not using NeoTicker® for a while.

The data request options are the same as the nightly data retrieval. See ***Retrieving Data from Data Vendor Every Night*** (on page 456) for more information.

Many Internet data vendors have a limit on data retrieval per session to restrict large scale data download. This will cause data holes when you try to retrieve long historical data of many symbols. To work around this problem, you should only retrieve a short list of symbols (around 50 should be safe) at a time, then replace the symbol list, disconnect and reconnect to the data server, and download.

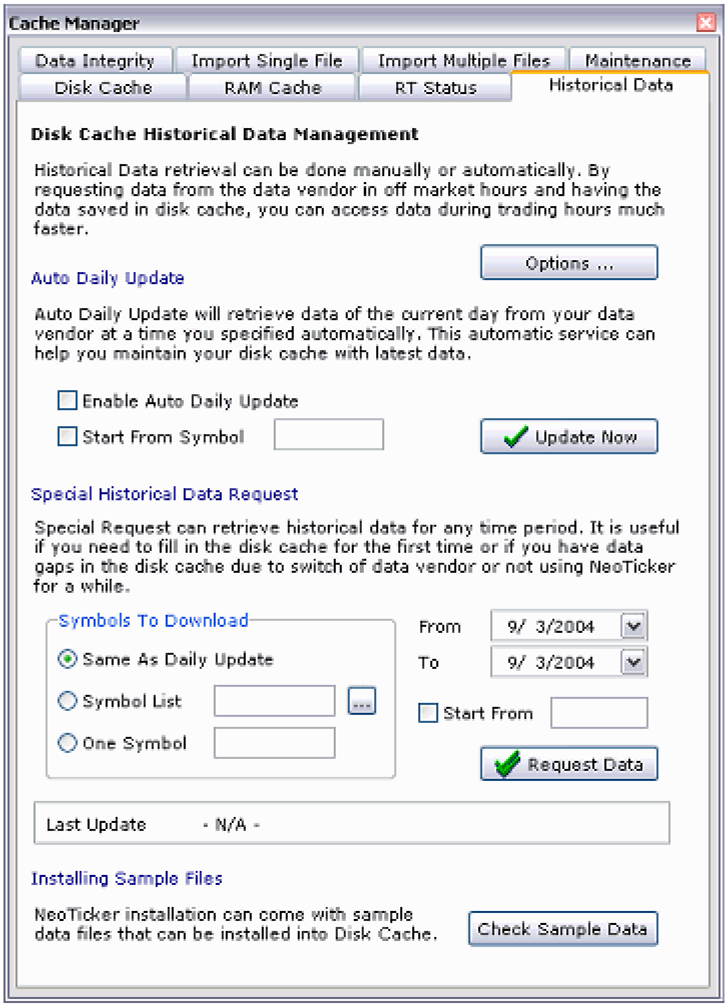
Retrieving Data from Data Vendor Every Night

CNeoTicker® can chart a lot faster if the data is already in the disk cache because less data are requested from the server. If you track a lot of different symbols, you can download data nightly to the disk cache. The next day, you will have last trading day's data in disk cache.

To set up:

1. Choose **Manager>Cache** from the main window.
2. Press the **Historical Data** tab.
3. Turn on the option **Enable Auto Daily Update**.
4. Press the **Option** button.
5. Specify a download time.
6. Assume your data vendor clean up the data in their historical database, leave the **Auto sync RAM Cache data** option on. When this option is on, disk cache data will be sent to RAM Cache after download, ensuring a clean RAM Cache the next day.
7. If you know there are gaps in intraday data caused by data interruption, turn on the **Forced Update On All Files** option. When this option is on, data request will always request data from the data vendor to fill a disk cache file, rather than skipping data files that appear completed to improve data request performance.
8. Select the source of symbol. It is either the symbols currently in the disk cache, or a symbol list you specified.
9. Decide the type of data you want to request. Only turn on tick request if you have lots of disk space.
10. Press the **OK** button.
11. Close the cache manager.

You need to leave NeoTicker® running at overnight in order for the download to happen. Only the current trading day's data is retrieved.



Disk Cache Historical Data Management

Historical Data retrieval can be done manually or automatically. By requesting data from the data vendor in off market hours and having the data saved in disk cache, you can access data during trading hours much faster.

Auto Daily Update

Auto Daily Update will retrieve data of the current day from your data vendor at a time you specified automatically. This automatic service can help you maintain your disk cache with latest data.

□ Enable Auto Daily Update

□ Start From Symbol

Special Historical Data Request

Special Request can retrieve historical data for any time period. It is useful if you need to fill in the disk cache for the first time or if you have data gaps in the disk cache due to switch of data vendor or not using NeoTicker for a while.

Symbols To Download

● Same As Daily Update

○ Symbol List

○ One Symbol

From 9/ 3/2004

To 9/ 3/2004

□ Start From

Last Update - N/A -

Installing Sample Files

NeoTicker installation can come with sample data files that can be installed into Disk Cache.

Manually Starting an Nightly Update

Press the **Update Now** button.

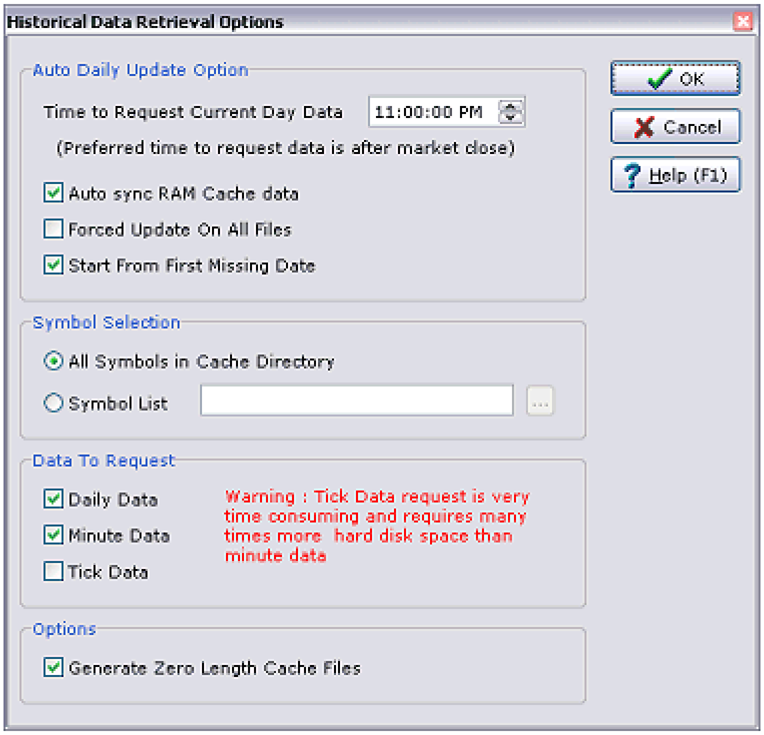
Update Options

Start From Symbol

Historical update will start from the symbol specified, skipping all symbols prior to it.

Example usage: When a historical update fails (e.g. data vendor server went down), historical update will leave a message on the status bar. You will be told at which symbol the update fails. You can turn on this option, enter the symbol and press **Request Data** button to restart the update.

Additional options are set when you press the **Options** button.



Historical Data Retrieval Options

Auto Daily Update Option

Time to Request Current Day Data 11:00:00 PM

(Preferred time to request data is after market close)

☑ Auto sync RAM Cache data

□ Forced Update On All Files

☑ Start From First Missing Date

Symbol Selection

● All Symbols in Cache Directory

○ Symbol List

Data To Request

☑ Daily Data

☑ Minute Data

□ Tick Data

Warning : Tick Data request is very time consuming and requires many times more hard disk space than minute data

Options

☑ Generate Zero Length Cache Files

Time to Request Current Day Data

The time that NeoTicker® will launch the auto update everyday. It is best that you set it to a time after market closed for a few hours.

Auto sync RAM Cache data

RAM Cache data will be synchronized using the data downloaded, i.e. RAM Cache will be changed by the disk cache data.

Forced Update On All Files

When this option is on, data request will always request data from the data vendor to fill a disk cache file, rather than skipping data files that appear completed to improve data request performance.

Start From First Missing Date

Automatically looks for the first date that does not have data, and start downloading from this date.

This option is on by default. You should only turn this option off if your data vendor is slow and you do not require more than one to two days of historical data.

All Symbols in Cache Directory

Choose this option to let NeoTicker® update to all the symbols you have used in real-time.

Symbol List

Choose this option if you must specific a symbol list as the reference to what data to retrieve.

Daily Data

Check this option to enable retrieving of daily data.

Minute Data

Check this option to enable retrieving of minute bar data.

Tick Data

Check this option to enable retrieving of tick data. Even though NeoTicker® compresses cached tick data, it will still require much more disk space than minute data. Make sure you have enough free space on your hard disk before you enable this option.

Generate Zero Length Cache Files

If data vendor does not return any data for a date, create a zero length cache file.

This option is on by default. On holidays and weekends, data vendor does not return data. Having a zero length cache file created signifies that no data is normal and NeoTicker® will not bother to re-verify the data. This greatly improve multiple charting and scanning performance.

Turn this option off only if you have unreliable data vendor or Internet connection.

Cache Manager - RAM Cache

NeoTicker® puts trading data into RAM to optimize performance. This is known as RAM Cache. RAM Cache is responsible for the following tasks.

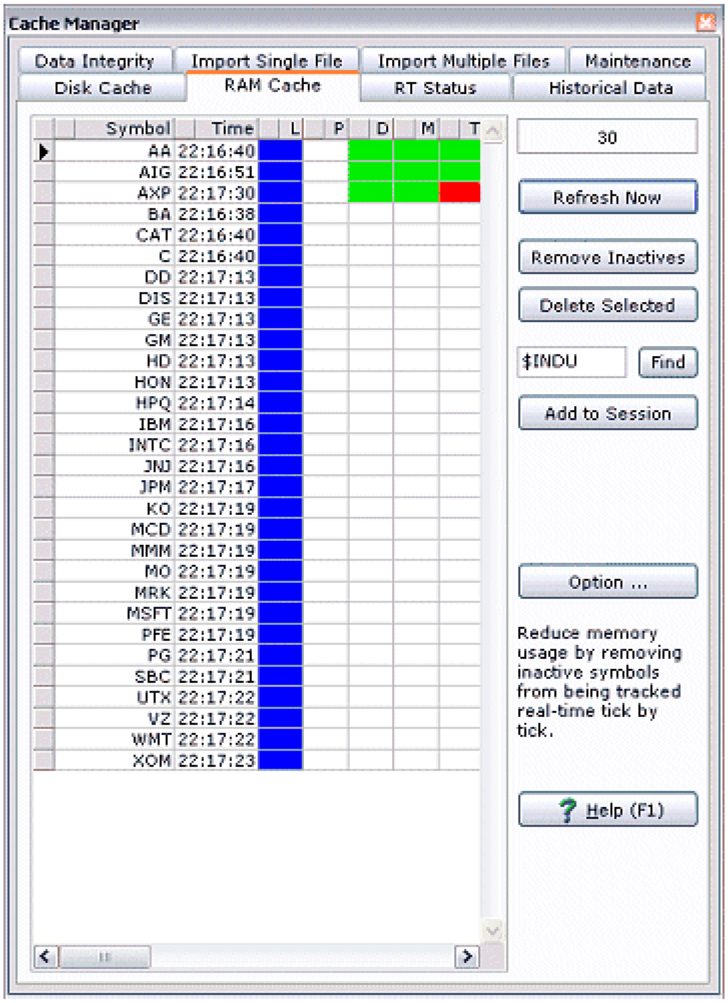
- Real-time tick-by-tick update of RAM Cache data
- Maintenance of data series and indicators used by the quote window, dynamic grid, cluster.
- Provide high speed data loading for the other function windows
- Schedule pre-load symbol list data loading

Basic Operations

To view the RAM Cache.

1. Choose **Manager>Cache** from the main window.
2. Press the **RAM Cache** tab.

3 Press the **Refresh Now** button to update RAM Cache status.



Symbol	Time	L	P	D	M	T
AA	22:16:40	🟦		🟩	🟩	🟩
AIG	22:16:51	🟦		🟩	🟩	🟩
AXP	22:17:30	🟦		🟩	🟩	🟥
BA	22:16:38	🟦				
CAT	22:16:40	🟦				
C	22:16:40	🟦				
DD	22:17:13	🟦				
DIS	22:17:13	🟦				
GE	22:17:13	🟦				
GM	22:17:13	🟦				
HD	22:17:13	🟦				
HON	22:17:13	🟦				
HPQ	22:17:14	🟦				
IBM	22:17:16	🟦				
INTC	22:17:16	🟦				
JNJ	22:17:16	🟦				
JPM	22:17:17	🟦				
KO	22:17:19	🟦				
MCD	22:17:19	🟦				
MMM	22:17:19	🟦				
MO	22:17:19	🟦				
MRK	22:17:19	🟦				
MSFT	22:17:19	🟦				
PFE	22:17:19	🟦				
PG	22:17:21	🟦				
SBC	22:17:21	🟦				
UTX	22:17:22	🟦				
VZ	22:17:22	🟦				
WMT	22:17:22	🟦				
XOM	22:17:23	🟦				

Reduce memory usage by removing inactive symbols from being tracked real-time tick by tick.

RAM Cache display is static. You must press the **Refresh Now** button to see the changes in RAM Cache.

You can click on the heading buttons to sort the listing according to the alphabetical order of the symbols or in chronological order of the last access of the cache data.

Adding Symbol to RAM Cache

See also ***Automatically Insert Symbols into RAM Cache*** (on page 464).

Adding a Single Symbol

To add a cache symbol similar to the pre-load symbol list service:

1 Make sure you are under cache manager's **RAM Cache** tab.

2 Enter the symbol in the field above the **Add to Session** button.

3 Press the **Add to Seesion** button.

4 You can press the **Refresh Now** button to see the progress.

If you add a symbol this way, the symbol will be locked (with a blue mark in the **L** column). Locked symbols are not removed during auto cleanup.

Adding a Symbol List

You will use the pre-load symbol list feature to help you.

1 Make sure you are under the **RAM Cache** tab in cache manager.

2 Press the **Option** button.

3 Press the **Preload Symbol List** tab.

4 Choose **Use Symbol List** and provide a symbol list.

5 Press the **Add Symbols to RAM Cache from Designated Symbol List Now** button.

6 If you do not want the symbol list to be pre-loaded at next startup, choose **Empty List on Restart**.

Auto Cleanup, Locked and Protected Symbols

Auto Cleanup

Symbols in RAM Cache occupies memory and if you use an Internet data feed, these symbols are counted towards your symbol limit. Therefore, RAM Cache has a schedule to automatically clean up the symbols in use.

You can observe the clean up status of a symbol in Cache Manager's **RAM Cache** tab. For symbols that are not accessed for a long period of time, the symbol will be displayed in yellow background. For symbols that are not accessed for even longer period of time and will be removed in the next auto clean up service will be displayed in magenta.

The clean up happens every 5 minutes and unused symbols are removed from the RAM Cache. You can configure auto cleanup schedule by:

1 Make sure you are under **RAM Cache** tab in cache manager.

2 Press the **Option** button.

3 Press the **Auto Cleanup** tab.

4 Make adjustment to schedules.

Locked and Protected Symbols

Symbols in the RAM Cache are inserted when you access the symbol in charts or when specified as pre-load. There are three status for a symbol: temp, locked and protected. In cache manager's **RAM Cache** tab, locked symbol has a blue color code under the **L** column and protected symbol has a red color code under the **P** column. Temp symbols has no special marking.

Symbol locking is used to control how symbols are auto-removed from the RAM Cache.

Temp symbol is removed by auto cleanup if the symbol is not used by any window for a while.

Locked symbol stays in the RAM Cache until you must remove it manually.

Protected symbols are never removed. They are used when you use an Internet data feed that does not provide historical data (e.g. Interactive Brokers, DDE feeds). Symbol protection ensures that NeoTicker® always collects data for the protected symbol.

Automatically Insert Symbols into RAM Cache

Auto inserting RAM Cache will allow fast access for frequently used symbols as they are auto inserted into the RAM Cache. For example, if you open a chart for MSFT, RAM Cache will include MSFT automatically. Opening another chart of MSFT will be lightening fast.

To setup:

1 Make sure you are under the **RAM Cache** tab in cache manager.

2 Press the **Option** button.

3 Press the **Smart Tools** tab.

4 Choose one of **Disable**, **Insert as Locked Symbol** or **Insert as Temp Symbol**.

See ***Auto Cleanup, Locked and Protected Symbols*** (on page 463) for the definitions of locked symbol and temp symbol.

Auto insert symbols can use up memory quickly without you knowing it. If you experience stability problems, turn this feature off.

Bars Per Series

To adjust bars per series:

1 Make sure you are under the **RAM Cache** tab in cache manager.

2 Press the **Option** button.

3 Press the **Bars Per Series** tab.

4 Adjust the **Minimum** and **Maximum** bars per series.

What does Bars Per Series Do

Bars per series settings controls the number of bars stored in RAM Cache for indicators. **Minimum** value (default to 250 for intraday, 500 for daily) is used when an indicator is inserted into the RAM Cache. As data comes in and indicators are recalculated, additional bars are stored in the RAM Cache. **Maximum** limits the number of bars stored in RAM Cache to preserve memory.

In another words, you can think of RAM Cache as a 'hidden chart' that serves indicator bars to clients such as quote window. Bar per series is thus controls the minimum and maximum bar size of this 'hidden chart'.

The common usage for bars per series is in quote window formula. Suppose the quote window formula request data that is 345 bars ago (e.g. `mov(345, M5, "Simple", 10)` is the 5-minute 10-period simple moving average 345 bars ago). You will need to specify the minute parameters to at least 345 in order for this formula to work in quote window. Note that this setting is not applicable to time charts and pattern scanners. Although these features use RAM Cache, their indicators' correctness depends on the days to load within their function window.

Reset to Default

You can press the **Use Defaults** button to reset bars per series setting to default. This is useful when:

- Your indicator calculation does not return correct answer due to insufficient data.
- Your indicator calculation has initialization problem due to excessive data loading.

Changing RAM Cache Temporary Files Location

RAM Cache saves to temporary files for faster access. By default, RAM Cache temporary files are stored under the TempBuffer directory, e.g. `C:\Program Files\TickQuest\NeoTicker3\TempBuffer.`

You can change the temporary files location. This is useful if you want to store temporary files in a separate location, possibly a larger disk drive.

1 Choose **Program>User Preference** from main window.

2 Under the **Directories** tab, change the settings for `RAM Cache Temp Directory Location`.

For example, if you change the directory to `D:\`, temporary files will be saved to `D:\TempBuffer`.

You must exit and restart NeoTicker® for the changes to take effect.

Creating a Symbol List from Current RAM Cache Symbols

You can create a symbol list from the current symbols in RAM Cache.

1. Make sure you are under the **RAM Cache** tab in cache manager.
2. Press the **Option** button.
3. Press the **Preload Symbol List** tab.
4. Press the **Save Current RAM Cache Symbols to Designated Symbol List** button.
5. Provide the file name for the symbol list.

Data Integrity Checking

When NeoTicker® exits, RAM Cache data is saved to files. When NeoTicker® launches again, RAM Cache is filled using these data files.

NeoTicker® performs integrity checks on these data files before using them to fill RAM Cache. There are two levels of integrity checks - **Complete Sanity Check** and **Basic File Verification**. Complete Sanity Check is the default introduced in Version 4.1. Basic FIle Verification does minimal checking and is suitable for slower computers.

To adjust integrity check levels:

1. Open Cache Manager.
2. Press **RAM Cache** tab.
3. Press **Option** button to open RAM Cache Options window.
4. In RAM Cache Options window, press **Advance** tab.
5. Adjust the level under **RAM Cache Integrity**.
6. Press **OK** button.

Days of Data to Load

To adjust the settings for days of data to load in RAM Cache:

1. Make sure you are under the **RAM Cache** tab in cache manager.

2 Press the **Option** button.

3 Press the **Days to Auto Load** tab.

4 Adjust the settings.

What does Days of Data to Load Do

When a symbol is inserted into RAM Cache, historical data for the symbol is automatically requested. You can think of this as a 'hidden chart' that contains 1-tick, 1-minute and 1-day data. When a client such as pattern scanner requests data from the RAM Cache, RAM Cache will use this 'hidden chart' to construct the data for the client, e.g. RAM Cache can make 5-minute bar, 30-second bar, etc. for the client.

By default, only minute data and daily data is requested. You can turn on the optional tick data provided:

- You have enough memory (512M is recommended).
- Your data vendor provides decent historical tick data support.

Trading Days

RAM Cache days to load is specified by trading days. So it has settings for you to specify trading days definitions, i.e. days in a week, holidays.

If you prefer calendar days, simply turn on 7 days a week trading and turn off holiday list.

Examples

- In pattern scanner, if your scanning criteria needs more than 10 days of data for minute data, you can increase the number of days to load for minute. By having all the data within the RAM Cache, you can greatly improve the scanning speed.
- If you keep track of many symbols in RAM Cache, to avoid running out of memory, you can set a maximum on the days of data available.

Fixing RAM Cache Corruption

RAM Cache temporary files are temporary files that help NeoTicker® restarts faster.

RAM Cache temporary files are not meant to be persistent data storage. If NeoTicker® exits abnormally (e.g. power outage), it is possible for the RAM Cache temporary files to be corrupted.

You can fix this scenario by removing the RAM Cache temporary files. This is a safe operation.

1 Open cache manager.

2 Press the **Maintenance** tab.

3 Check **Tick** and/or **Min** for the data you want to delete.

4 Press the **Delete Selected Files Now** button.

Flushing RAM Cache Data into Disk Cache

RAM Cache data is flushed down to disk cache by default. It makes the data available for long term use and helps data consistency.

Automatic Saving

If you want to turn on/off automatic flushing:

1 Make sure you are under the **RAM Cache** tab in cache manager.

2 Press the **Option** button.

3 Press the **Smart Tools** tab.

4 Toggle **Enable Auto Save**.

Manual Saving

If you do not have automatic flushing enabled, and want to flush RAM Cache data to disk cache:

1 In cache manager, press the **Maintenance** tab.

2 Press the **Flush RAM Cache Today's Data to Disk** button or **Flush Protected Symbols Today's Data to Disk** button.

Locating Symbol

To locate a symbol in RAM Cache.

1 Make sure you are under the **RAM Cache** tab in cache manager.

2 Enter the symbol in the field besides the **Find** button.

3 Press the **Find** button.

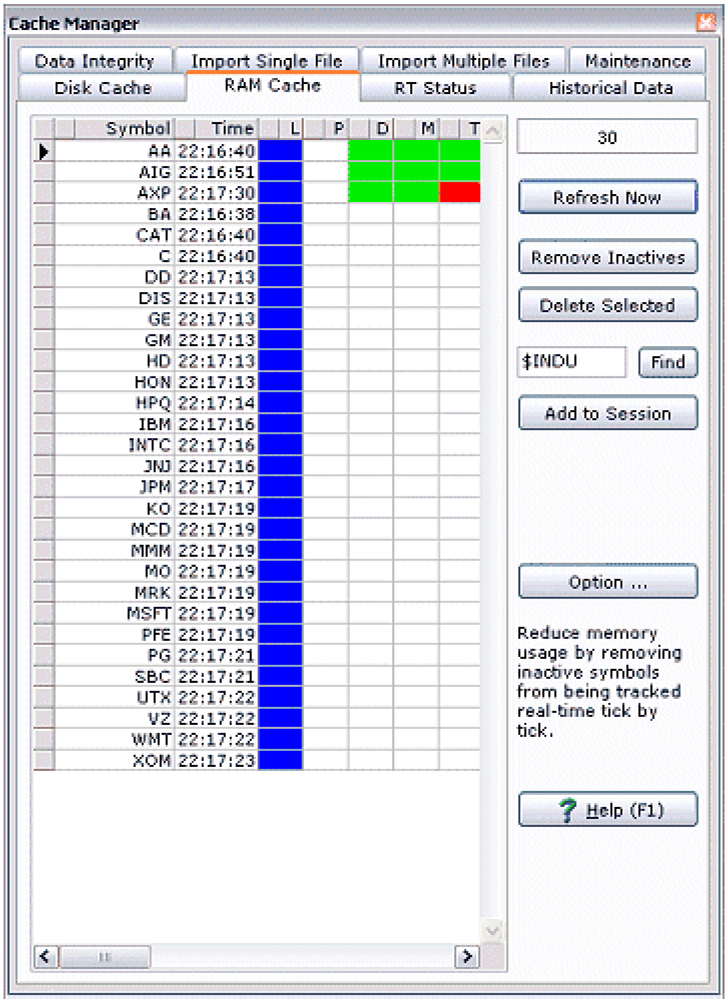
Monitor Data Status in RAM Cache

To monitor data status in RAM Cache:

1 Choose **manager>cache** from the main window to open the cache manager.

2 Press the **RAM Cache** tab.

3 You need to press the **Refresh Now** button to refresh the display.



Symbol	Time	L	P	D	M	T
AA	22:16:40	🟦		🟩	🟩	🟩
AIG	22:16:51	🟦		🟩	🟩	🟩
AXP	22:17:30	🟦		🟩	🟩	🟥
BA	22:16:38	🟦				
CAT	22:16:40	🟦				
C	22:16:40	🟦				
DD	22:17:13	🟦				
DIS	22:17:13	🟦				
GE	22:17:13	🟦				
GM	22:17:13	🟦				
HD	22:17:13	🟦				
HON	22:17:13	🟦				
HPQ	22:17:14	🟦				
IBM	22:17:16	🟦				
INTC	22:17:16	🟦				
JNJ	22:17:16	🟦				
JPM	22:17:17	🟦				
KO	22:17:19	🟦				
MCD	22:17:19	🟦				
MMM	22:17:19	🟦				
MO	22:17:19	🟦				
MRK	22:17:19	🟦				
MSFT	22:17:19	🟦				
PFE	22:17:19	🟦				
PG	22:17:21	🟦				
SBC	22:17:21	🟦				
UTX	22:17:22	🟦				
VZ	22:17:22	🟦				
WMT	22:17:22	🟦				
XOM	22:17:23	🟦				

Reduce memory usage by removing inactive symbols from being tracked real-time tick by tick.

D, M, T Columns

D, M, T stands for Daily, Minute and Tick. They are the raw data that is requested from the data feed. Note that second data is constructed from tick.

A green cell indicates the data is ready for use. A red cell indicates data is currently being loaded. A white cell indicates the data has not started loading.

DB, MB, TB Columns

DB, MB, TB stands for Daily Buffer, Minute Buffer and Tick Buffer. The numbers under this column tells you how many 1-day bars, 1-minute bars and 1-tick driven bars are available in the RAM Cache.

When a data series is constructed, if the amount of data needed can be constructed with what is available in the RAM Cache, speed is very fast. For example, a data series of 100 5-minute bar will be covered by a Minute Buffer of the size 500 or more.

If there are not enough data in the RAM Cache, the disk cache will be queried for data. If there are not enough data in the disk cache, the data feed will be queried.

DC, IC Columns

DC stands for Daily Series Count and IC stands for Intraday Series Count. They are the number of data series that are currently inside the RAM Cache. For example, a 5-minute data series and a 1-minute data series are counted as two data series.

The numbers shown are data series in use vs. data series available. For example, 2/5 stands for 5 data series and 2 of them are actively in use.

Performance Tuning

You can optimize RAM Cache for speed or for conserving memory. For most user, the default optimize for speed setting is recommended. Conserve memory is useful if you perform a lot temporary access to many symbols, e.g. large scale indicator scanning using pattern scanner.

To adjust the settings:

1 Open Cache Manager by choosing **Manager>Cache** from main window.

2 In Cache Manager, press the **RAM Cache** tab.

3 Press the **Option** button.

4 In RAM Cache Options, press **Advance** tab.

5 In the **Performance Tuning** box, choose either **Optimize Access Speed** or **Conserve Memory**.

6 Press **OK** button.

Optimize Access Speed

If you choose this option, if you access historical data that is not in RAM Cache, the data is requested from disk cache or data vendor, returned to you and cached into RAM Cache. Subsequent access to the same data will be very fast because the data is now available in RAM Cache.

Conserve Memory

If you choose this option, if you access historical data that is not in RAM Cache, the data is return directly from either disk cache or data vendor, then returned to you. The data is not cached in RAM to conserve memory. For subsequent access, the data has to be requested from disk cache.

Pre-loading Symbols at Startup

You can set a pre-load symbol list for the RAM Cache to load whenever you are reconnecting to your data feed.

The symbol list is a simply a file that contains one symbol per line. The file can be edited using Windows' Notepad application. NeoTicker® comes with several symbol lists and you can modify from these list for your own customized list.

Make sure you do not exceed your data vendor's imposed symbol limit when you create a symbol list for RAM Cache because all the symbols in the RAM Cache is counted as part of your symbol limit.

To setup:

1 Make sure you are under the **RAM Cache** tab in cache manager.

2 Press the **Option** button.

3 Press the **Preload Symbol List** tab.

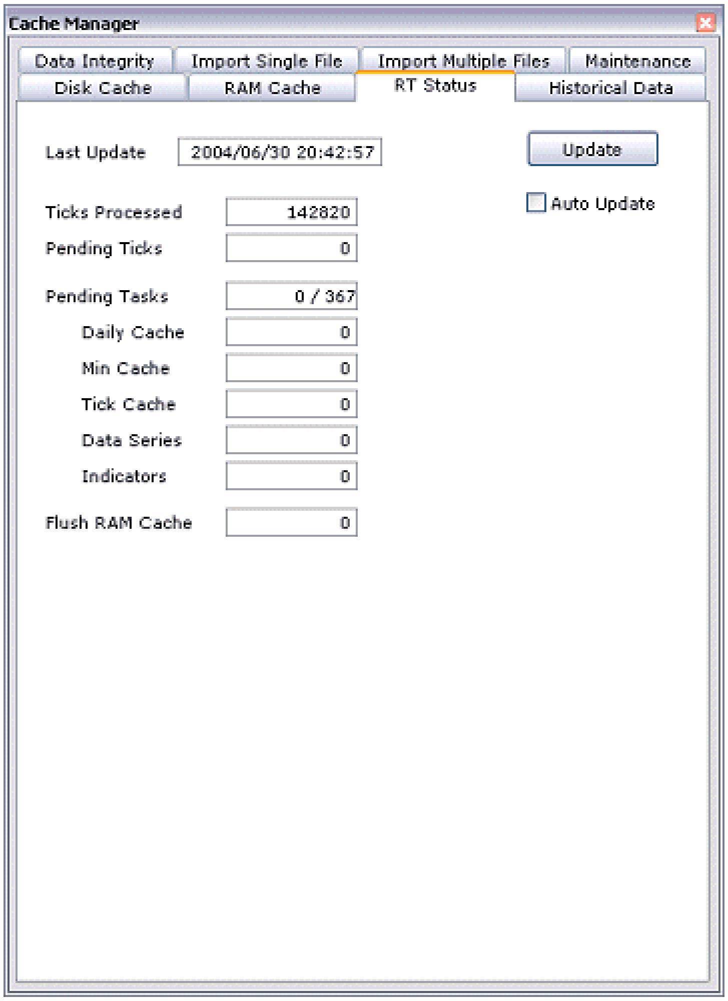
4 Choose **Use Symbol List** and provide a symbol list.

Real-time Status

You can view the RAM Cache real-time status for trouble shooting RAM Cache problems.

1 Open Cache Manager.

2 Press **RT Status** tab.



Last Update 2004/06/30 20:42:57

Ticks Processed 142820

Pending Ticks 0

Pending Tasks 0 / 367

Daily Cache 0

Min Cache 0

Tick Cache 0

Data Series 0

Indicators 0

Flush RAM Cache 0

Ticks Processed - Number of ticks processed by RAM Cache since program start

Pending Ticks - Number of ticks waiting to be processed. If this number continue to increase, it indicates your computer does not have enough processing power for the current set up.

Pending Tasks - Number of tasks to perform. This number should increase when you perform a task that involves RAM Cache. It will slowly decrease to 0.

Daily Cache - Number of raw daily series

Min Cache - Number of raw minute series

Tick Cache - Number of raw tick series

Data Series - Number of data series, compressed from the raw series

Indicators - Number of indicators

Flush RAM Cache - Number of flush jobs to perform. If this number continue to increase, it indicates your computer does not have enough processing power for the current set up.

Update / Auto Update - Press Update button to update this tab manually, or use Auto Update to have RT Status update itself every several seconds.

Removing Symbols from RAM Cache

If you are tight on memory or close to your symbol limit, you can manually remove symbols from the RAM Cache.

Remove Inactive Symbols

To remove inactive symbols:

1. Make sure you are under cache manager's **RAM Cache** tab.
2. Press the **Remove Inactives** button.
3. Removal will take a few seconds. You can press the **Refresh Now** button to see the progress.

Delete Selected

To remove specific symbols:

1. Make sure you are under cache manager's **RAM Cache** tab.
2. Click on the symbol you want to deleted to select it. You can select multiple symbols by SHIFT-click or CTRL-click.
3. Press the **Deleted Selected** button.

4 Removal will take a few seconds. You can press the **Refresh Now** button to see the progress.

If you are currently using the deleted symbol is currently in used, then the symbol will be added back to the list automatically.

Symbol Initialization Speed

You can control how fast a symbol is initialized when it is added to RAM Cache. The trade off is a faster initialization speed requires more CPU power. If you have a slow computer, the overall feeling of the application will become sluggish.

To adjust symbol initialization speed:

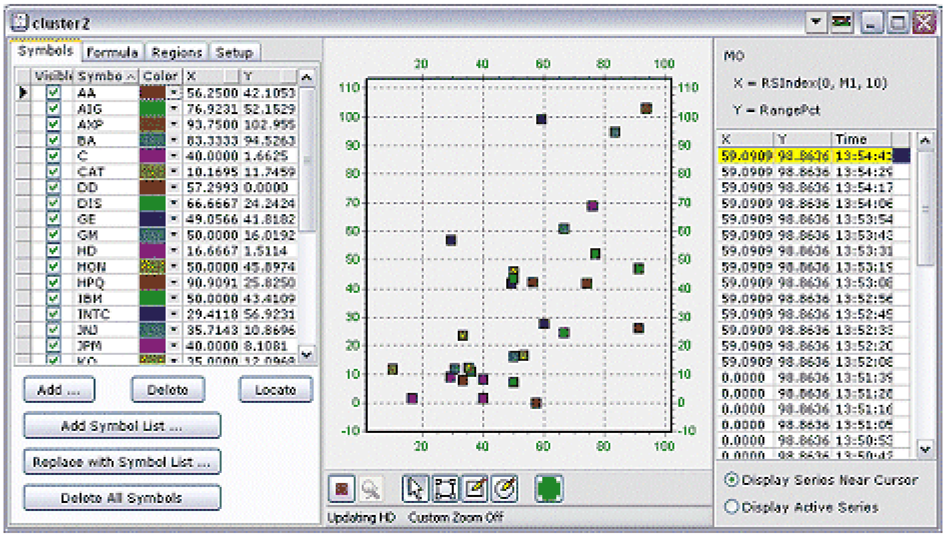
1 Make sure you are under **RAM Cache** tab in cache manager.

2 Press the **Option** button.

3 Press the **Advance** tab.

4 Select one of the options under **Symbol Initialization Speed**.

Cluster Window Operation Guide



	Visible	Symbo ^	Color	X	Y
▶	☑	AA	🟫	56.2500	42.1053
	☑	AIG	🟩	76.9231	52.1529
	☑	AXP	▨	93.7500	102.955
	☑	BA	▨	83.3333	94.5263
	☑	C	🟪	40.0000	1.6625
	☑	CAT	▨	10.1695	11.7459
	☑	DD	🟫	57.2993	0.0000
	☑	DIS	🟩	66.6667	24.2424
	☑	GE	🟦	49.0566	41.8182
	☑	GM	▨	50.0000	16.0192
	☑	HD	🟪	16.6667	1.5114
	☑	HON	▨	50.0000	45.8974
	☑	HPQ	🟫	90.9091	25.8250
	☑	IBM	🟩	50.0000	43.4109
	☑	INTC	🟦	29.4118	56.9231
	☑	JNJ	▨	35.7143	10.8696
	☑	JPM	🟪	40.0000	8.1081
	☑	KO	▨	35.0000	12.0968

MO

X = RSIndex(0, M1, 10)

Y = RangePct

X	Y	Time
59.0909	98.8636	13:54:4
59.0909	98.8636	13:54:29
59.0909	98.8636	13:54:17
59.0909	98.8636	13:54:06
59.0909	98.8636	13:53:54
59.0909	98.8636	13:53:43
59.0909	98.8636	13:53:31
59.0909	98.8636	13:53:19
59.0909	98.8636	13:53:08
59.0909	98.8636	13:52:56
59.0909	98.8636	13:52:45
59.0909	98.8636	13:52:33
59.0909	98.8636	13:52:20
59.0909	98.8636	13:52:08
0.0000	98.8636	13:51:39
0.0000	98.8636	13:51:28
0.0000	98.8636	13:51:16
0.0000	98.8636	13:51:05
0.0000	98.8636	13:50:53
0.0000	98.8636	13:50:4

● Display Series Near Cursor

○ Display Active Series

Cluster window plots a list of symbols in a 2-dimensional map using formulas. This unique NeoTicker® feature calculates an X formula and Y formula for each symbol in the list. The X and Y values of the formula becomes the coordinates of the symbol in the 2-dimensional map.

You can discover trading opportunities by visually detecting irregular behavior of stocks. Cluster window is well integrated with other parts of NeoTicker® so you can send specific symbols to charts, level 2 windows, etc.

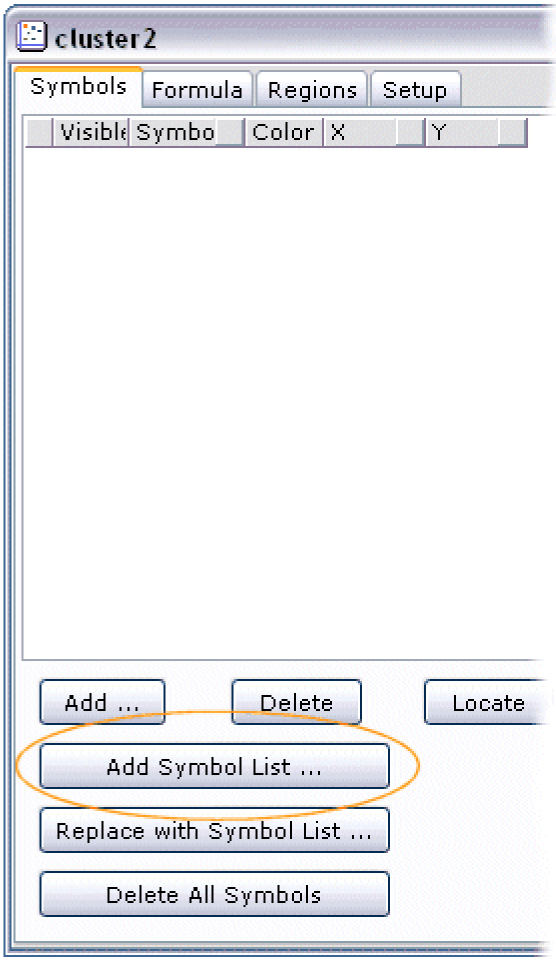
Creating a Cluster

To create a cluster, you need to provide three things:

- Symbols
- X value formula
- Y value formula

Open a cluster window by clicking on the cluster window button or by choosing **Window>New>Cluster** in the main window.

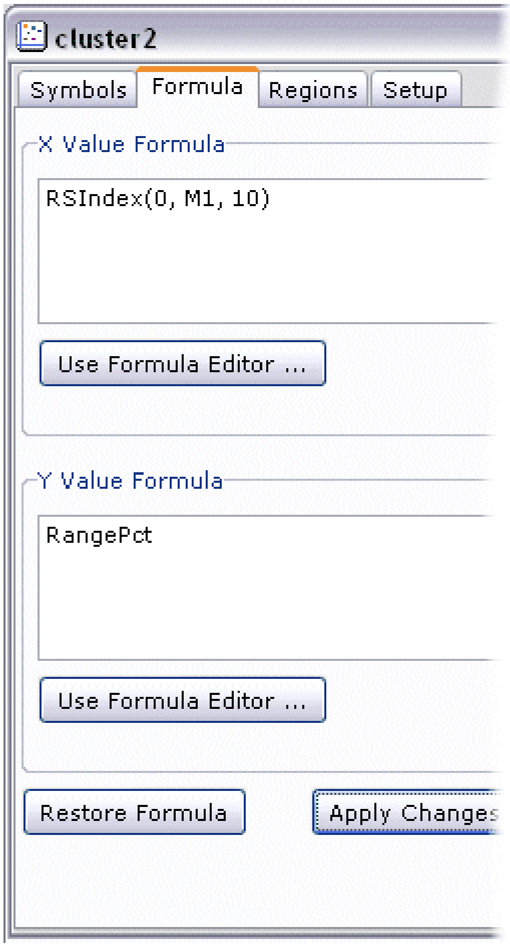
Once the cluster window is opened. Under the **Symbol** tab, press the **Add Symbol List** button. Choose a symbol list. In this example, we use the Dow 30 symbol list



You can use the **Add** button to add the symbols one-by-one, but using a symbol list is more convenient for adding many symbols.

Now press the **Formula** tab. You can specify the x and y value formula under this tab. Cluster window already provides two example formulas. In this example, we will replace the formula under **X Value Formula** with:

```
RSIndex(0,M1,10)
```



X Value Formula

```
RSIndex(0, M1, 10)
```

Y Value Formula

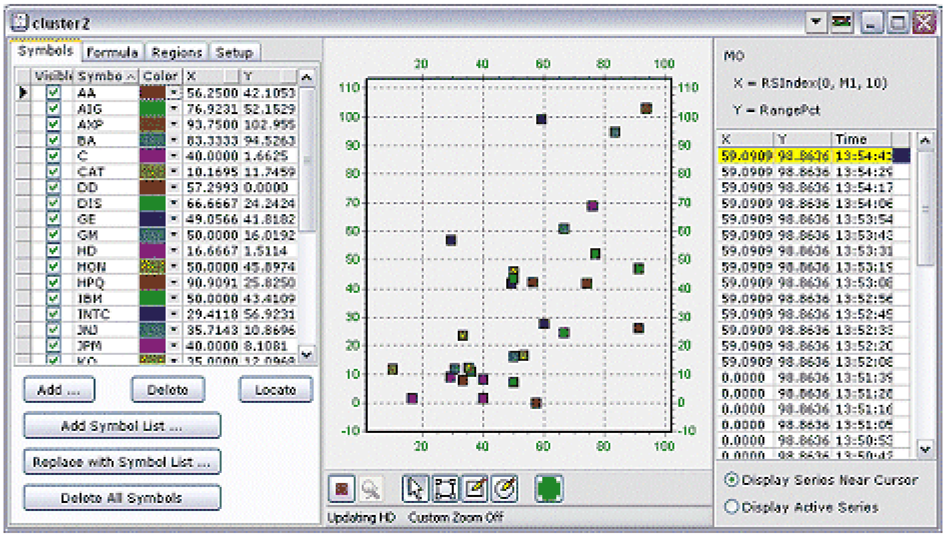
```
RangePct
```

This formula calculates 10-period 1-minute RSI indicator. Cluster window uses the same type of formula as in quote window. You can exchange formula between the two windows without modification.

Press the **Apply Changes** button. Pressing this button tells cluster window the formula changes has been finalized.

Press the start updating button ▷ on the cluster window. The cluster window will start plotting the values using the X, Y formulas as coordinates. When using cluster window, you can stop the cluster window from updating to investigate certain situation. You need to restart updating afterwards.

It can take several minutes before all symbols are updated. Cluster window needs to connect to your data vendor for real-time update and retrieves historical data for indicator calculation. The exact speed depends on your RAM and Disk Cache settings, your data vendor and market conditions. For cache settings, refer to the ***Cache Manager*** (see "Cache Manager - Introduction" on page 441) reference.



Visible	Symbo ^	Color	X	Y
☑	AA	🟫	56.2500	42.1053
☑	AIG	🟩	76.9231	52.1529
☑	AXP	🟫	93.7500	102.955
☑	BA	▨	83.3333	94.5263
☑	C	🟪	40.0000	1.6625
☑	CAT	▨	10.1695	11.7459
☑	DD	🟫	57.2993	0.0000
☑	DIS	🟩	66.6667	24.2424
☑	GE	■	49.0566	41.8182
☑	GM	▨	50.0000	16.0192
☑	HD	🟪	16.6667	1.5114
☑	HON	▨	50.0000	45.8974
☑	HPQ	🟫	90.9091	25.8250
☑	IBM	🟩	50.0000	43.4109
☑	INTC	■	29.4118	56.9231
☑	JNJ	▨	35.7143	10.8696
☑	JPM	🟪	40.0000	8.1081
☑	KO	▨	35.0000	12.0968

MO

X = RSIndex(0, M1, 10)

Y = RangePct

X	Y	Time
59.0909	98.8636	13:54:42
59.0909	98.8636	13:54:29
59.0909	98.8636	13:54:17
59.0909	98.8636	13:54:06
59.0909	98.8636	13:53:54
59.0909	98.8636	13:53:43
59.0909	98.8636	13:53:31
59.0909	98.8636	13:53:19
59.0909	98.8636	13:53:08
59.0909	98.8636	13:52:56
59.0909	98.8636	13:52:45
59.0909	98.8636	13:52:33
59.0909	98.8636	13:52:20
59.0909	98.8636	13:52:08
0.0000	98.8636	13:51:39
0.0000	98.8636	13:51:28
0.0000	98.8636	13:51:16
0.0000	98.8636	13:51:05
0.0000	98.8636	13:50:53
0.0000	98.8636	13:50:42

Cluster Tour

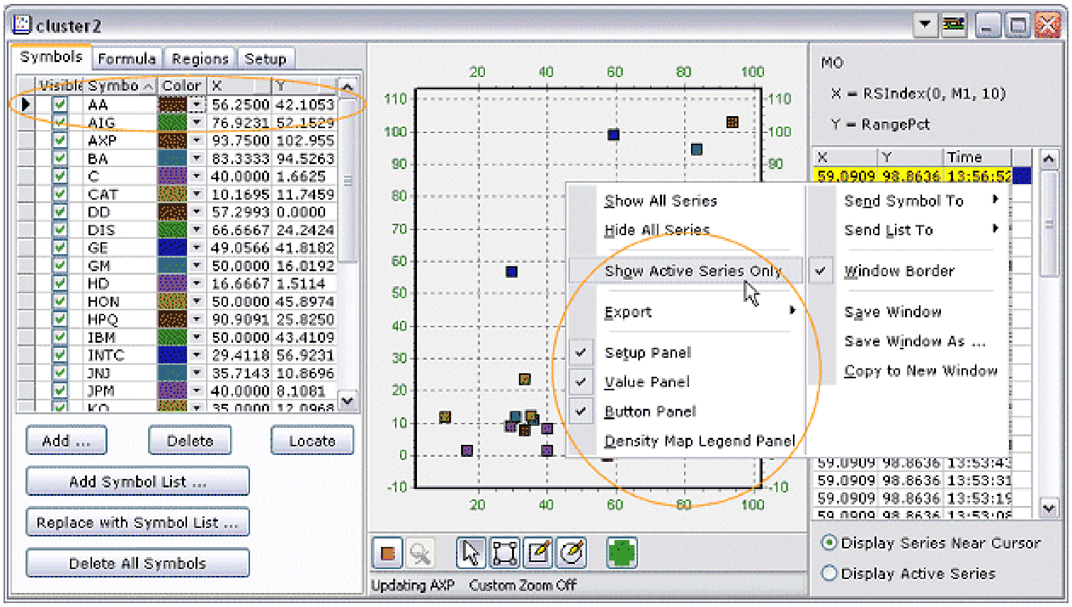
In this example, we will show you a tour of the cluster.

Press the **Symbols** tab. You can see that under **Symbols** tab are X, Y values of the formula. After you start cluster's updating, the values are constantly updating as symbols change in values. Because the X, Y values are updating, each symbol will generate a series of values in time.

Notice that to the left of one of the symbol is a right triangle ▶. The triangle marks the active series of the cluster window. The cluster window remembers the most recent symbol you work on as the active series. You can change the active series by clicking on another symbol.

You can hide all series except the active series. To do this, right click on the cluster to open the pop up menu, and choose **Show Active Series Only**. Choose it again to show all series.

If you have many series and you want to locate a particular series in the map, **Locate** button can help you. First, click on the symbol in the left to select the series to become the active series. Then, press and hold on the **Locate** button. The active series is then highlighted.



Visible	Symbo	Color	X	Y
☑	AA	▨	56.2500	42.1053
☑	AIG	▨	76.9231	52.1529
☑	AXP	▨	93.7500	102.955
☑	BA	▨	83.3333	94.5263
☑	C	▨	40.0000	1.6625
☑	CAT	▨	10.1695	11.7459
☑	DD	▨	57.2993	0.0000
☑	DIS	▨	66.6667	24.2424
☑	GE	▨	49.0566	41.8182
☑	GM	▨	50.0000	16.0192
☑	HD	▨	16.6667	1.5114
☑	HON	▨	50.0000	45.8974
☑	HPQ	▨	90.9091	25.8250
☑	IBM	▨	50.0000	43.4109
☑	INTC	▨	29.4118	56.9231
☑	JNJ	▨	35.7143	10.8696
☑	JPM	▨	40.0000	8.1081
☑	KO	▨	35.0000	12.0968

MO

X = RSIndex(0, M1, 10)

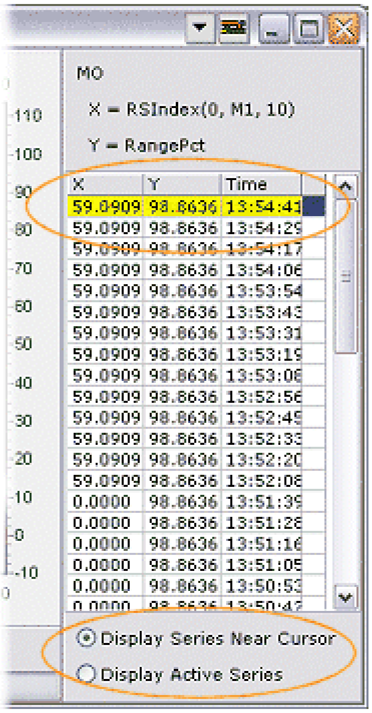
Y = RangePct

X	Y	Time
59.0909	98.8636	13:56:52
59.0909	98.8636	13:53:43
59.0909	98.8636	13:53:31
59.0909	98.8636	13:53:19
59.0909	98.8636	13:53:08

● Display Series Near Cursor

○ Display Active Series

In the right hand side of the cluster window is the Value Panel. Value Panel displays the values of a single series. By default the value displayed is the where your mouse points to. You can tell Value Panel to display the values of the active series by choosing **Display Active Series** in the pop up menu.



MO

X = RSIndex(0, M1, 10)

Y = RangePct

X	Y	Time
59.0909	98.8636	13:54:41
59.0909	98.8636	13:54:29
59.0909	98.8636	13:54:17
59.0909	98.8636	13:54:06
59.0909	98.8636	13:53:54
59.0909	98.8636	13:53:43
59.0909	98.8636	13:53:31
59.0909	98.8636	13:53:19
59.0909	98.8636	13:53:08
59.0909	98.8636	13:52:56
59.0909	98.8636	13:52:45
59.0909	98.8636	13:52:33
59.0909	98.8636	13:52:20
59.0909	98.8636	13:52:08
0.0000	98.8636	13:51:39
0.0000	98.8636	13:51:28
0.0000	98.8636	13:51:16
0.0000	98.8636	13:51:05
0.0000	98.8636	13:50:53
0.0000	98.8636	13:50:42

● Display Series Near Cursor

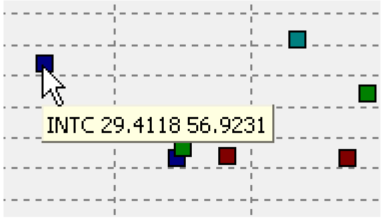
○ Display Active Series

This brings to the panels of the cluster window. The cluster window is divided into multiple panels to display different information. You can turn on/off the panels by using the pop up menu entries: Setup Panel, Value Panel, Button Panel and Density Map Legend Panel.

You can zoom into an area in the cluster window by click and hold the mouse on a point, then move the mouse to the lower left and release the button to specify a zoom region.

Use the un-zoom button 🔍 to reset to the default zoom level.

If you leave the mouse on a point in the map, the x, y value of the point is shown.



INTC 29.4118 56.9231

Hiding and Show Series

Individual series visibility can be set by toggling the **Visible** option in **Symbols** tab in setup panel.

You can hide and show series with the pop up menu. The following options are available:

Item	Function
Show All Series	Turns the visibility of all series on
Hide All Series	Turns the visibility of all series off
Show Active Series Only	When checked, only display the active series in the cluster, ignoring the visibility settings of the other series

Hiding and Showing Different Panels

Cluster window is divided into different panels. You can use the pop up menu to hide and show all panels except the cluster itself.

Locating a Series in Cluster

With so many markings in cluster window, it can become difficult to locate a series in a cluster.

The **Locate** button in **Symbols** tab in setup panel can help you.

1. Make sure the setup panel is visible and **Symbols** tab is pressed.
2. Click on the series you want to locate. This will make the series active.
3. Press and hold the **Locate** button.

A crosshair is shown in the cluster to help you locate the series.

Zooming

In the map area, you can zoom into an area by click-and-dragging the mouse towards the lower right direction. Once you zoomed into a region, the scaling for the map is "locked". Cluster window will not update the x and y axis values. The message "Custom Zoom On" will be shown in the status bar.

To un-zoom, either click-and-dragging the mouse towards the upper left direction, or press the un-zoom button 🔍.

Visual Settings

Color and Grid Settings

Color and grid settings can be changed:

1. Make sure the setup panel is visible and **Setup** tab is pressed.
2. Change the settings.

Dot Size

To change series dot size:

1. Make sure the setup panel is visible and **Setup** tab is pressed.
2. Press the **More Options** button.
3. Change **Series Dot Size** under the **Visual** tab.

Setup Panel Reference

Symbols Tab

You can manage the symbols cluster window is tracking under the **Symbols** tab.

Item	Function
Visible column	Toggles series visibility
Symbol column	Symbol of the series
Color column	Series color
X column	X formula value of the series
Y column	Y formula value of the series
Add button	Adds a series
Delete button	Deletes a series
Locate button	Press and hold this button to locate a series in the map
Add Symbol List button	Adds a symbol list to the current series
Replace with Symbol List button	Replaces the current series with a symbol list
Delete All Symbols button	Deletes all series

Formula Tab

The X, Y formula for the coordinates are defined in this tab.

Item	Function
X Value Formula	Formula that calculates the X value. You must press the **Apply Changes** button to be effective.

Y Value Formula	Formula that calculates the Y value. You must press the **Apply Changes** button to be effective.
Use Formula Editor button	Launches formula editor to edit the formula.
Restore Formula button	Restores X, Y formula to previously applied values.
Apply Changes button	Applies the changes in the edit area to the X, Y formula.

Regions Tab

This tab is for defining regions to track series. For more information about regions, refer to ***Regions and Alerts*** (on page 489).

Item	Function
Name column	Region names
Alert column	Whether alert is enabled for the region.
Select Region button	Selects the region that is clicked on the region list. This is an alternative to select an region by directly clicking on the region.
Name button	Changes the name of the region.
Region Alerts button	Sets up region alert.
Track Region button	Tracks the region. When a region is tracked, the series is highlighted in yellow. The series that are inside and outside of the tracked region are displayed.
Delete button	Deletes a region.

Setup Tab

Under this tab is the various setup of cluster.

Item	Function
Chart Color	Color scheme.
Grid	Whether to show horizontal and vertical grid lines.

Fading series color When more than one dot per series are shown, whether to fade the color of the dots to background color.

Time Frame Time frame used for indicator calculation in X, Y formula.

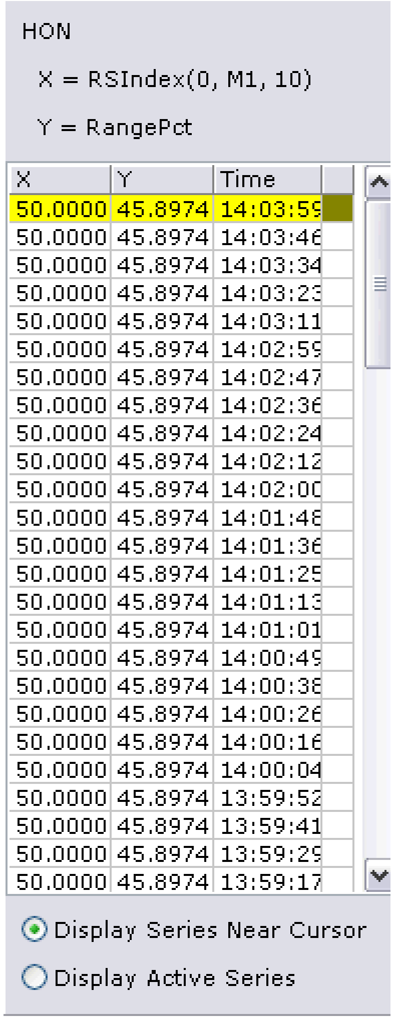
Time Frame Manager Opens time frame manager.

Reset to Default Resets the values in this tab to default.

More Options Shows more options.

Value Panel Reference

Value panel focus on the values of a single series.



HON

X = RSIndex(0, M1, 10)

Y = RangePct

X	Y	Time
50.0000	45.8974	14:03:59
50.0000	45.8974	14:03:46
50.0000	45.8974	14:03:34
50.0000	45.8974	14:03:23
50.0000	45.8974	14:03:11
50.0000	45.8974	14:02:59
50.0000	45.8974	14:02:47
50.0000	45.8974	14:02:36
50.0000	45.8974	14:02:24
50.0000	45.8974	14:02:12
50.0000	45.8974	14:02:00
50.0000	45.8974	14:01:48
50.0000	45.8974	14:01:36
50.0000	45.8974	14:01:25
50.0000	45.8974	14:01:13
50.0000	45.8974	14:01:01
50.0000	45.8974	14:00:49
50.0000	45.8974	14:00:38
50.0000	45.8974	14:00:26
50.0000	45.8974	14:00:16
50.0000	45.8974	14:00:04
50.0000	45.8974	13:59:52
50.0000	45.8974	13:59:41
50.0000	45.8974	13:59:29
50.0000	45.8974	13:59:17

● Display Series Near Cursor

○ Display Active Series

The top area of the value panel displays the following information:

- the symbol which the value panel is displaying values for
- X and Y formula that are currently in-use

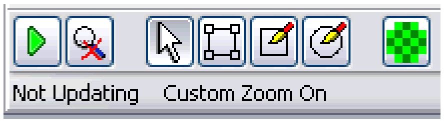
The value list displays the X, Y value and the time stamp the X, Y values are calculated. A color legend is provided to show what color is used by the X, Y value. Note that when there are more than one dot per series, the color may be different for all dots.

The following options are available for value panel:

Display Series Near Cursor - When checked, value panel will display the values of the series that is near the mouse position in the map.

Display Active Series - When checked, value panel will display the values of the active series. Active series is the series you have clicked on.

Button Panel Reference



Button panel provides tool buttons and status information.

The top area of the button panel contains a list of buttons that carries out certain actions. The bottom area of the button panel is a status bar showing the status of the cluster window.

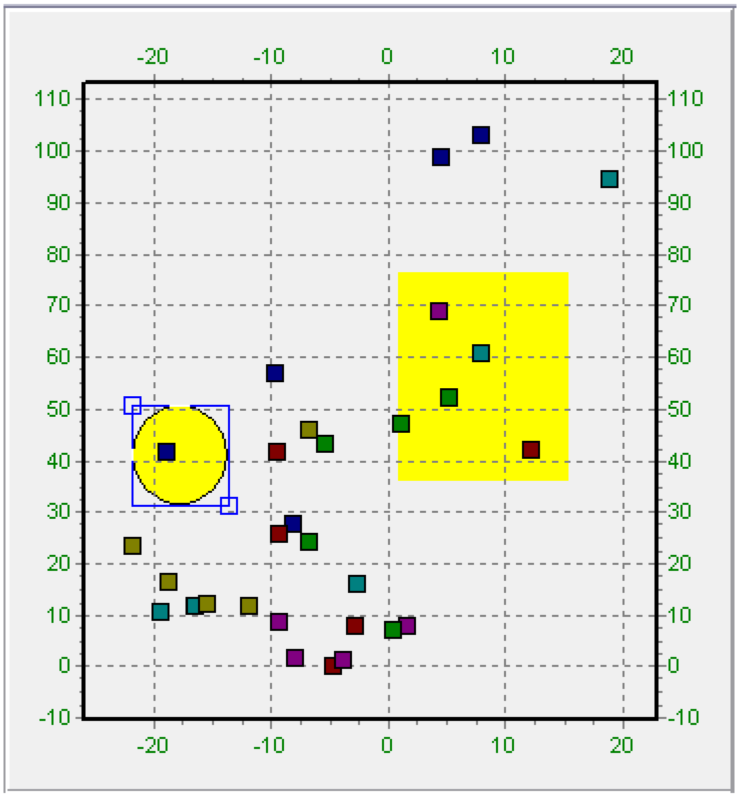
Button	Function
▷	Starts/stops cluster from updating
🔍	Un-zooms
↖	Normal cursor. When pressed, you can zoom under this cursor, but cannot select region
⛶	Press this button to select a region by mouse click
✎	Draws a rectangular region
✎	Draws an oval region
▦	Toggles density map display

Regions and Alerts

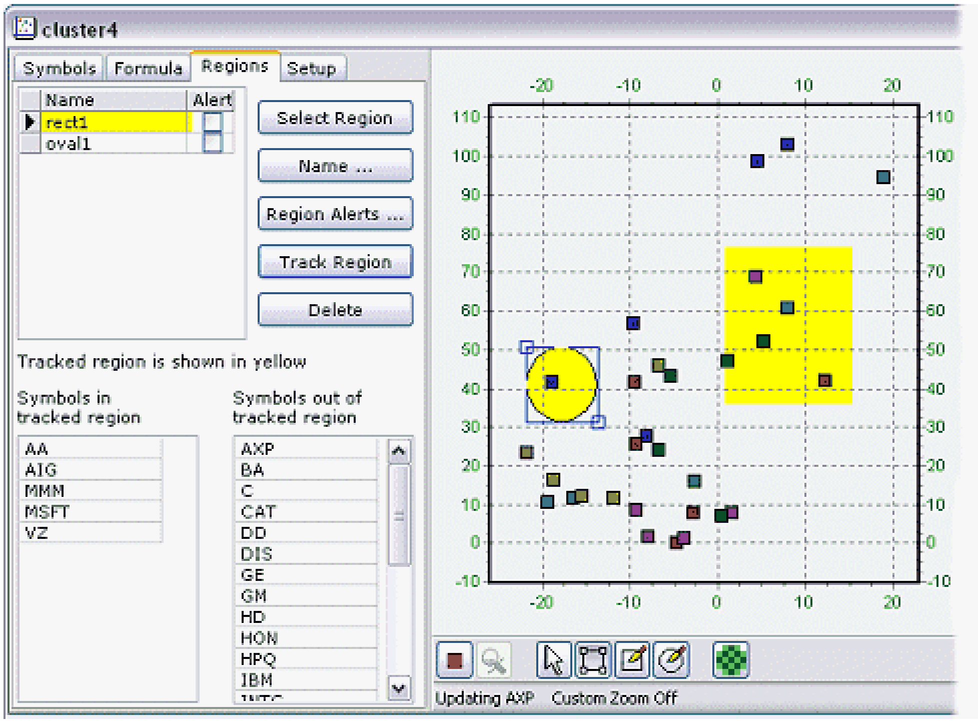
You can define regions in cluster window. A region is an area in the cluster window that helps tracking symbols. You can also set up alerts using regions.

To create a region, press one of the region buttons to create rectangular or oval region. Then in the cluster, click, drag and release the mouse on the cluster window to define the region.

Once the region is drawn, it is shown in yellow on the cluster. To modify a region, first press the select region edit button . Then you can click on a region to select it. A boundary of the region will be shown and you can move or resize the region. Press the normal cursor button to finish region editing.



You can quickly tell what symbols are in a region and what are not. Press the **Regions** tab in the left. If you have a region defined, you will see it listed under this tab. Click on the region you want to tell what symbols are in and out, and press the **Track Region** button.



Name	Alert
rect1	□
oval1	□

Tracked region is shown in yellow

Symbols in tracked region

AA
AIG
MMM
MSFT
VZ

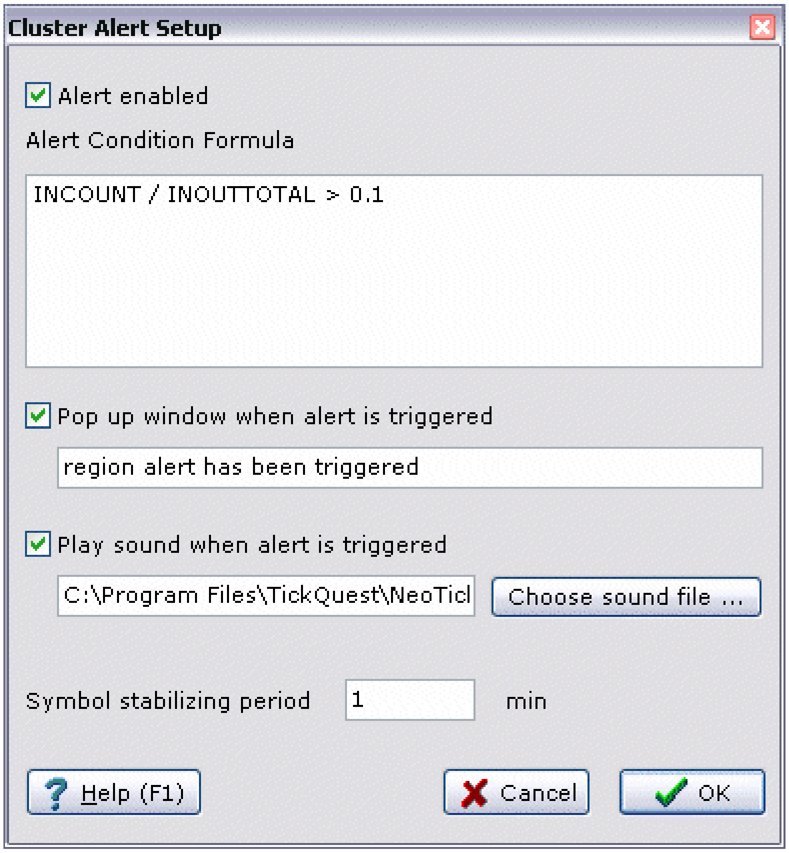
Symbols out of tracked region

AXP
BA
C
CAT
DD
DIS
GE
GM
HD
HON
HPQ
IBM
INTC

Updating AXP Custom Zoom Off

Once a region is tracked, the region's name is shown in yellow in the region list, and symbols that are in and out of the tracked region is listed in the boxes below the region list.

You can set up alerts that are based on regions. Select the region you want to set up alert, and press the **Region Alerts** button. Cluster window will open a dialog to let you enter the information for the alert.


Cluster Alert Setup

☑ Alert enabled

Alert Condition Formula

INCOUNT / INOUTTOTAL > 0.1

☑ Pop up window when alert is triggered

region alert has been triggered

☑ Play sound when alert is triggered

C:\Program Files\TickQuest\NeoTicl

Symbol stabilizing period 1 min

Let's talk briefly how region alert works. You provide an alert condition for a region. When cluster window is updating, each symbol will be evaluated against the alert condition. If the condition becomes true, alert is triggered.

Symbol stabilizing period parameter tells the alert to exclude the alert condition evaluation for symbols that are not in the cluster for long enough. For example, if the stabilizing period is one minute, symbols that are in the cluster for less than one minute will not trigger the alert.

Alert condition is specified by formula. Alert formula is similar to regular formulas. In addition, cluster alerts support the following functions to help you construct alerts in cluster:

Cluster alert only tracks what's inside and outside of region. It does not evaluate indicators like the x, y value formula.

Here are the functions you can use inside cluster alert formula:

`INREGION(symbol)` - returns true when the series specified by symbol is in the region

`ENTERREGION` - returns true when any series enters the region. By entering, the series must first be outside of the region, then moves in.

`ENTERREGION(symbol)` - returns true when the series specified by symbol enters the region. By entering, the series must first be outside of the region, then moves in.

`LEAVEREGION` - returns true when any series leaves the region. By leaving, the series must first be inside of the region, then moves out.

`LEAVEREGION (symbol)` - returns true when the series specified by symbol leaves the region. By leaving, the series must first be inside of the region, then moves out.

`INCOUNT` - returns the number of series in the region

`INOUTTOTAL` - returns the total number of series that are inside and outside of the region. In out total does not include series that are not updated yet. Non-updating series are usually due to bad symbol or slow data feed issues.

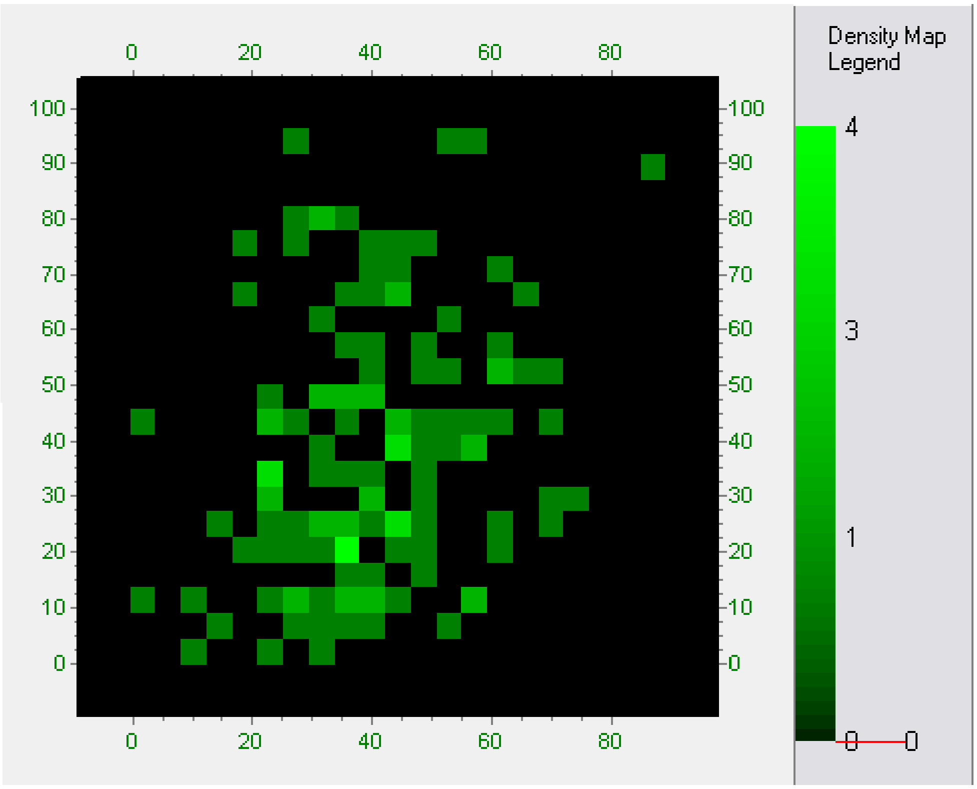
Do not use INREGION, INCOUNT and INOUTTOTAL as the sole condition for the alert. These function will return true or a value larger than 0 without any resetting mechanism. This may cause repeat triggering of alerts.

Density Map

Density map lets you visualize the density of the cluster. To enable density map, press the density map button ▦.

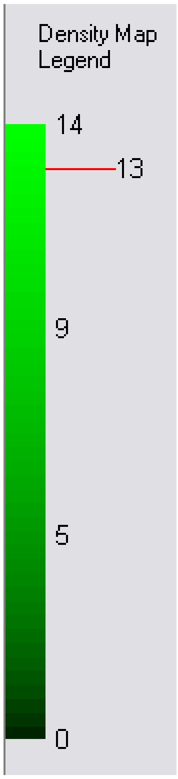
Density map is active only if cluster window is updating.

The dark area in the density map represents a low density area and the bright color area represents a high density area. You can turn on the density map legend to help you visualize the density values: right click on the cluster to open the pop up menu, and choose **Density Map Legend Panel**.



To turn off density map, press the density map button again.

Density Map Legend



Density map legend shows vertically the density each color represents. The red line in the density map legend is the density of where your cursor points to in the cluster map.

Options

Additional density map option such as resolution and color can be set:

1. Make sure the setup panel is visible and **Setup** tab is pressed.
2. Press the **More Options** button.
3. Press the **Density Map** tab.

Use one of the color boosting option to improve color quality for different densities. Boosting option changes the gamma of the density color.

Displaying More than One Dot per Series

You can track down how a symbol moves in the cluster in time by showing more than one dot per series. This feature is best used with only a few series in the cluster, otherwise the cluster map can get very crowded. Also, trail is useful only for smoothly moving formula to avoid a jumpy series.

Here is an example:

1. Under the **Symbols** tab, remove all series from the cluster by pressing the **Delete All Symbols** button. Press the **Add** button and enter a symbol to add a single series. In this example, we use `MSFT`.
2. Under the formula tab, change the **X Value Formula** to: `mov(0, M1, "Simple", 10)`
3. Change **Y Value Formula** to: `mov(0, M1, "Exponential", 10)`
4. Make sure the cluster is updating by inspecting the status area. If not, press the update button ▶.
5. Press the **Setup** tab. Press the **More Options** button. Under the **Visual** tab, change the **Dots Per Series** to 10. Press the **OK** button.

Now MSFT will be displayed in a series of 10 dots. The color is changed gradually from the full color for the most recent dot to a less saturated color for the less recent dots.

Fading Effect

Series uses a fading effect to gradually change color from the series color (for more recent dot) to background color (for earlier dots).

To toggle on/off the fading effect:

1. Make sure the setup panel is visible and **Setup** tab is pressed.
2. Toggle **Fading series color**.

Taking Screen Shots

Choose one of the item under **Export** in pop up menu to take screen shot for cluster window.

Update Frequency

By default, cluster window updates the X, Y formula values for all symbols every 10 second. This is the average time only and is not guaranteed because the formula complexity and number of series can affect update time.

To change the update frequency:

To change series dot size:

1. Make sure the setup panel is visible and **Setup** tab is pressed.
2. Press the **More Options** button.
3. Change **Update Frequency** parameter under the **Updates** tab.

Configuration Guide

Confirmation Dialogs

Certain confirmation dialogs can be annoying to expert users. You can turn off confirmation dialogs by:

1 Choose **Program>User Preference** from the main window.

2 Press the **General** tab.

3 Choose one of the settings under **Confirmation Dialogs.**

There are two types of confirmation. One type is a reminder type for actions that are reversible (e.g. closing user panel). Another type is for actions that are not reversible (e.g. closing a chart without saving).

So that makes three settings for the option:

Always ask

Always ask before taking an action.

Ask before serious actions

Ask only if the action is considered irreversible.

Do not ask

Carry out the action without asking.

Request for Dialog Confirmation

Because dialog must be changed to recognized the confirmation options. For this reason, we keep an active thread in our support forum for users to make requests for dialog behavior. To visit the thread, click ***here*** http://www.tickquest.com/forums/showthread.php?threadid=450.

Customizing Tool Menu

See ***Tool Menu*** (on page 1229).

Customizable Main Window Tool Bar

The main window tool buttons are customizable. You can re-arrange the button groups in any way you like and NeoTicker® will save the settings.

Button Groups

You can enable or disable button groups under the **View** menu in the main window.

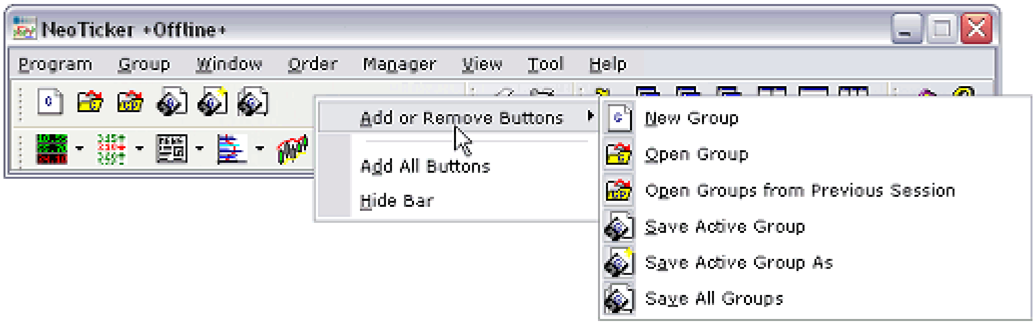
Individual Buttons

Individual tool buttons can be added or removed inside a button group using pop up menu:

1 Right click on the button group to open pop up menu.

2 Under **Add or Remove Buttons**, click on the menu item representing the button to enable/disable it, or

3 Choose **Add All Buttons** to enable all buttons.



Other Tool Bars

There are also floating customizable tool bars to help you work with charts. To learn more, go to ***Tool Bar*** (on page 1209).

Decimal Places

You can set up your prefer number of decimal place in various places. At the basic level, there is an application wide decimal places setting. This setting tells NeoTicker® the default number of decimal places to use. If you mainly trade stocks/futures, you should set application wide decimal places to 2. If you mainly trade Forex, you should set this value to 4.

Number of decimal places can be overridden in individual windows.

Application Wide

To set application wide decimal place preference:

1. In main window, choose **Program>User Preference** to open User Preference.
2. In User Preference, under General tab, set decimal places to either **Use Default** or set a customize value.

The default number of decimal places is 2.

Quote Windows

When a Quote Window is created, it will consult the application wide decimal place preference. Note that for a quote window, you can override the decimal preference for individual columns, see ***Quote Window Operation Guide, Formatting Values*** (see "Formatting Values" on page 854).

Time Chart

Time Chart decimal places are specified on a per pane basis. By default, Time Chart uses smart mode for determining decimal places. When a symbol is placed in a pane, Time Chart will first consider the type of the symbol (from data vendor), then consider application wide preference to determine the number of decimal places.

To override decimal places in a Time Chart, see ***Change Price Axis Scaling and Labelling*** (on page 1049).

Time and Sales Window

Time and Sales window considers application wide decimal place preference. To override, see ***Time and Sales Window Operation Guide, Price Format*** (see "Price Format" on page 1007).

Dynamic Table/Dynamic Grid

Dynamic Table/Dynamic Grid considers application wide decimal place preference. You can override the decimal preference on a cell level. See ***Dynamic Grid Operation Guide, Cell Level Setup*** (see "Cell Level Setup" on page 553).

Decimal Places for Orders

Order placement requires more precise decimal place settings on a per symbol basis. This is done using Symbol Info Manager. In general, decimal place setting in Symbol Info Manager will override the application wide setting.

Note that the settings in Symbol Info Manager only affects the following services:

- Order placement
- Trade Simulator
- Trading System Reporting

It does not affect other services such as charting.

To set decimal place setting in Symbol Info Manager:

1 In main window, choose Manager>Symbol Info to open Symbol Info Manager.

2 In Symbol Info Manager, if the symbol is already defined, select, otherwise, type in the symbol.

3 Specify the decimal places under **Deci / Frac**.

4 Press **Add/Replace** button.

5 Press **Save List** button.

Default Date Time Format

You can set a default date time format. Default date time format is not a single value. It is a group of formatting rules that tells NeoTicker® what date time format to use under different circumstances.

To set default date time format:

1. Choose **Program>User Preference** from the main window.
2. Press the **Date Time Format Preference** button under the **General** tab.
3. The **Date Time Format Preference** dialog is opened.
4. Each group in the dialog represents a rule you can edit.
5. Press the **OK** button to accept the changes.

Rules

Most of the rules have two choices. The first choice is **System default**. This means the format is constructed using your computer's default day time settings. The other choice is Customize. This means the format is a user specified formatting string.

Here are the rules:

Year Month Day Hour Minute Second Format

This rule is used when a complete date time is output. For example, the report area (top part) of a time chart and data view use this rule.

Month Day Hour Minute Second Format

This rule is used when a complete date time except year is required. For example, the cursor value of a chart and time and sales window use this rule.

Year Month Day Format Format

This rule is used when year month day are needed. Examples are alert time when the alert was happened not in the current trading day.

Month Day Format

This rule is used when only month and day are needed. Examples are quote window date time when the quote is not from the current trading day.

Year Only Format

This rule is used when only the year is displayed. Examples are time labels of long term charts.

Hour Minute Second Format

This rule is used when time is displayed. Examples are the update time labels of data view.

Hour Minute Format

This rule is used when second information is not needed. Examples are time labels of intraday charts.

Second Only Format

This rule is used when only seconds are displayed. Example are time labels of intraday charts.

Format String

The date time format string contains characters the tells NeoTicker® how to format a date time under a specific rule. For example, the format string `yyyy/mm/dd` will format to a date that looks like `2003/07/24`.

Here is the format string specification:

Code	Meaning
`d`	Displays the day as a number without a leading zero (1-31).
`dd`	Displays the day as a number with a leading zero (01-31).
`ddd`	Displays the day as an abbreviation (Sun-Sat).
`dddd`	Displays the day as a full name (Sunday-Saturday).
`m`	Displays the month as a number without a leading zero (1-12). If the `m` specifier immed specifier, the minute rather than the month is displayed.
`mm`	Displays the month as a number with a leading zero (01-12). If the `mm` specifier immed specifier, the minute rather than the month is displayed.
`mmm`	Displays the month as an abbreviation (Jan-Dec)
`mmmm`	Displays the month as a full name (January-December)
`yy`	Displays the year as a two-digit number (00-99).
`yyyy`	Displays the year as a four-digit number (0000-9999).
`h`	Displays the hour without a leading zero (0-23).
`hh`	Displays the hour with a leading zero (00-23).
`n`	Displays the minute without a leading zero (0-59).
`nn`	Displays the minute with a leading zero (00-59).
`s`	Displays the second without a leading zero (0-59).
`ss`	Displays the second with a leading zero (00-59).
`am/pm`	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'am' for any hou any hour after noon. The am/pm specifier can use lower, upper, or mixed case, and the accordingly.
`a/p`	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'a' for any hour hour after noon. The a/p specifier can use lower, upper, or mixed case, and the result is
`/`	Displays the date separator character

:	Displays the time separator character
`'xx'/"xx"`	Characters enclosed in single or double quotes are displayed as-is, and do not affect for

Default Time Frame

Default time frame controls time frame (trading time, etc) NeoTicker® will use when handling a symbol. This setting is overridden by settings in ***Symbol Info Manager*** (on page 921).

For example, when you create a chart, the chart needs to know the trading time of the symbol. Trading time is part of time frame information and is determined as follows (highest priority listed first):

- If there is a overriding time frame setting in the chart, the chart's setting is used
- If there is an entry for the symbol in Symbol Info Manager, the Symbol Info Manager's setting is used
- Default time frame is used

To set default time frame:

1. Choose **Program>User Preference** from the main window.
2. Under **General** tab.
3. Choose a setting for **Default Time Frame**.

Function Window Title Size

NeoTicker® supports two styles of function window title size, normal and large. Large title size is the default. To change window title size:

1. Choose **Program>User Preference** from the main window.
2. Press the **General** tab.
3. Choose **Normal** or Large under **Function Window Title Size**.
4. Restarts NeoTicker® for the settings to take effect.

Maximize and minimize buttons are not available with normal size. They are shown only with large size. However, you can right click on the title bar to access maximize and minimize functions.

Grid Computing

NeoTicker® nodes can be configured as a grid computation node for other TickQuest products. For example, it can be configured as a computation node for Grid Optimizer.

Options to configure grid computing is under **Program>User Preference**, **Grid Computing** tab.

Accept Local Commands - Accept grid computing commands from the local computer.

Accept Remote Commands - Accept grid computing commands from remote computers.

Full Speed Computation in Offline Mode - This option allows you to maximize the amount computation done by NeoTicker®. This option is not suitable if you use NeoTicker® with a real-time data feed.

TCP Port - TCP port where NeoTicker® listens for grid computing commands.

UDP Port - UDP port where NeoTicker® listens for grid computing commands.

Indicator Directory

By default, all NeoTicker® indicators are stored under the `indicator` directory in the installation, e.g. `C:\Program Files\TickQuest\NeoTicker4\indicator`. You can customize this location to another directory.

Example usage. If you have multiple NeoTicker® installed across a network for several traders, you can share a single indicator directory such that these traders will have access to the same set of indicators.

To set indicator directory:

1. In main window, choose **Program>User Preference**.
2. Press **Directories** tab.
3. Specify the indicator location in the **Indicator Directory Location** box.
4. Restart NeoTicker®.

Office Style Menu

NeoTicker® supports a modern looking office style menu. This menu system is on by default.

If you prefer the classic Windows menu style, you can turn office style menu off by:

1. In main window, choose **Program>User Preference**.
2. Press **General** tab.
3. Toggle **Office Style Menu**.
4. Restarts program.

Playing Sound

You can configure how NeoTicker® play sound, e.g. audio alerts. The options are useful if you encounter performance problems (e.g. too many audio sounds generated by alerts).

To configure sound playing options:

1. In main window, choose **Program>User Preference** to open User Preference window.
2. Choose **General** tab.
3. Select one of the options under **Sound Playing Options**.

The available options are:

- **Smart mode** - If a sound cannot be played in time, it will be dropped. Duplicate sounds are also removed.
- **Remove duplicates only** - Remove duplicate sounds.
- **All sounds in sequence** - All sounds are played.

Tick Count Break

The tick count is reset at the beginning of each trading day. You can tell NeoTicker® not to break tick driven bars at the end of the trading day.

1. Choose **Program>User Preference** from the main window.
2. Press the **Real-Time Data** tab.
3. The option is **Break on Day Change**.

If this option is off, tick count will start at the beginning of data loaded with no break.

XP Theme Support

On Windows XP, NeoTicker® supports XP theme user interface, i.e. If Windows is set to XP theme, NeoTicker® will use XP theme. If Windows is set to classic theme, NeoTicker® will use classic theme.

You can force NeoTicker® to use classic theme under Windows XP theme:

1 In main window, choose **Program>User Preference**.

2 Press **General** tab.

3 Toggle **XP Theme Support**.

4 Restarts program.

Continuous Contract Manager

Pre-defined Continuous Contracts

The following symbols are pre-defined continuous contracts. You can use these continuous contract symbols just like any other symbols.

```
#ER2
#ES
#ND
#NQ
#SP
#US
#YM
```

Continuous contracts automatically resolve to underlying contract(s) based on the current date. So if an underlying contract is not available, the continuous contract symbol will not work. See the data vendor sections below for more information.

Continuous contracts work seamlessly if you use the same vendor for data and order placement, i.e. you can use the same symbol to request data and place orders. The contract will translate automatically to the underlying contract.

If you use separate vendors for data and order placement, you will need to use Symbol Info Manager to define order routing for the continuous contract symbol.

Define Your Own Continuous Contract

Continuous contracts are resolved on NeoTicker® side, so you can define your own contract. Continuous contracts symbols takes the following form:

```
#symbol
```

For example,

#ES - ES continuous contract.

#NQ - NQ continuous contract.

#US - US Bond continuous contract

You can define continuous contracts by using Continuous Contract Manager. We will illustrate how it works by defining ER2 using QCharts symbology. Note that in QCharts, ER2's symbols is MR. The continuous contract defined will translate #ER2 automatically to MR05Z when the current contract is December 2005.

1 In Main Window, choose **Manager>Continuous Contract** to open Continuous Contract Manager.

2 In **Continuous Contract** field, enter the symbol for the continuous contract. You do not need to specify the # sign. So enter ER2 here.

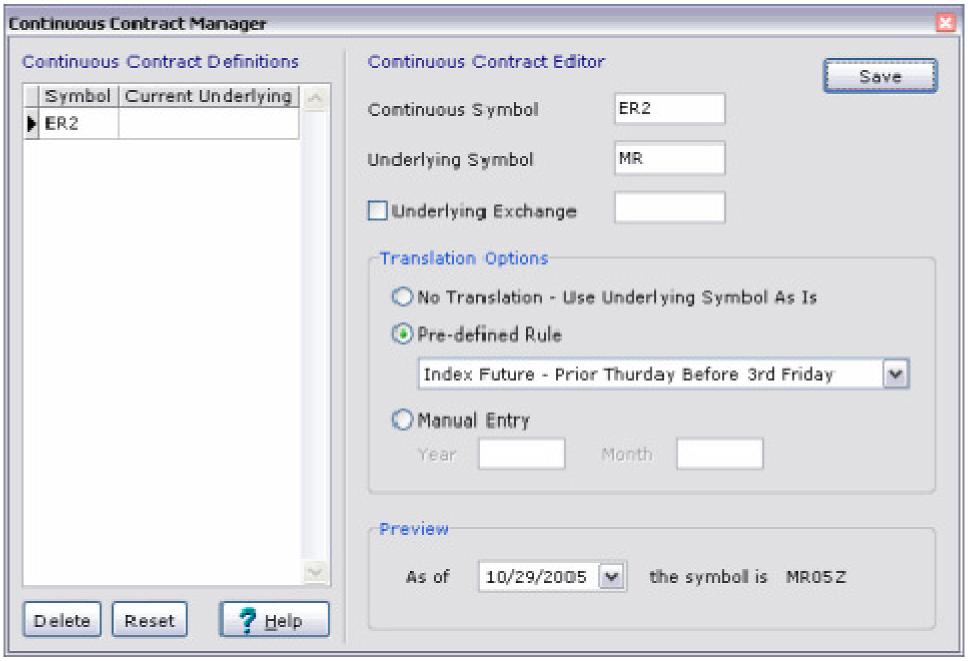
3 In **Underlying Symbol** field, enter the underlying symbol for the contract. So enter MR here. You do not need to specify the current contract (i.e. 05Z) here. Note that for some data vendors, you will need to specify a prefix. See the data vendor sections below for more information.

4 **Underlying Exchange** field is required by some data vendors. In our ER2 example, there is no need to specify the underlying exchange. See the data vendor sections below for more information.

5 Choose one of translation options. Choose **No Translation** if you want to use the symbol as is i.e. no current contract and exchange information will be added to the symbol. Choose **Pre-defined Rule** if you know how the contract rollover. Choose **Manual** if none of the pre-defined rules matches the underlying contract. In our ER2 example, choose the **Index Future** pre-defined rule.

6 You can use the **Preview** to see how the continuous contract symbol is translated.

7 Press the **Save** button to save the continuous contract definition.


Continuous Contract Manager
Continuous Contract Definitions
Continuous Contract Editor
Save
Delete
Reset
Help

Symbol	Current Underlying
ER2	

Continuous Symbol: ER2

Underlying Symbol: MR

□ Underlying Exchange:

Translation Options

○ No Translation - Use Underlying Symbol As Is

● Pre-defined Rule

Index Future - Prior Thurday Before 3rd Friday

○ Manual Entry

Year Month

Preview

As of 10/29/2005 the symbol is MR05Z

eSignal

eSignal does not require a prefix when specifying underlying symbol.

If the underlying eSignal symbol does not require an exchange specification (e.g. ES Z5), you do not need to specify the exchange in Continuous Contract Manager.

If the underlying eSignal symbol requires an exchange specification (e.g. AX Z5-DT), you need to specify the exchange in Continuous Contract Manager. You do not need to specify the dash sign (e.g. DT, not -DT).

Interactive Brokers

IB does not require a prefix when specifying underlying symbol.

IB requires specifying the underlying exchange. Use TWS to figure out the underlying exchange for a contract, for example, GLOBEX for contracts (you do not need to type in a colon).

IQFeed

IQFeed requires you to specify a prefix when specifying underlying symbol. In IQFeed, some contracts has a / prefix and some don't. Contact IQFeed support for exact symbol for the contract.

IQFeed does not require specifying underlying exchange.

MB Trading

The following pre-defined symbols are not available with MB Trading - ND, SP, US.

MB Trading does not require a prefix when specifying underlying symbol.

MB Trading does not require specifying underlying exchange.

QCharts

QCharts does not require a prefix when specifying underlying symbol.

QCharts does not require specifying underlying exchange.

RealTick

RealTick does not require a prefix when specifying underlying symbol.

RealTick does not require specifying underlying exchange.

Data Related Operations

Trouble Shooting

Real-time data feeds are not perfect. From time to time, you will encounter corrupt and missing data. This section will help you trouble shoot and fix corrupted data.

Symptom - Holes in time chart

First, make sure the holes are not caused by holiday and trading time. For these, you should specify holiday list and adjust trading time.

If holiday and trading time is not the cause, you can point the mouse to the area you suspect there is a data corruption, and press the F11 key. This method will fix missing/corrupted data in disk cache. Disk cache is used for all Internet data feed, and used optionally for satellite data feed.

If this method fails, try to remove the RAM Cache temporary file. Refer to ***Removing RAM Cache Temporary Files*** (see "Fixing RAM Cache Corruption" on page 467) for the procedure. This method will fix data corruption problem due to abnormal exit of NeoTicker®.

Symptom - Poorly updating real-time data (delays, drop ticks, frequent disconnection, etc)

Choose from main window, **Program>Server Disconnect/Reconnect**.

This method will fix issues with real-time update. The cause of the problem is NeoTicker® is connected to a stale server, and by re-connecting, there is a good chance you will switch to another server. This method is useful for Internet data feed only.

If you use eSignal, you can also try to exit eSignal's Data Manager.

Symptom - It takes a long time for Indicators in Quote window to return values

First make sure you are connecting to a good server.

Then, does the indicator implicitly require loading of lots of data?

Quote window formula relies on the RAM Cache for indicator calculation. The RAM Cache has a setting for number of data bars to request. The default value is 250 bars. Therefore, a 60-min indicator (e.g. bband(M60, 0, 0)) will require 60 * 250 = 15000 minutes of data = 38 days of minute data. This would be a slow operation for any Internet data feed.

Solution is to adjust the default value to a smaller size. See ***Adjusting RAM Cache Size*** (see "Bars Per Series" on page 464).

Symptom - No data in chart

One obvious cause is your data vendor is out. In this case, you will only get disk cache data, i.e. all your charts will have no data or have large gaps in them.

Another cause is the symbol is no longer valid. This can happen to stocks if there is a name change (e.g. QQQ was renamed to QQQQ in 2005), also to expired futures.

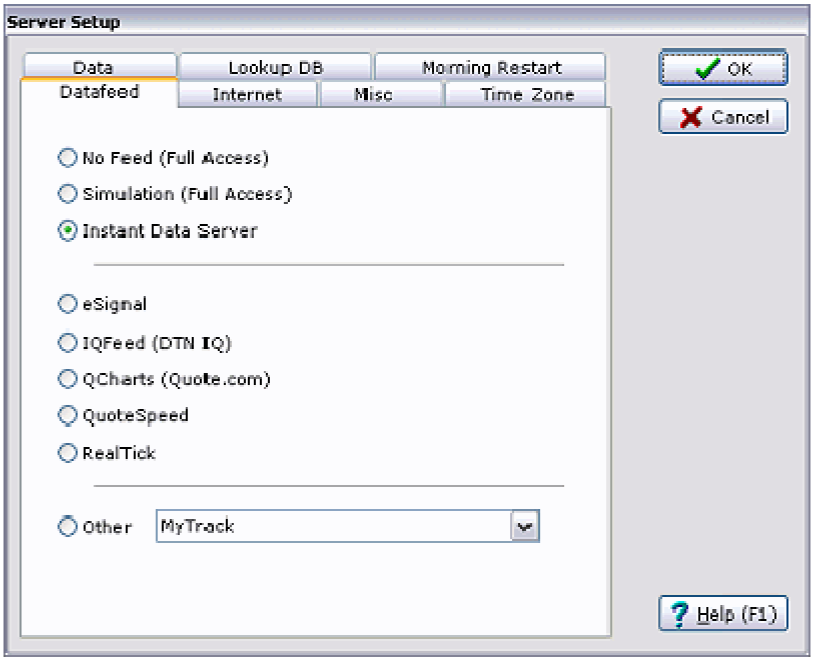
Finally, you could be running out of memory. In this case, you will notice a "Out of Memory" message in the spinning dialog when loading the data. Also, a memory error log is recorded in **Program>System Log**, under **Error Events**.

Server Setup Reference

Server Setup is for configuring NeoTicker® to handle data server. Information set as log in, look up settings is under this dialog.

To open Server Setup, choose **Program>Server Setup** from main window.

Datafeed Tab



○ No Feed (Full Access)
○ Simulation (Full Access)
● Instant Data Server

○ eSignal
○ IQFeed (DTN IQ)
○ QCharts (Quote.com)
○ QuoteSpeed
○ RealTick

○ Other MyTrack

This tab lets you select the data feed NeoTicker® talks to. The selection depends on how you intend to use NeoTicker®.

Simulation

If you are new to trading, consider setting up NeoTicker® as a simulator by choosing **Simulation**. See ***Setting up NeoTicker® as Simulator*** (on page 9) to get started.

Real-Time Trading

To trade in real-time using NeoTicker®, you will need a real-time data feed. Many vendors provides real-time data for a fee. Alternatively, you may use your brokerage account as a feed.

If you need help choosing a data feed, see ***Choosing A Data Feed*** (on page 16).

If you already have a real-time data subscription, the following table will help you quickly set up NeoTicker® to receive real-time data from your service.

Data Service / Broker	Steps	Notes	Additional Information
eSignal	1 Choose **eSignal.** 2 Press **OK** button.		***Configuring eSignal Conne***
IQFeed / DTN.IQ	1 Choose **IQFeed (DTN IQ)**. 2 Press **OK** button.	IQFeed Client required.	***Configuring Instant Data S Feed*** (on page 21)
QCharts	1 Choose **QCharts (Quote.com)**. 2 Go to **Log in** tab to fill in user id and password. 3 Press **OK** button.		***Configuring QCharts (Quo*** page 39)
QuoteSpeed	Follow steps in ***Configuring QuoteSpeed*** (on page 29).		***Configuring QuoteSpeed*** (o
RealTick	1 Choose **RealTick.** 2 Press **OK** button.		***Configuring RealTick*** (on p
MyTrack	1 Choose **Other**. 2 Select **MyTrack**. 3 Go to **Log in** tab to fill in user id and password. 4 Press **OK** button.		***Configuring myTrack Conn***

MB Trading	Follow steps in ***Configuring MB Trading as Real-Time Data Feed*** (on page 33).		***Configuring MB Trading a*** page 33)
EFX Group	Follow steps in ***Configuring EFX Group as Real-Time Data Feed*** (on page 36).		***Configuring EFX Group as*** page 36)
Interactive Brokers	Follow steps in ***Configuring Interactive Brokers as Real-Time Data Feed*** (on page 23).		***Configuring Interactive Bro Feed*** (on page 23)

Off Line Research and Analysis (No Feed)

The **No Feed** option is for advanced users who want to use NeoTicker® for off line research and analysis. If you choose this option, you can access historical data from TickQuest's Backfill Server. See ***Backfill Tab*** (on page 524) for options.

Additional Information

See ***Choosing and Configuring a Real-time Data Feed*** (on page 15).

Symbols Tab

If your data feed does not have a pre-defined symbol limit, this tab is not visible.

If NeoTicker® can automatically retrieve your symbol limit, options under this tab are disabled.

Max Symbols is the maximum number of symbols NeoTicker® will request from your data vendor for real-time streaming.

Max Options is the maximum number of option symbols NeoTicker® will request from your data vendor for real-time streaming.

Set options under this tab if:

- You encounter data problems because you routinely request more symbols than your subscription limit, or
- You have multiple applications running on the same feed and you need to restrict the symbols used by NeoTicker®

You should set **Max Symbols** and **Max Options** below the symbol limit of your data feed subscription. The values in this tab specify the upper limit of the number of symbols NeoTicker® will try to request from your data feed.

Login Tab

This tab is not visible if your data feed does not require you to log in.

You can specify your user id and password under this tab.

You will need to **Retype Password** to confirm it.

If you have difficulties entering password without seeing it, press the **Enter Password Without Blocking** button to open a dialog to let you enter your password with you seeing it.

The **Retries Before Disconnect** parameter tells NeoTicker® how many times to retry when the login fails before NeoTicker® gives up.

Internet Tab

This tab is not visible if you are not using an Internet data feed.

This tab is not applicable to Interactive Brokers as a data feed, or any data feed that is configured using NeoTicker®'s DDE feed interface.

The **Timeout** value is the time NeoTicker® will give up after making a data request, for example, requesting for a symbol. Timeout is necessary because there is no guarantee an answer will be returned for any Internet request.

If you have a slow Internet connection, you may want to increase the parameter to avoid premature timeout.

The **Auto Reconnect** check box tells NeoTicker® to automatically reconnect to the data feed when a connection is dropped.

Misc Tab

Ignore case for symbols

Leave this check on.

On Datafeed Reconnect, Default To Complete Reload

When reconnecting to a real-time data feed, a dialog will show up to ask you how you want to handle the reconnecting (see ***Connecting to a Server*** (on page 571)). If you are attending to NeoTicker®, you can choose the options with discretion.

If you plan to run NeoTicker® unattended for a long period of time (e.g. with an automatic trading system running), you will want to choose a default re-connection option here that is best suit to your need. The settings here allow you to do that.

The basic idea is you want to default to complete reload historical data only when necessary. Complete reload is a timely operation that can hold up other operations.

The following default settings are available:

- **Always** - always default to complete reload.
- **Never** - never default to complete reload, i.e. reconnecting only.
- **Only If Disconnected For More Than N Min** - default to complete reload only if the disconnection period is longer than the specified minutes. Use this option if a data gap longer than certain amount of time is deemed unacceptable.
- **Except During** - do not default to complete reload during the specified time period, e.g. you can set this up to avoid complete reload during trading hours.

Note that these are default settings. If you are present during reconnecting, you can override the default manually.

Time Zone Tab

This option defines the time zone that NeoTicker® will use to display the data. The default is **Current Windows Time Zone Settings**.

NeoTicker® has the ability to display data received from any data vendor in any time zone you want. All settings like quote update time to trading hour of a time chart are affected.

For example, if you set the time zone to **New York**, then the regular trading hours for the stocks listed in NYSE is 9:30 AM to 4:00 PM. If you switch to use **Chicago** time, then the trading hours for the same stocks is 8:30 AM - 3:00 PM.

Current Windows Time Zone Settings

NeoTicker® will obtain the current time zone information from Windows. If you change the time zone information using the Windows Date and Time Properties, NeoTicker® will change its time zone to that new settings on next ***Morning Restart*** (see "Morning Restart Tab" on page 522) or next time you launch NeoTicker®

Selected City

You can choose to display data using the time zone of a specific city. This way you can ensure your data will be displayed consistency without being affected by your computer locale setting.

Morning Restart Tab

Options under this tab controls the time when NeoTicker® performs a morning restart.

Set the options under this tab if you plan to run NeoTicker® continuously for several days.

Morning restart is the time NeoTicker® clears up its usage of your computer resources. Morning restart eliminates possible memory problems and allows NeoTicker® to run continuously. By default, NeoTicker® performs the morning restarts at 5:00am from Monday to Friday.

Morning restart should be set to a time you do not plan to actively use NeoTicker®.

Data Tab

Enable Real-Time Bid/Ask Tick Update

Enable this check box to receive bid/ask data.

Use News Server

Enable this check box will allow the news function window to receive streaming news from your data vendor.

Availability of streaming news depends on your data vendor and your subscription.

Enable disk cache integration with broadcast feeds

With a broadcast feed, i.e. through IDS, allow reading of disk cache and integrate the data with IDS data.

Daily Data Integration

NeoTicker® allows you to integrate EOD data from an EOD data vendor seamlessly with real-time intraday data. Precedence is given to EOD data from your real-time data vendor, but if there is missing EOD data from your real-time data vendor, the missing data will be retrieved from your EOD data vendor. This feature allows you to construct long term indicators in your real-time charts.

Use Datafeed Data Only

Only use data from the data feed. No data integration is performed. This is the default.

Auto Merge with Internet Data

Integrate with EOD data retrieved from Internet websites.

Auto Merge with Quotes Plus

Integrate with EOD data retrieved from Quotes Plus. You must be a Quotes Plus subscriber for this feature to work.

Auto Merge with TC2005

Integrate with EOD data retrieved from Worden Brothers, Inc.'s TC2005. You must be a TC2005 subscriber for this feature to work.

Lookup DB Tab

The Lookup DB is a table that helps NeoTicker® performs certain real-time calculations.

Caution: Use **Enable RT lookup DB** with caution. This option can significantly degrade start up performance for users of Internet data feeds.

Enable EOD Lookup DB

Whether the EOD lookup DB will be built. Some features (such as quote window's multi-period new high and new low) require EOD lookup DB to work. Enable this option if you want to use these features. Enabling this option slows down program restart. The default is enabled. You will need to specify the number of trading days NeoTicker® looks back to construct the EOD lookup table. The default is 20 days.

Enable RT Lookup DB

Whether the RT lookup DB will be built. Some features (such as quote window's tick count) require RT lookup table to work. Enable this option if you want to use these features. Enabling this option slows down program restart. The default is enabled.

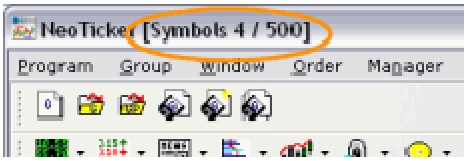
Backfill Tab

See ***Setting Up Backfill*** (on page 541).

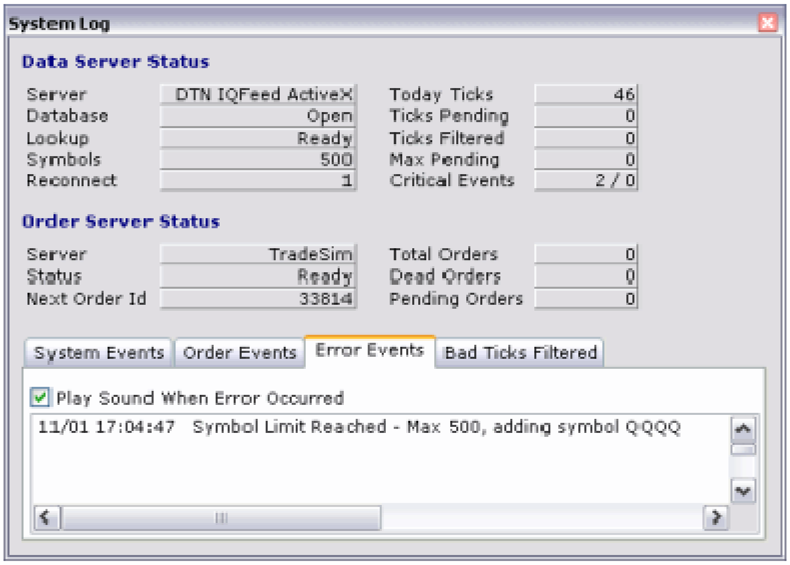
Symbol Limit

Most internet-based data vendors impose a symbol limit, i.e. the number of symbols you can receive streaming real-time data simultaneously.

In NeoTicker®, you can observe the current symbol usage and symbol limit in Main Window's caption bar. In the figure below, the first number is the current symbol usage. The second number is the limit.



NeoTicker® will refuse to track a symbol if you are at your symbol limit. When you attempt to go over the limit, NeoTicker® will produce a "bong" sound and you can see a symbol limit error message in System Log, under the **Error Events** tab. See ***Observing Real-Time Performance with System Log*** (on page 525) for information about System Log.



Data Server Status

Server	DTN IQFeed ActiveX	Today Ticks	46
Database	Open	Ticks Pending	0
Lookup	Ready	Ticks Filtered	0
Symbols	500	Max Pending	0
Reconnect	1	Critical Events	2 / 0

Order Server Status

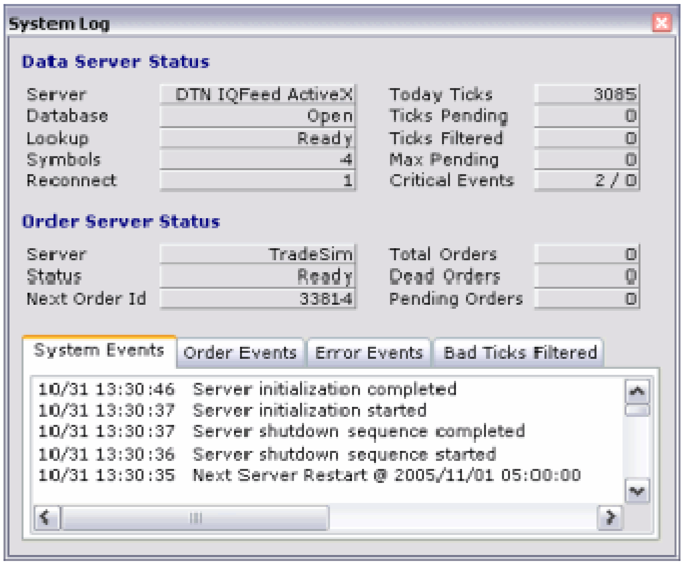
Server	TradeSim	Total Orders	0
Status	Ready	Dead Orders	0
Next Order Id	33814	Pending Orders	0

☑ Play Sound When Error Occurred

```
11/01 17:04:47   Symbol Limit Reached - Max 500, adding symbol QQQQ
```

Observing Real-Time Performance with System Log

System Log reports the current real-time performance to you. To open, choose from main window, **Program>System Log**.



Data Server Status

Server	DTN IQFeed ActiveX	Today Ticks	3085
Database	Open	Ticks Pending	0
Lookup	Ready	Ticks Filtered	0
Symbols	4	Max Pending	0
Reconnect	1	Critical Events	2 / 0

Order Server Status

Server	TradeSim	Total Orders	0
Status	Ready	Dead Orders	0
Next Order Id	33814	Pending Orders	0

```
10/31 13:30:46   Server initialization completed
10/31 13:30:37   Server initialization started
10/31 13:30:37   Server shutdown sequence completed
10/31 13:30:36   Server shutdown sequence started
10/31 13:30:35   Next Server Restart @ 2005/11/01 05:00:00
```

Data Server Status

These are statistics related to real-time data.

Item	Meaning
Server	Real-time data server NeoTicker® is configured to connect to.
Database	Whether NeoTicker®'s internal database is connected to real-time server. If the connection off, the database is closed.
Lookup	Whether the Real-Time and EOD Lookup tables are ready. These tables are used to store data that are not directly available from data server.
Symbols	Number of symbols that are currently being tracked. These are the symbols that count towards your data server's symbol limit.
Reconnect	Number of times NeoTicker® has re-connected to the real-time server since morning restart.
Today Ticks	Number of ticks processed since morning restart.
Ticks Pending	NeoTicker® buffers ticks before dispatching them to function windows for processing. If your computer is running low in CPU power, you will see **Ticks Pending** increases and it is a sign you should reduce your CPU load.
Ticks Filtered	Number of ticks filtered by the feed level bad tick filter.
Max Pending	Number of ticks pending to be processed. If this number is increasing, it indicates your computer is overloaded.
Critical Events	Number of critical event pending to be processed. Number of ticks pending to be processed. If this number is increasing, it indicates your computer is overloaded.

Order Server Status

These are statistics related to your order server. Order Server handles orders from trading systems and manual order entries (F2/F3). Default server is Trade Simulator. It can also be your real-life broker.

Item	Meaning

Server Order server NeoTicker® is configured to connect to.

Status Whether order server is ready to take orders.

Next Order Id Internal order id to be processed next.

Total Orders Number of orders processed since start up.

Dead Orders Number of orders that cannot be successfully processed.

Pending Orders Orders that have been issued, but have not been filled.

System Events Tab

NeoTicker® system events are listed here.

Order Events Tab

Order processing is logged here.

Error Events

Errors (e.g. out of memory) is logged here.

Bad Ticks Filtered

Details about ticks being filtered.

Importing Text Files

This section tells you how to import intraday data that are stored in text files. You can use the text file data as historical data in NeoTicker®.

Compatible Data File Formats

You should find out the type of text file you are using. NeoTicker® supports the following types of text files: CSV format (comma separated), space separated and tab separated. The file formats supported are flexible and many programs can export it.

The first line of the data file specifies the fields of the file, the rest of the file are fixed time interval data for an instrument.

The fields can be in any order. The first line specifies the order. The field names in the first line can be unquoted, single quoted or double quoted. All fields except date(or datetime) and close are optional.

If you use the real-time version of NeoTicker®, you can import 1-tick CSV file using the Tick CSV format.

The following table lists the supported fields in the text file:

Date	No short form. The format is the same as your Windows date format setting. Commonly used date format is automatically recognized. Required unless DateTime is specified
Time	No short form. Commonly used time format is automatically recognized.
DateTime	Short form DT. The format is Microsoft's date time format in floating point number. Required unless Date is specified
Open	Short form O
High	Short form H
Low	Short form L
Close	Short form C. This field is required Instead of Close, you can use the alternative field names Last or Price.
Volume	Short form V. Instead of the field name Volume or V, you can use the alternative field name Size.
OpenInterest	Short form OI

TradeType No short form. This field is useful only for importing tick files into disk cache.

If value is 0, the tick is treated as trade tick. If value is 1, the tick is treated as bid tick. If value is 2, the tick is treated as ask tick.

Tick No short form. This is the total number of ticks

The following is a few lines of an 15-minute space separated data file:

```
"Date" "Time" "Open" "High" "Low" "Close" "Volume"
1/25/2001 945  1367.500000 1371.500000 1367.000000 1370.500000 141
1/25/2001 1000 1370.000000 1373.500000 1369.000000 1373.000000 124
1/25/2001 1015 1372.900000 1376.500000 1367.000000 1369.000000 141
1/25/2001 1030 1367.000000 1369.000000 1366.000000 1368.500000 116
1/25/2001 1045 1369.500000 1371.000000 1365.000000 1365.000000 88
1/25/2001 1100 1364.500000 1366.000000 1363.500000 1364.500000 123
1/25/2001 1115 1365.000000 1366.500000 1363.600000 1366.500000 130
```

The following is a few lines of a 1-tick comma separated file:

```
Date,Time,Open,High,Low,Close,Volume,Tick,OI
20030925,093034,141.150000,141.150000,141.150000,141.150000,19700,
1,0
20030925,093034,141.150000,141.150000,141.150000,141.150000,1400,1
,0
20030925,093034,141.150000,141.150000,141.150000,141.150000,200,1,
0
20030925,093034,141.150000,141.150000,141.150000,141.150000,200,1,
0
20030925,093035,141.150000,141.150000,141.150000,141.150000,300,1,
0
20030925,093035,141.150000,141.150000,141.150000,141.150000,100,1,
0
20030925,093035,141.150000,141.150000,141.150000,141.150000,100,1,
0
20030925,093035,141.150000,141.150000,141.150000,141.150000,200,1,
0
20030925,093035,141.150000,141.150000,141.150000,141.150000,100,1,
0
20030925,093038,141.150000,141.150000,141.150000,141.150000,100,1,
0
20030925,093038,141.150000,141.150000,141.150000,141.150000,100,1,
0
20030925,093040,141.150000,141.150000,141.150000,141.150000,100,1,
0
20030925,093046,141.150000,141.150000,141.150000,141.150000,100,1,
0
20030925,093050,141.150000,141.150000,141.150000,141.150000,100,1,
0
```

The following is a few lines of a 1-minute comma separated file, including the tick field:

```
Date,Time,Open,High,Low,Close,Volume,Tick,OI
20030923,160000,142.630000,142.630000,142.630000,142.630000,600,1,
0
20030924,093100,142.650000,142.660000,142.650000,142.660000,15600,
1,0
20030924,093200,142.650000,142.700000,142.650000,142.700000,700,1,
0
20030924,093300,142.690000,142.730000,142.690000,142.720000,1600,1
,0
20030924,093400,142.730000,142.730000,142.700000,142.720000,3600,1
,0
20030924,093500,142.730000,142.730000,142.660000,142.660000,3300,1
,0
20030924,093600,142.670000,142.670000,142.650000,142.650000,2400,1
,0
20030924,093700,142.650000,142.650000,142.630000,142.630000,3700,1
,0
20030924,093800,142.630000,142.630000,142.560000,142.560000,3200,1
,0
20030924,093900,142.560000,142.590000,142.550000,142.550000,2500,1
,0
20030924,094000,142.550000,142.550000,142.550000,142.550000,1500,1
,0
20030924,094100,142.510000,142.600000,142.510000,142.590000,21200,
1,0
20030924,094200,142.570000,142.600000,142.570000,142.600000,2200,1
,0
20030924,094300,142.590000,142.600000,142.580000,142.580000,4600,1
,0
```

The following is a few lines of a daily comma separated file:

```
Date,Open,High,Low,Close,Volume,Tick,OI
20030102,61.660000,63.375000,61.525000,63.375000,4358200,1,0
20030103,63.125000,63.370000,62.865000,63.135000,2751600,1,0
20030106,62.850000,63.695000,62.760000,63.595000,3824600,1,0
20030107,63.150000,63.925000,63.150000,63.535000,4320600,1,0
20030108,63.400000,63.400000,62.580000,62.850000,4605600,1,0
20030109,63.000000,63.900000,62.780000,63.845000,3804000,1,0
20030110,63.200000,63.690000,62.925000,63.310000,3993200,1,0
```

Importing Text File to Time Chart

When to Import to Time Chart

You want to import text file to a time chart if you only want to look at the data. You do not expect the data to show up in other parts of NeoTicker® and do not expect the data to integrate with your real-time data feed.

Steps

1. In a time chart, choose **Add Data** from the pop up menu to open the add data dialog .
2. Press the **More Options** button if the dialog is not expanded.
3. Press the **Source** tab in the dialog.
4. Select **Other Source**, and choose the intended text format (e.g. **Text (Space Separated)**).
5. You will need to provide the **Type**, **Bar Size** as if you are adding other type of data series.
6. You will need to specify the directory of the text file. You can press the **Refresh Dir** button to see what files are available in the directory.
7. **Symbol** must be set to the file name of the text file.
8. Press the **Apply** button.

Trouble Shooting

Common problems encountered when importing text files:

- The data format is wrong. Check ***Compatible Data File Formats*** (on page 530) and make sure your text file is compatible.
- The data is not in chronological order.
- You select the wrong text format, e.g. you use **Text (Space Separated)** for a comma separated file.
- The data lies in a range outside of the time chart's trading period, e.g. you imported after market data to a chart that does not show after market bars.
- **Symbol** and file name does not match.

Importing Text File to Disk Cache

When to Import to Disk Cache

Import the text file to disk cache if you want the data to integrate with real-time server. Typically usage is import additional historical data that your data feed does not provide.

Importing a Single Text File

1. Choose **Manager>Cache** in the main window to open cache manager.
2. Press **Import Single File** tab in cache manager.
3. In the **Source** group, choose the file format, enter directory and file name, choose data type.

4 In the **Target** group, enter **Symbol** (this is symbol which data will be imported to) and choose other options.

5 Press **Import Now** button.

Importing Multiple Text Files

1 Choose Manager>Cache in the main window to open cache manager.

2 Press **Import Multiple Files** tab.

3 In the **Source** group, choose the file format, enter directory, choose data type.

4 Choose options in the **Target** group.

5 Press the **Read Dir** button. The files in the directory will be read into the source file/target symbol list.

6 You can choose which file to import, and the target file name for each text file.

7 Press the **Import Now** button.

Settings for importing multiple files can be saved to a file. You can later re-use the same settings by loading them from a file. To save/load settings, press the **Save / Load** button.

Option

The following options under **Option** group are available when importing text files into disk cache:

Option	Usage
Adjust Time Stamp	Offset the time stamp by the specified number of seconds. This option is for importing text files that have a different time stamp convention from NeoTicker®'s. NeoTicker® expects bars to be stamped at end of a time period.
Adjust Volume	Multiple the volume by a multiple. This is for text files that has a volume reported in lots. NeoTicker® expects volumes in number of contracts/stocks.
Construct Minute Bars From Imported Tick Data	When importing tick data, construct minute bars based on the imported tick data.

Compatible Text File Formats

See ***Compatible Data File Formats*** (on page 530).

Daily Data Time Stamp

When you import daily data, it is automatically amended to the trading end time of the day. This is the default and is suitable for most situations.

If your daily data has a time in addition to date, and you want to use this time instead of appending the data to end of trading time:

1 Choose **Program>User Preference** from the main window.

2 Press the **Real-Time Data** tab.

3 Choose **As is** under **Text Files Daily Bars Time Stamp**.

Integrating End-of-Day Data with Real-time Data

You can tell NeoTicker® to use daily data from a EOD data vendor and integrates the data seamless with your real-time data feed.

This is useful if your real-time data feed does not provide daily data that is as high quality or as long as a EOD data vendor. To set up:

1 Choose **Program>Server Setup** from main window.

2 Press the **Data** tab.

3 Check one of the option under **Daily Data Integration**.

If you choose **Use Datafeed data only**, only EOD data from the real-time data feed are used.

If you choose **Auto Merge with Internet Data**, the daily data will obtain from a website NeoTicker® chooses for you.

If you choose **Auto Merge with Quotes Plus**, the daily data will obtain from Quotes Plus (subscription to Quotes Plus required). Quotes Plus is a fast and reliable source for daily data, with commodities data.

If you choose **Auto Merge with TC2005**, the daily data will obtain from TC2005 (subscription to TC2005 required). TC2005 is a fast and reliable source for daily data.

Internet Setup for EOD Data

This section has been moved to ***Internet EOD Data Connection Setting*** (on page 609).

End Of Day Data Vendors and Formats Supported

Internet EOD Data

Cache

Internet EOD data is cached on your hard drive. You do not have to concern about the cache as EOD data occupies very little space on your hard drive.

Stock Exchange Extensions

NeoTicker® uses a Yahoo! style extension for stock symbols:

Market	Extension
US stocks	No extension
Toronto	.TO
Montreal	.M
Vancouver	.V
Alberta	.AL
Australia	.AX
New Zealand	.NZ
Hong Kong	.HK (Note: Your must fill in all digits in the ticker, e.g. 0005.HK)

Commodities

Commodities data uses the following format: `@XXYYM`, where XX is the symbol, YY is the 2-digit year code, M is the month code. For example: `@ES04H` is the symbol for March 2004 S&P 500 e-mini contract. `@NQ04M` is the symbol for June 2004 Nasdaq e-mini contract.

Continuous contract is supported for ES, NQ and YM. You can simply type the symbol `@ES`, `@NQ` and `@YM` without the year and month code to chart these symbols. The data is stitched automatically using multiple contracts. The roll over date is the Thursday one week ago from the third Friday of the expiration month.

When accessing a continuous contract, expired contract is not retrieved from the Internet as such data is no longer available. So data for expired contract must already present in the cache. NeoTicker® already comes with some data and you will accumulate more data as you continue using NeoTicker®.

Lycos Data

Internet EOD Data uses an automatic scheme to retrieve data. By default, data comes from Yahoo for stocks and Lycos for commodities.

You can force Internet EOD Data to request stock data from Lycos by adding an exchange prefix. The following tables list several examples.

Symbol	Source
IBM	IBM data from Yahoo
MSFT	MSFT data from Yahoo
NYSE:IBM	IBM data from Lycos
NASDAQ:MSFT	MSFT data from Lycos

Below is for NeoTicker® EOD only.

Quote Window

Quote window supports delay quote for commodities data. The symbol format is the same as in charts.

Quote window supports some European stock exchanges:

Market	Extension
Berlin	.BE
Düsseldorf	.D
Frankfurt	.F
Hamburg	.H
Hannover	.HA
Madrid	.MC

Milano	.MI
Munich	.MU
Paris	.PA
Stuttgart	.SG
Xetra	.DE
London	.L

Quotes Plus

If you are a subscriber to the data vendor Quotes Plus, NeoTicker® can access data directly from Quotes Plus. You do not have to export the data to files manually from Quotes Plus.

TC2005

If you are a subscriber to the data vendor TC2005, NeoTicker® can access data directly from TC2005. You do not have to export the data to files manually from TC2005.

Metastock Files

NeoTicker® supports Metastock files. When you add a data series, you need to provide the directory where the Metastock files are stored.

CSI/CSIM Files

NeoTicker® supports CSI/CSIM files. When you add a data series, you need to provide the directory where the CSI/CSIM files are stored.

ASCII Files (EOD only)

The **ASCII Files** format is for end of day data only, for intraday data stored in files, use one of **Text** formats.

NeoTicker® provides a Text Import Wizard to let you configure NeoTicker® to read a wide variety of EOD ASCII files (text files).

Use the following steps to import EOD ASCII files:

1. When you add a data series, choose **ASCII** as the data source.
2. Specify the directory where the text files are located.
3. Press the **S** button to open the text file setup window.
4. Choose **Start Text Import Wizard**.
5. Follow the steps in the wizard.

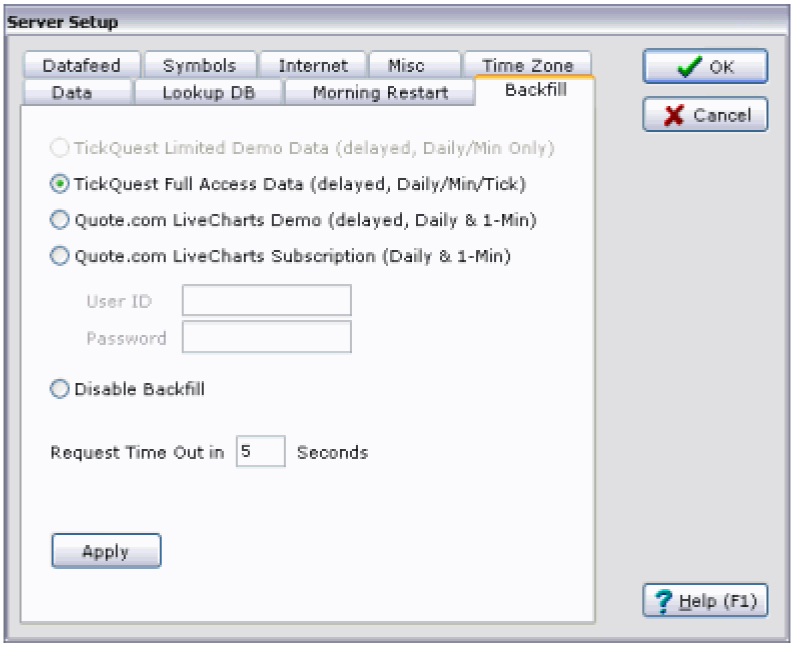
Once a directory is configured, NeoTicker® recognizes all text files in the directory. NeoTicker® only supports PC text files. If your text files are originated from Apple MacIntosh or Unix, they may not be ready to be used in NeoTicker® until you convert them to PC text files by using a third party utility.

Setting Up Backfill

NeoTicker® can automatically backfill historical data when using:

- Supported brokerage feeds such as MB Trading and Interactive Brokers.
- Offline mode

You can tell by choosing **Program>Server Setup** from main window to open Server Setup. Services that support backfill will have a **Backfill** tab.



Server Setup

○ TickQuest Limited Demo Data (delayed, Daily/Min Only)

● TickQuest Full Access Data (delayed, Daily/Min/Tick)

○ Quote.com LiveCharts Demo (delayed, Daily & 1-Min)

○ Quote.com LiveCharts Subscription (Daily & 1-Min)

User ID

Password

○ Disable Backfill

Request Time Out in 5 Seconds

Backfill is a set once and forget it option. It is applied automatically when NeoTicker® requests for historical data and is transparent to you.

If you use a real-time data feed (e.g. MB Trading) with delay back fill, there will be a data hole when you open a chart because of the time discrepancy between the latest bar in back fill and the real-time data. This hole can be repaired in charts by pressing the F11 key after the delay period is over.

Available options are:

TickQuest Limited Demo Data

This option is available when you use demo version of NeoTicker®.

Choosing this option will let you backfill using a limited subset of **TickQuest Full Access Data**. You have access to up to date Forex data and delayed minute data for other symbols.

This option is for demoing NeoTicker® and the service we provide. The data itself is excellent for research and some real-time use.

Backfill Server is hosted by TickQuest. Because we are serving an exclusive class of clients, we can provide much higher quality and quantity of data than standard backfill service offered by brokerages. For up to date information on Backfill Server, visit ***Backfill Server Webpage*** (http://www.tickquest.com/backfillserver.html).

TickQuest Full Access Data

This option is available when you use full or lease version of NeoTicker®.

Choosing this option will let you access all data on our Backfill Server. You have access to up to date Forex data and delayed minute data for other symbols. Historical tick data is also available.

The data itself is excellent for research and simulation use, and is suitable for some real-time use.

Backfill Server is hosted by TickQuest. Because we are serving an exclusive class of clients, we can provide much higher quality and quantity of data than standard backfill service offered by brokerages. For up to date information on Backfill Server, visit ***Backfill Server Webpage*** (http://www.tickquest.com/backfillserver.html).

Quote.com LiveCharts Demo

Free delay data from Quote.com for most stocks and futures. Up to 500 bars are returned. Service can be unreliable during market hours.

LiveCharts demo is suitable only for demonstrating the LiveCharts service. For production system, consider using LiveCharts subscription.

Quote.com LiveCharts Subscription

Choose this option if you have a LiveCharts account.

LiveCharts provide non-delayed data for exchanges you have subscribed to. Up to 500 bars are returned.

LiveCharts subscription is very fast and reliable. It is suitable for production use.

Disable Backfill

Turn off backfill.

Request Timeout

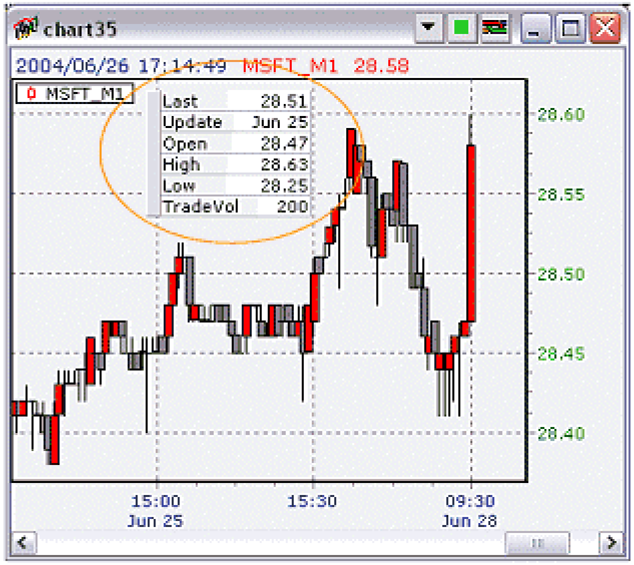
Timeout before a backfill request is consider a failure. If you have a slow Internet connection or backfill data feed, increase the timeout value.

How Does Backfill Work

Backfill is done automatically when historical data is required, for example, in charts. You don't have to do anything. All symbology is translated automatically, and you work using your real-time data symbology. Only if you want to change backfill service you need to visit this tab again.

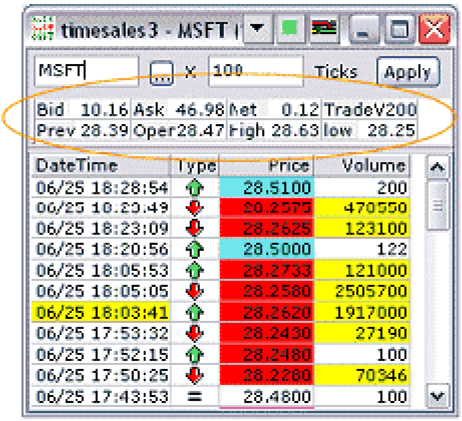
Dynamic Grid Operation Guide

Dynamic grid is the embedded quote service used in time and sales window (as the summary panel), time charts (as the floating grid) and Dynamic Table. In addition to quotes, it provides formula calculation and charting. Dynamic grid can monitor data can be from any security, and is not limited by the one security you are currently monitoring.



2004/06/26 17:14:49 MSFT_M1 28.58

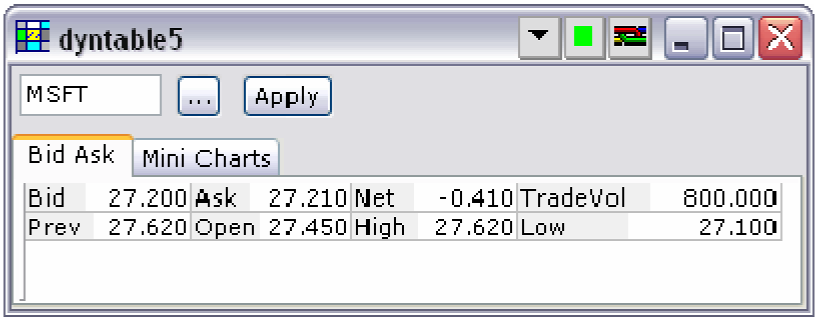
Last	28.51
Update	Jun 25
Open	28.47
High	28.63
Low	28.25
TradeVol	200



MSFT x 100

Bid 10.16	Ask 46.98	Net 0.12	TradeV200
Prev 28.39	Oper28.47	High 28.63	low 28.25

DateTime	Type	Price	Volume
06/25 18:28:54	↑	28.5100	200
06/25 18:23:49	↓	28.2575	470550
06/25 18:23:09	↓	28.2625	123100
06/25 18:20:56	↑	28.5000	122
06/25 18:05:53	↑	28.2733	121000
06/25 18:05:05	↓	28.2580	2505700
06/25 18:03:41	↑	28.2620	1917000
06/25 17:53:32	↓	28.2430	27190
06/25 17:52:15	↑	28.2480	100
06/25 17:50:25	↓	28.2280	70346
06/25 17:43:53	=	28.4800	100



MSFT

Bid	27.200	Ask	27.210	Net	-0.410	TradeVol	800.000
Prev	27.620	Open	27.450	High	27.620	Low	27.100

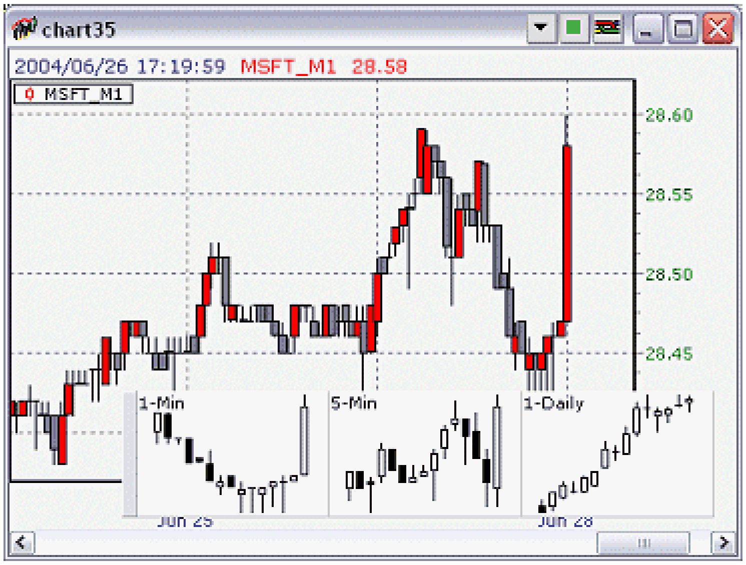
Dynamic grid works on two levels. The basic level is an overall grid that has provides the general defaults. The next level is the cells that display information. The cells can have different setups to override the defaults provided by the grid.

Quick Grid Setup

The whole Dynamic Grid can be setup quickly with one of the built in templates.

To setup, right click on Dynamic Grid to open pop up menu, and choose one of the templates under **Grid Setup**.

For example, right click on a time chart's dynamic grid and choose **Grid Setup>3 Charts**.



Grid Level Setup

Setting up a grid will let you change the default properties.

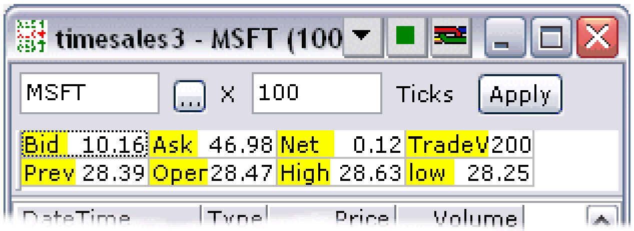
Example

In this example, we will change the label's background color from gray to yellow.

1 Right click on the dynamic grid to open the pop up menu and choose **Grid Setup>Open Grid Setup Window**. This will open a setup window.

2 Click on the **Visual** tab, and change the **Label** color to yellow. Press the **OK** button.

All the labels in the grid will now have a yellow background. This is how the dynamic grid will look in a time and sales window.


timesales3 - MSFT (100

MSFT ... x 100 Ticks Apply

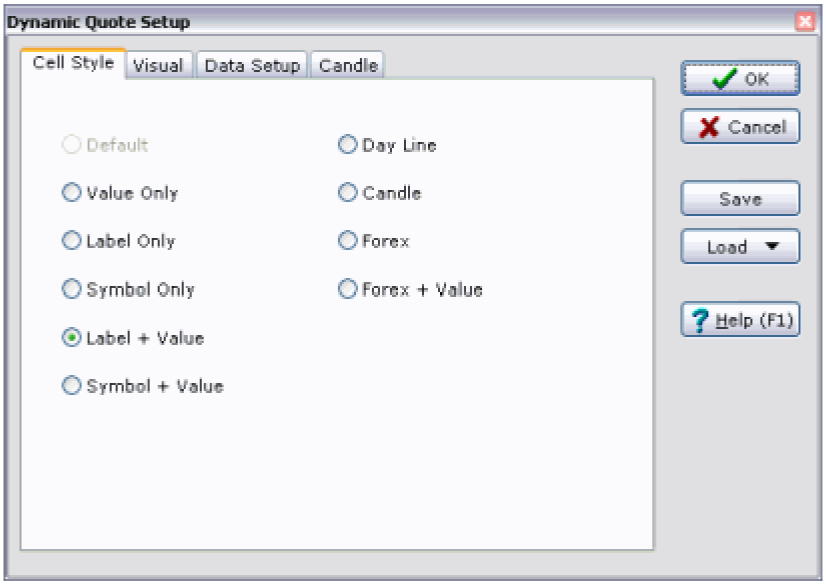
Bid	10.16	Ask	46.98	Net	0.12	TradeV	200
Prev	28.39	Oper	28.47	High	28.63	low	28.25

DateTime | Type | Price | Volume

Similarly, you can change the font, marker position for the grid.

Grid level setup provides default settings for cells. When a cell is displayed, the default settings are used if the cell does not override the settings.

Cell Style Tab


Dynamic Quote Setup

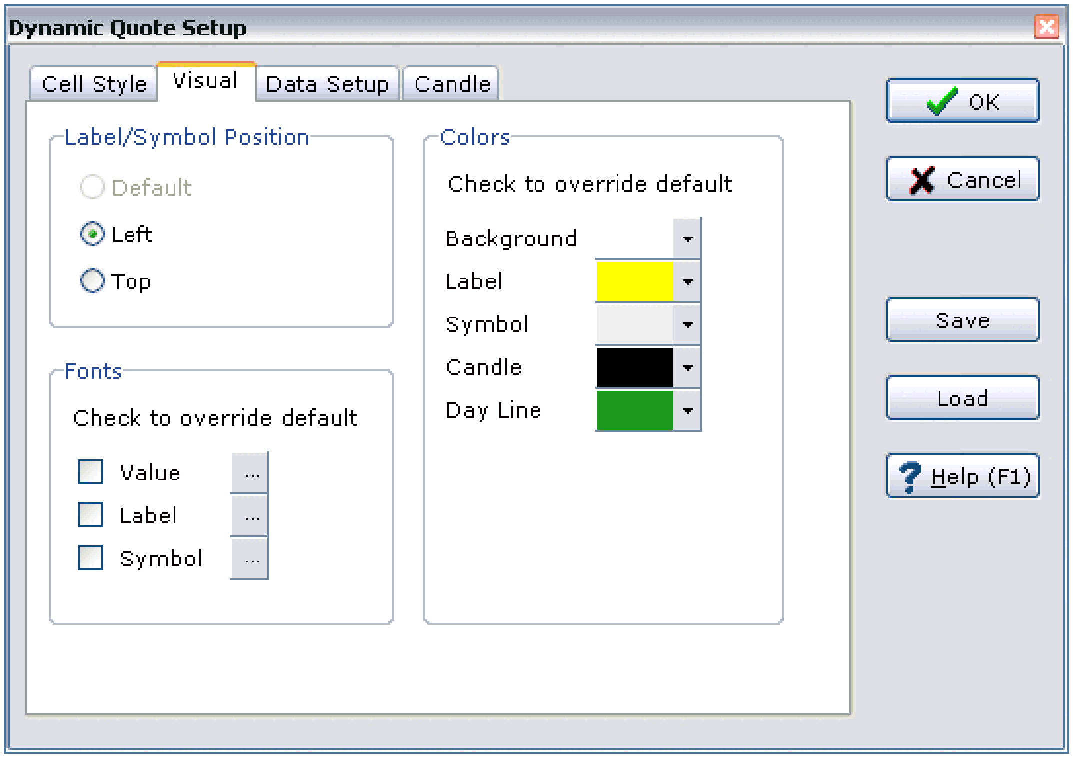
Cell Style | Visual | Data Setup | Candle

○ Default
○ Value Only
○ Label Only
○ Symbol Only
● Label + Value
○ Symbol + Value
○ Day Line
○ Candle
○ Forex
○ Forex + Value

This tab specifies the how the data is displayed in cells.

Setting	Meaning
Value Only	Displays the value of the cell only. The value is typically price, volume or the result of indicator calculation (e.g. 13.50, 55.10)
Label Only	Displays the label of the cell only (e.g. bid, ask)
Symbol Only	Displays the symbol of the cell only (e.g. MSFT)
Label + Value	Displays label first, then value (e.g. bid 13.50)
Symbol + Value	Displays the symbol first, the value (e.g. MSFT 49.50)
Day Line	Displays a mini chart of the day in line form
Candle	Displays a mini candlestick chart
Forex	Format specific for Forex trading (bid/ask/low/high)
Forex + Value	Format specific for Forex trading (bid/ask/low/high), plus a custom formula calculation value

Visual Tab



Label/Symbol Position

○ Default
● Left
○ Top

Fonts

Check to override default

□ Value ...
□ Label ...
□ Symbol ...

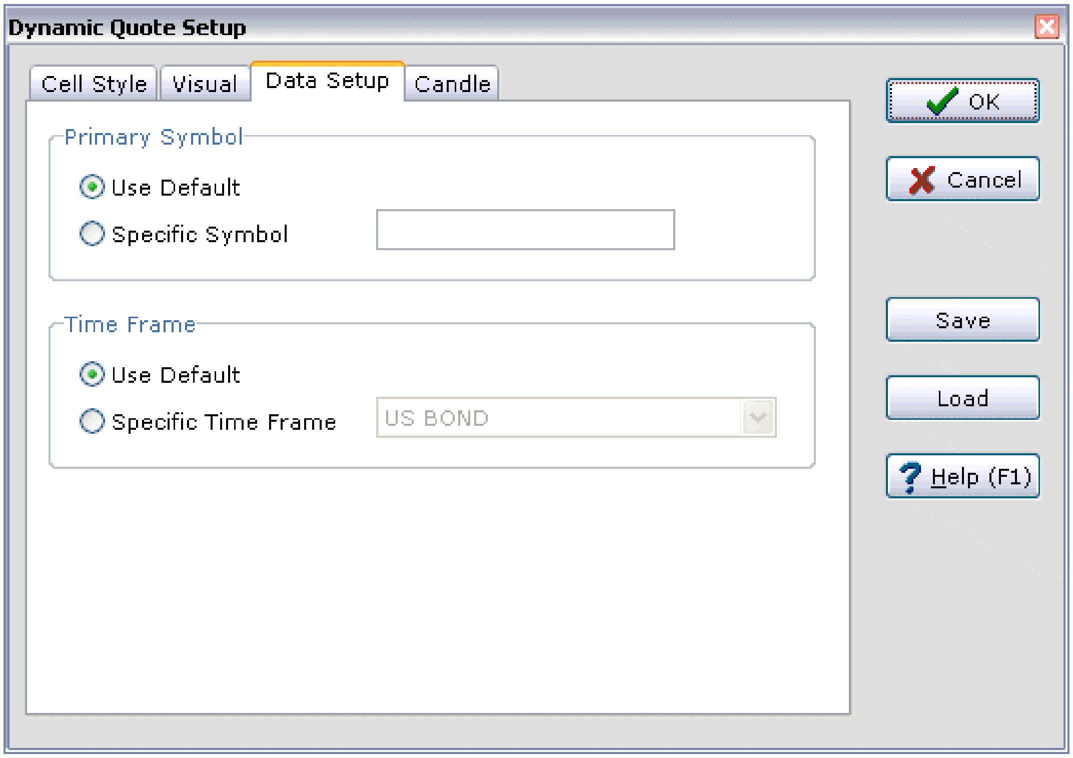
Colors

Check to override default

Background	
Label	🟨
Symbol	⬜
Candle	⬛
Day Line	🟩

You can set things such as font, color, label/symbol position under this tab.

Data Setup Tab



Dynamic Quote Setup

Primary Symbol

● Use Default

○ Specific Symbol

Time Frame

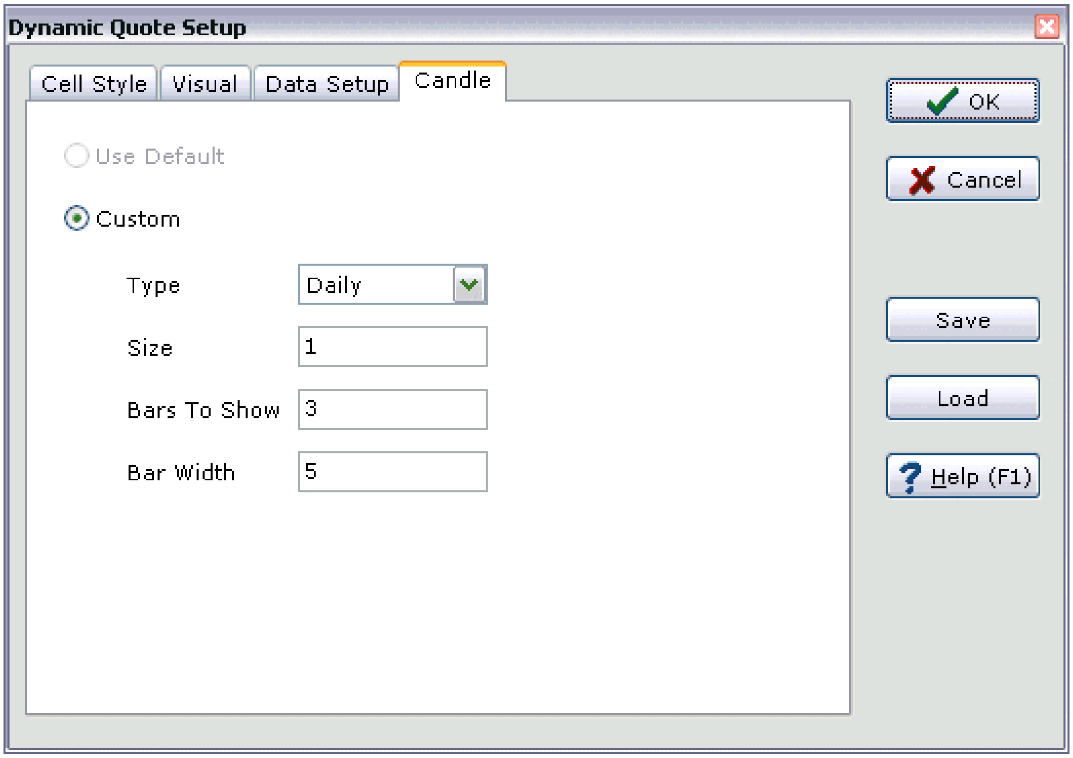
● Use Default

○ Specific Time Frame US BOND

Primary Symbol is the symbol of the grid. If **Use Default** is chosen, the symbol is inherited from the container of the dynamic grid (e.g. symbol of the time and sales window). You can specify another symbol as the primary symbol.

Time Frame is the trading time and holiday information of the dynamic grid for indicator calculation. If **Use Default** is chosen, time frame is inherited from the container. You can specify any time frame that is defined by time frame manager.

Candle Tab



Dynamic Quote Setup

○ Use Default

● Custom

Type: Daily

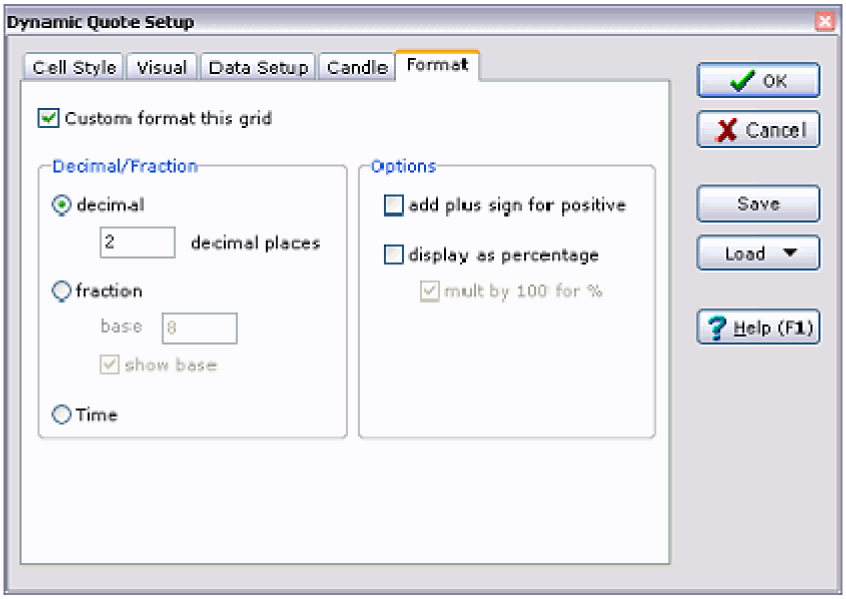
Size: 1

Bars To Show: 3

Bar Width: 5

If a mini candlestick chart is displayed, this tab provides the settings for the candle.

Format Tab



☑ Custom format this grid

● decimal

2 decimal places

○ fraction

base 8

☑ show base

○ Time

□ add plus sign for positive

□ display as percentage

☑ mult by 100 for %

This tab is for formatting the cell values.

Setting	Meaning
Custom format this grid	Toggle custom formatting.
decimal	Values are displayed with decimal places
fraction	Values are displayed with fractions
time	Values are formatted as time
add plus sign for positive	Add a plus sign in front of positive values
display as percentage	Values are formatted as percentage. If **mult by 100 for %** is enabled, the values are multiplied by 100 before formatting.

Grid Visual

The grid visual actions let you quickly adjust the physical size of grid/cells. The grid visual actions are under pop up menu, **Grid Visual**.

Fit to Grid - Resize the grid such that all cells will fit into the grid.

Default Font - Set all cells to use the default font.

Set All Row Height To Current - Set all rows to the same height as current row.

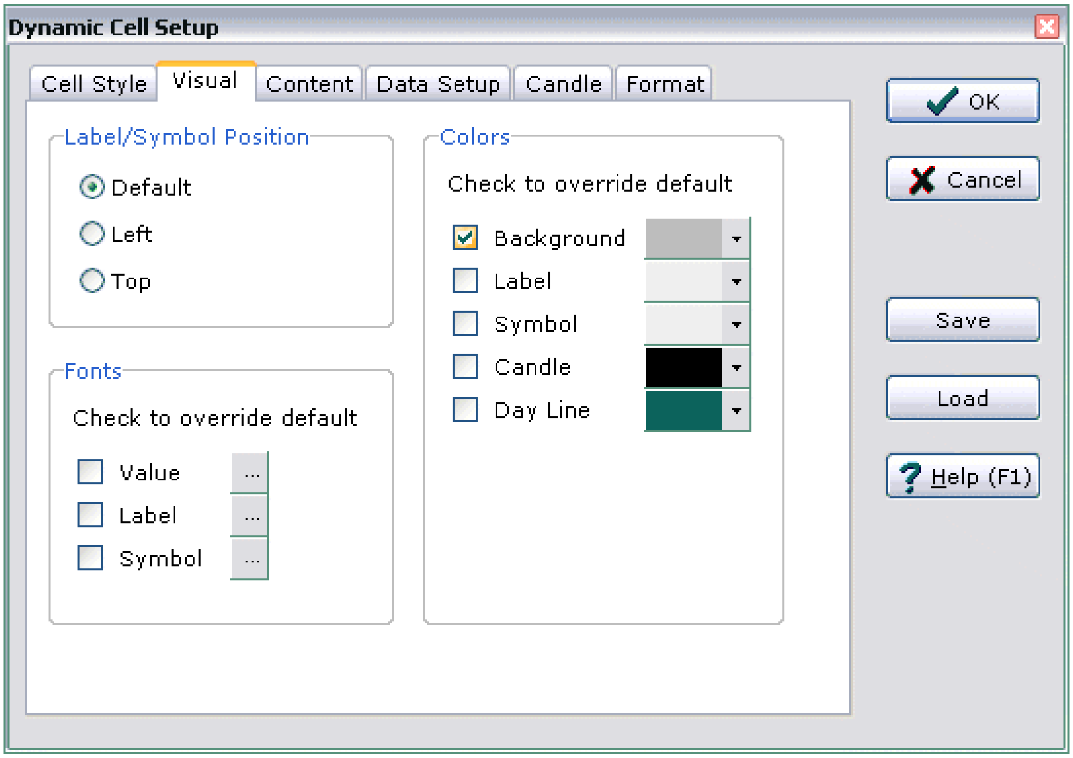
Set All Column Width To Current - Set all columns to the same width as current column.

Cell Level Setup

Setting up a Cell

You can set up a cell to override the settings provide by the grid.

Right click on a cell (for example, bid) to open the pop up menu, and choose **Cell Setup>Open Cell Setup Window**. This will open a setup window. Click on the **Visual** tab, and change the **Background** color to gray and check the check box besides **Background**. Press the **OK** button.


Dynamic Cell Setup

Label/Symbol Position

● Default
○ Left
○ Top

Fonts

Check to override default

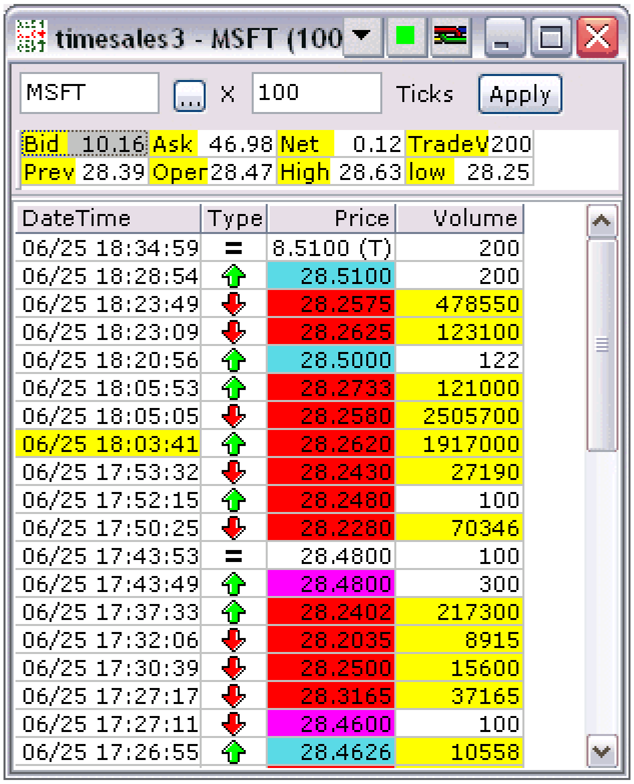
□ Value
□ Label
□ Symbol

Colors

Check to override default

☑ Background
□ Label
□ Symbol
□ Candle
□ Day Line

The background color for bid will become gray.



timesales3 - MSFT (100

MSFT ... x 100 Ticks Apply

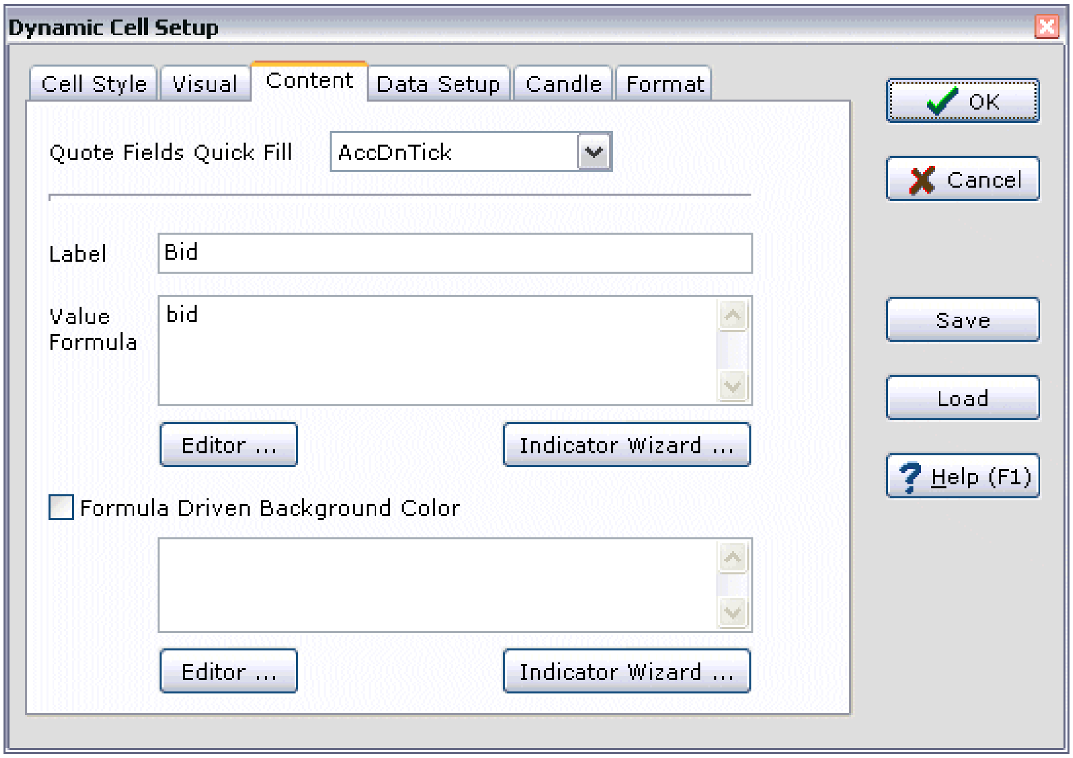
Bid 10.16 Ask 46.98 Net 0.12 TradeV 200
Prev 28.39 Oper 28.47 High 28.63 low 28.25

DateTime	Type	Price	Volume
06/25 18:34:59	=	8.5100 (T)	200
06/25 18:28:54	↑	28.5100	200
06/25 18:23:49	↓	28.2575	478550
06/25 18:23:09	↓	28.2625	123100
06/25 18:20:56	↑	28.5000	122
06/25 18:05:53	↑	28.2733	121000
06/25 18:05:05	↓	28.2580	2505700
06/25 18:03:41	↑	28.2620	1917000
06/25 17:53:32	↓	28.2430	27190
06/25 17:52:15	↑	28.2480	100
06/25 17:50:25	↓	28.2280	70346
06/25 17:43:53	=	28.4800	100
06/25 17:43:49	↑	28.4800	300
06/25 17:37:33	↑	28.2402	217300
06/25 17:32:06	↓	28.2035	8915
06/25 17:30:39	↓	28.2500	15600
06/25 17:27:17	↓	28.3165	37165
06/25 17:27:11	↓	28.4600	100
06/25 17:26:55	↑	28.4626	10558

Setup

Cell level setup is almost identical to grid level setup. You can refer to ***Grid Level Setup*** (on page 547) for more information.

Cell level setup provides an additional **Content** tab for quote and formula settings.



Quote Fields Quick Fill: AccDnTick

Label: Bid

Value Formula: bid

□ Formula Driven Background Color

Setting	Function
Quote Field Quick Fill	Choose any quote window field to quickly fill **Label** and **Formula**
Label	The label if the grid/cell setup allows label display
Formula	The cell value. Cell value is a formula. The formula syntax is identical to that of quote window. You can also use the Indicator Wizard to help you specify indicators.
Formula Driven Background Color	Background color, driven by formula.

Chart in a Cell

See ***Dynamic Grid Operation Guide, Setting up a Cell to Display a Chart*** (see "Setting up a Cell to Display a Chart" on page 560).

Formula in a Cell

See ***Dynamic Grid Operation Guide, Setting up a Cell to Display Formula*** (see "Setting up a Cell to Display Formula" on page 562).

How Formula Driven Background Color Works

If **Formula Driven Background Color** option is checked, then the cell's background color will be driven by formula. The formula should return an integer value representing the color. For example, the following lines are valid formula driven background colors:

```
clRed
```

and

```
if (mov(0,M1,"Simple",10) > mov(0,M1,"Simple",20), clRed,
clGreen)
```

Note that for color, it uses standard Windows color naming. The following pre-defined colors are available:

```
clAqua
clBlack
clBlue
clDkGray
clFuchsia
clGray
clGreen
clLime
clLtGray
clMaroon
clNavy
clOlive
clPurple
```

```
clRed
clSilver
clTeal
clWhite
clYellow
```

You can also use integer value for color. The integer value is decimal representation of the following hexadecimal number BBGGAA, where BB is 2-digit hex representing blue, GG is 2-digit hex representing green, RR is 2-digit hex representing red. For example, 000066 (hex) = 102 (decimal) represents a dark red color.

Saving and Loading Setups

Both grid and cell setups can be saved for later reuse.

To save grid setup,

1. Right click on Dynamic Grid to open pop up menu.
2. Choose **Grid Setup>Open Grid Setup Window**.
3. Press **Save** button.
4. Give the setup a name.

To load grid setup, use Grid Setup Window, **Load** button.

To save cell setup,

1. Right click on Dynamic Grid to open pop up menu.
2. Choose **Cell Setup>Open Cell Setup Window**.
3. Press **Save** button.
4. Give the setup a name.

To load cell setup, use Cell Setup Window, **Load** button, or choose from pop up menu, **Cell Setup>Load Recently Used**.

Rearranging Cells

Drag and Drop

You can drag and drop cells to rearrange them.

Auto Arrange

You can auto arrange cell by:

1. Right click on Dynamic Grid to open pop up menu.
2. Choose one of the options under **Arrange**.

Vertical will arrange all cells in a single vertical column.

Horizontal will arrange all cells in a single horizontal row.

Custom will arrange cells in a customized row x column grid.

Dynamic Grid is not resized after auto arrange. You may need to resize Dynamic Grid to see all cells after an arrange.

Adding More Cells

You can add more cells for displaying information. Right click on a cell to open the pop up menu, and choose **Insert>Row**. This will add a row of cells.

If you choose **Add to Last>Row**, the row of cells will be added to the end of the grid.

Dynamic Grid is not resized after adding cells. You may need to resize Dynamic Grid to see all cells after adding.

Deleting Cells

You can delete cell rows or columns by right click on Dynamic Grid to open pop up menu and choose **Delete>Row** or **Delete>Column**.

You can delete a single cell by choosing **Delete>Cell**. After you deleting a single cell, cells to the right of the deleted cell will be moved from right to left and from bottom to top to cover the deleted cell. Because the cells in the summary panel have to be in rectangular arrangement, a blank cell will be created to preserve the rectangular arrangement of cells.

Copying a Cell

You can copy the setting of a cell to other cells from the pop up menu item **Copy**.

To/From Temp File

You can copy a cell to a temp file first. Then on the destination cell, copy the setting from the temp file.

You can also use this method to copy cells between different Dynamic Grids.

Destination Cells

For a direct cell copy without using a temp file, you can copy a cell to:

- All cells
- Cells in current row
- Cells in current column
- Cells in the current row, right side of the current cell
- Cells in the current column, below current cell

There is a menu entry corresponds to each of these destinations.

With Symbol Settings

If you copy with symbol settings, all the settings of a cell is copied.

Without Symbol Settings

If you copy without symbol settings, the symbol settings (the settings under **Cell Setup**>**Data Setup** tab>**Primary Symbol**) are not copied. Copying this way is useful if you are setting up Dynamic Grid to display data from multiple symbols, in which case it is handy to copy things like formula over to other cells without affecting their symbols.

Symbol Setting Only

You can copy only the symbol settings. This is useful for setting up multiple cells to analyze a single symbol. For example, you have 3 cells displaying different charts for a symbol. You can quickly change the symbol by copying the symbol setting from one cell to other cells.

Clearing Cells

You can use one of the action under pop up menu, **Clear** to clear cells. A cleared cell will have all its settings reset to the default state, i.e. what a new cell looks like.

Distribute Symbols

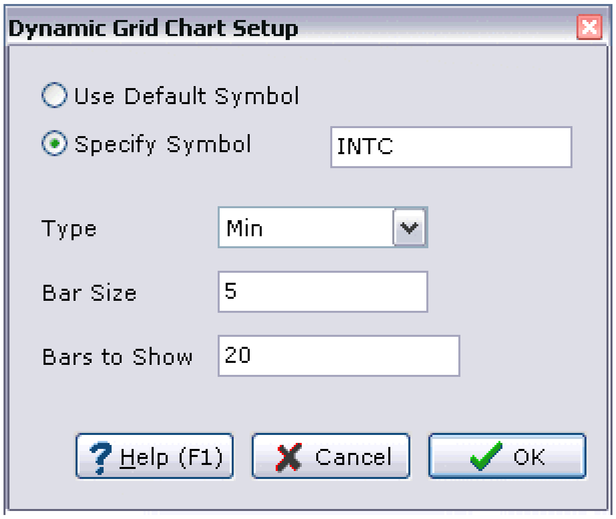
You can distribute symbols in a symbol list to the cells in Dynamic Grid. Symbol list is a simple text file, with each line containing a single symbol. You can use Windows Notepad or Symbol List Manager to create a symbol list.

To distribute symbols, choose a target from **Distribute Symbols** from pop up menu, and select a symbol list.

When distributed, each symbol in the symbol list will send to a cell in the targe cells.

Setting up a Cell to Display a Chart

To set up a chart in a cell, right click on a cell to open the pop up menu, and choose one of the templates under **Cell Setup>Quick Chart**. You can also choose **Cell Setup>Quick Chart>Other Chart** to specify parameters.



Dynamic Grid Chart Setup

○ Use Default Symbol

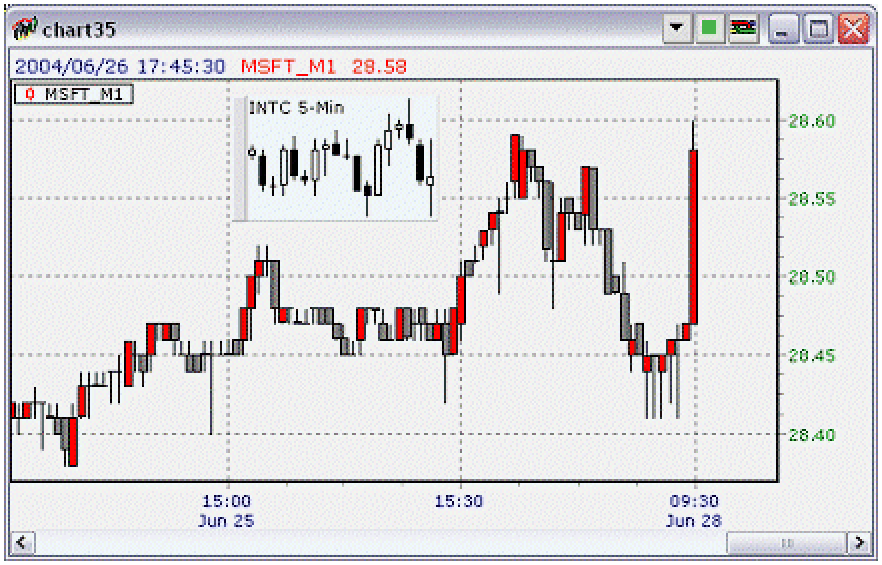
● Specify Symbol INTC

Type Min

Bar Size 5

Bars to Show 20

For example, this lets you set up a 5-minute INTC chart in a dynamic cell within a time chart.



Setting up a Cell to Display Formula

You can display result of formula calculations in cells. This section illustrates how to display a formula that calculates moving average in a cell.

If you have not created some empty cells, create some by choosing **Insert>Row** or **Insert>Column**.

Right click on an empty cell to open the pop up menu and choose **Cell Setup>Open Cell Setup Window**.

In the Cell Setup Window, press the **Content** tab. In the **Label** area, type **MA**.

In the formula area, type `mov(0, M1, "Simple", 10)`. Instead of typing, you can also use the Indicator Wizard to enter the formula.

Press the **OK** button. This will create a cell that calculates moving average.

Note that the formula in Dynamic Grid supports quote window formula. You can copy a quote window formula and use it in Dynamic Grid directly.

Dynamic Grid supports additional function for accessing cell values. For more information see ***Context-Driven Functions for Dynamic Grid*** (on page 308).

Debugging

When you encounter a formula error, you can choose **Error Report** from pop up menu to view a detail description of the error.

Formula Driven Colors

You can change the background color of a cell by formula. This is a useful technique to quickly raise your concern if some special condition has happened.

Right click on a cell to open the pop up menu and choose **Cell Setup>Open Cell Setup Window**.

In the cell setup window, press the *Content* tab. Check the **Formula Driven Background Color** option and enter the following formula in the area below:

```
if (mov(0,M1,"Simple",10) > mov(0,M1,"Simple",20), clRed,
clGreen)
```

This will make the background color for the cell red if the 10- period simple moving average is above the 20-period simple moving average. The background color of the cell will be green otherwise.

Instead of typing, you can use the Indicator Wizard to enter the indicator part of the formula.

Note that for color, it uses standard Windows color naming. The following colors are available:

```
clAqua
clBlack
clBlue
clDkGray
clFuchsia
clGray
clGreen
clLime
clLtGray
clMaroon
clNavy
clOlive
clPurple
clRed
clSilver
clTeal
clWhite
clYellow
```

You can also use integer for color. The integer color is decimal representation of the following hexadecimal number BBGGRR, where BB is 2-digit hex representing blue, GG is 2-digit hex representing green, RR is 2-digit hex representing red. For example, 000066 (hex) = 102 (decimal). This is a dark red color.

Double Click Action

You can configure Dynamic Grid to respond differently when you double click on a cell.

1 Right click on Dynamic Grid.

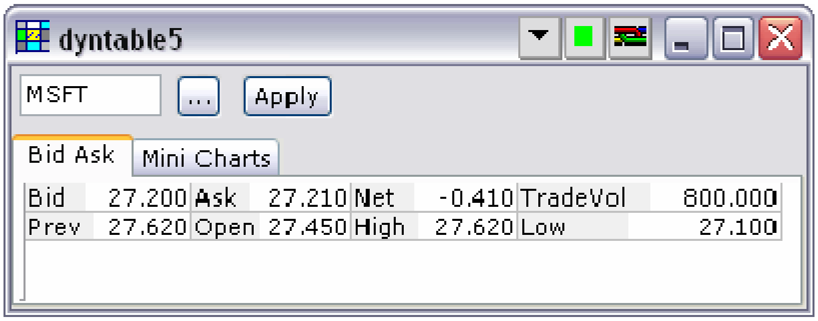
2 Choose one of the option under **Grid Double Click**.

Show Reference Symbol

It is often difficult to tell which cell is referencing which symbol. A quick way to tell is to toggle **Show Reference Symbol** in pop up menu. When this option is on, instead of the data, the symbol a cell is referencing to is displayed.

Dynamic Table Operation Guide

Dynamic table is a standalone window that can hold multiple Dynamic Grids.

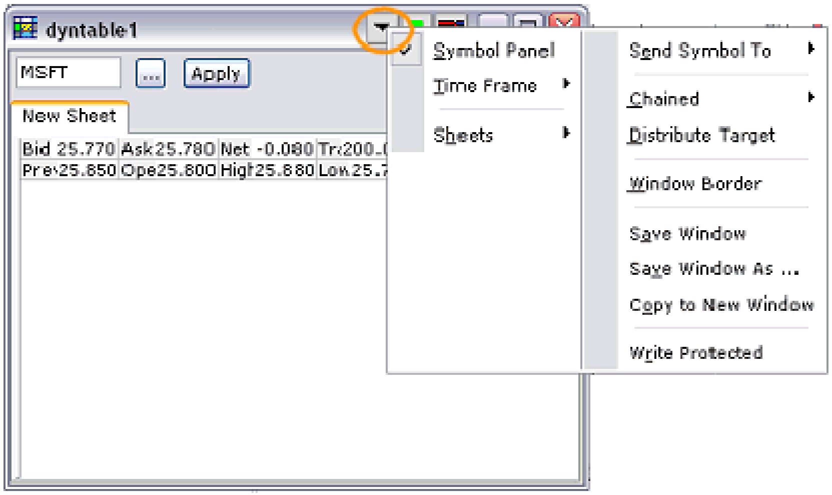


Bid	27.200	Ask	27.210	Net	-0.410	TradeVol	800.000
Prev	27.620	Open	27.450	High	27.620	Low	27.100

Dynamic Table Set up

A Dynamic Table consists of multiple sheets. Each sheet holds a Dynamic Grid.

In general, when you right click on a cell, you are opening the pop up menu of the cell. But if you want to open the pop up menu of Dynamic Table, you need to right click on an empty area of the table, or use the menu button in caption.



Bid 25.770	Ask25.780	Net -0.080	Tr:200.(
Pre\25.850	Ope25.800	Higł25.880	Low25.7

The Dynamic Grids share a common default symbol.

Default Symbol

To change default symbol, simply change the symbol in Dynamic Table and press Apply button.

Symbol Panel

Symbol Panel is top area in Dynamic Table. You can hide/show the Symbol Panel:

1 Right click on any area except inside the grid to open pop up menu.

2 Toggle **Symbol Panel** to hide/show Symbol Panel.

Sheets

To create/delete/edit/copy sheets:

1 Right click on any area except inside the grid to open pop up menu.

2 Choose one of the option under **Sheet.**

To move/reorder Sheets:

1 Drag the tab of the sheet into the target position.

You can drag a sheet across to another Dynamic Window. In this case, a copy of the dragged sheet will be created and insert into the target window.

Time Frame

Since the grid can contain indicator calculation, time frame information (trading time, holidays, etc) can affect indicator value. To adjust time frame:

1 Right click on any area except inside the grid to open pop up menu.

2 Choose an option under the menu item **Time Frame**.

Grid Operations

Each grid under a sheet is Dynamic Grid. For Dynamic Grid operations, refer to ***Dynamic Grid Operation Guide*** (on page 545).

General Operations

Adding and Removing Windows from a Group

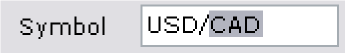
Your session can contain multiple groups to let you quickly switch between different groups of windows.

You can add and remove window to/from groups by either:

- Choose **Group>Group and Window List** and use the group and window list window to manage windows, or
- Select the window you want to add/remove, and choose **Window>Add to Active Group** or **Window>Remove from Active Group**.

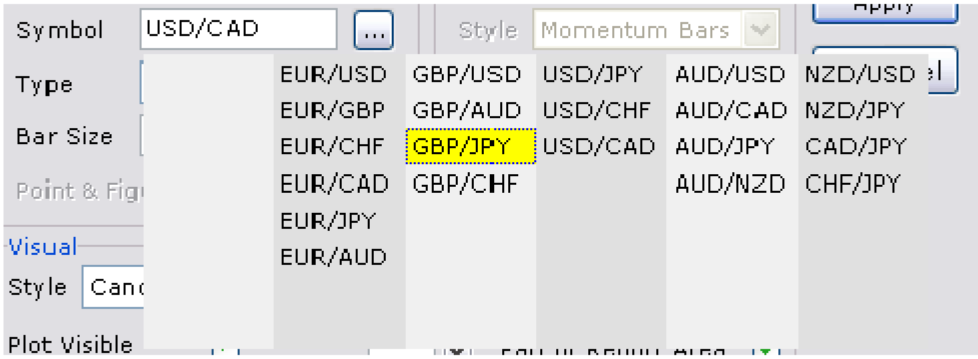
Auto Symbol Completion

In fields where you type in symbols, NeoTicker® will attempt to auto complete the symbol for you. If more than one choice of a symbol is available, the ambiguous part of the symbol is highlighted. You can either correct the ambiguous part by typing, or use the up/down arrow keys on the keyboard to make a selection.



Symbol USD/CAD

If you right click on the symbol entry field, or press the menu button on the keyboard, a menu will be opened to let you quickly select from a list of symbols.



Symbol USD/CAD

Style Momentum Bars

If you are uncomfortable with auto symbol completion, you can disable this feature in User Preference:

1. In Main Window, choose **Program>User Preference** to open User Preference window.
2. In User Preference window, press **General** tab.
3. Toggle **Auto Complete Symbol Entry**.

4 Press **OK** button.

Customization

The list of symbols used for this feature is stored in the `SymbolList/QuickEntry` file in your NeoTicker® installation. This is a text file you can customize.

Bad Ticks Filtering

Two levels of bad tick filter exists: a basic tick filter that works on feed level and tick filters that work on function window level.

Basic Tick Filter for Feed Level Filtering

The basic bad tick filter discard bad ticks when the data is coming from the data feed. Bad ticks do not enter the disk cache, and subsequently do not enter any function window. This level of bad tick filter only filters out ticks that are way out of line of normal price.

A tick is filter out by the basic tick filter if:

- The instrument has positive values only (some index has negative values), and
- The tick is way above or below of the last price

Basic tick filter is enabled by default. To enable/disable it manually:

1 Choose **Program>User Preference** from the main window

2 Click the **Real-Time Data** tab

3 To enable tick filtering, check **Reject Possible Bad Ticks**; to disable, check off this option

4 (optional) If your data feed makes the distinction between normal symbol and index type symbol, you can instruct the tick filter to ignore index type symbols.

If the symbol is defined in the Symbol Info Manager, it has a field to instruct whether the basic tick filter should be applied on the symbol. For more information, see Symbol Info Manager.

To observe the actions of bad tick filter, choose from main window, **Program>System Log**, and press the **Bad Ticks Filtered** tab.

Bad Tick Filter in Chart

Tick filtering in chart provides sophisticated filtering capability that is formula driven, i.e. you can design highly instrument specific filter with formula language and use the filter in charts. For more information, refer to ***Specifying Tick Filtering for Data Series*** (on page 1122).

Bad Tick Filter in Alert Window

Alert window has its own bad tick filter because alert window is more sensitive to bad data than other types of windows. See ***Alert Window Setup*** (on page 425).

Backing Up, Moving to a New Computer, Restoring after a Hard Drive Crash

Print out this section and store it somewhere. You will not have the electronic version handy after a hard drive crash.

Backing Up

Everything you do in NeoTicker® is saved in the installation directory (e.g. `C:\Program Files\TickQuest\NeoTicker4`).

All you need to do is copy everything under this directory to your backup medium (e.g. CDR, tape, etc).

Moving to a New Computer and Restoring after a Hard Drive Crash

The procedure is the same.

1. Copy everything either from old computer or from the backup to the folder you intend to install NeoTicker®.
2. Re-installing NeoTicker®.

Step 2 is important because it registers NeoTicker® to Windows.

Remember to install the hardware key driver. The driver comes with the NeoTicker® CD.

Borderless Window

You can turn off window border for all function windows (charts, quote windows, etc) to better utilize your screen space.

Turning Window Border On/Off

From the function window's pop up menu, choose **Window Border** to toggle window border on/off.

Because time chart has multiple pop up menu, you may need to press the ESC key first before opening the pop up menu.

A Note About Ticker

Ticker window has its own borderless window support. For ticker window, choose **Show Window Border** from ticker window's pop up menu instead.

Moving and Resizing Borderless Window

You can move a borderless window using the ALT-M shortcut.

1. Click on the window you want to move to make it the active window.
2. Press the ALT-M key.
3. Use the arrow keys to move the window.
4. After you press any arrow key, you can use mouse to move the window.
5. Press Enter to accept the move. Press ESC to cancel the move.

You can resize a borderless window using the ALT-S shortcut.

1. Click on the window you want to resize to make it the active window.
2. Press the ALT-S key.
3. Use the arrow keys to resize the window.
4. After you press any arrow key, you can use mouse to resize the window.
5. Press Enter to accept the resizing. Press ESC to cancel the resizing.

Connecting to a Server

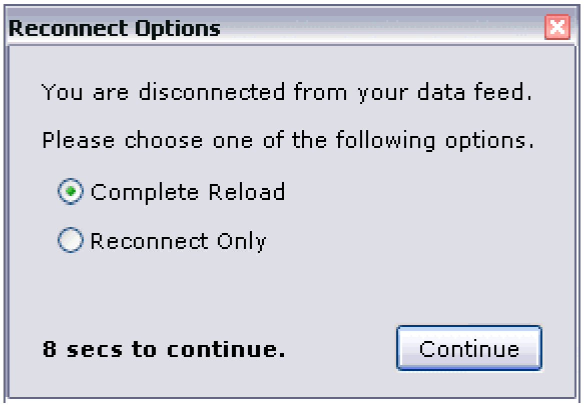
If you lost the connection to a real-time data server, or the connection does not seem right, e.g. some symbols takes a long time to get quotes, you can re-connect to a real-time server. To reconnect:

1 Choose **Program>Server Disconnect/Reconnect**.

2 Choose either **Connect to Server** or **Disconnect then reconnect to server**.

3 Press the **OK** button.

When NeoTicker® reconnects to the server, you have two options: **Complete Reload** and **Reconnect Only**.



You are disconnected from your data feed.

Please choose one of the following options.

● Complete Reload

○ Reconnect Only

8 secs to continue.

If you choose **Complete Reload**, all the data in the function window will be reloaded. This will fill in any data that is lost when NeoTicker® is disconnected, but reloading data is time consuming.

If you choose **Reconnect Only**, NeoTicker® will resume almost immediately, but any data that is lost when NeoTicker® is disconnected will not be filled back in.

Reconnection option can default to either **Complete Reload** or **Reconnect Only** depending on a number of factors. See ***Server Setup Reference, Misc Tab*** (see "Misc Tab" on page 521) for more information.

Closing a Function Window

When you close a function window, the action differs depending on how many groups the window belongs to.

If the window belongs to only one group, you will be asked whether you want to close the window. If you choose yes, the window will be closed and removed from the group.

If the window belongs to more than one group, you have two options:

1 Close the window only for the current group. In this case, the window is actually not closed, but removed from the current group.

2 Close the window from all groups. In this case, the window is closed and remove from all groups it belong.

Finding a Lost Window

Finding a Lost Function Window (Charts, Quote Windows)

If you desktop span across several monitors, it is very easily to 'lost' a window, i.e. you forgot which monitor you put the window into.

To find a lost window:

1 Click the name of the window in **Window** menu in main window or in group and window list. If the window is hidden beneath another window, it will be pop to the top.

2 If step 1 does not work, choose **Window>Arrange>Move Near Main Window**. This will move the window near the main window.

3 If step 2 does not work, choose **Window>Arrange>Restore Window to Default Position**. This will move the window to the default position, which is where this type of window will be created. In general the window will appear in your primary monitor.

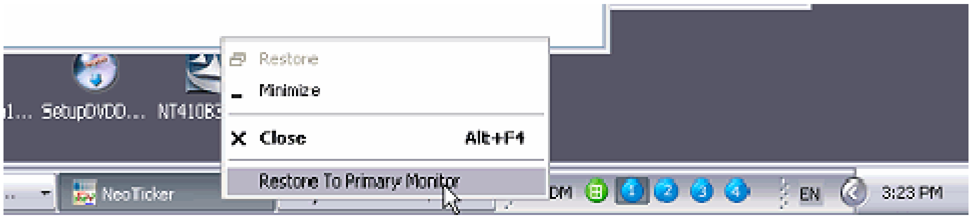
Finding a Lost Main Window

The Main Window can be lost if you have re-arrange your monitor configuration. To find a lost main window:

1 In Windows task bar, find NeoTicker®'s icon.

2 Right click on the icon to open pop up menu.

3 Choose **Restore To Primary Monitor**.



This operation will move Main Window to your primary monitor.

Function Window Arrangement

NeoTicker® provides commands to help you arrange multiple function windows to desired position. The window arrangement commands are under **Window>Arrange** in the main window.

The following window arrangement commands are available:

Move Near Main Window	Moves the active window near the main window
Restore Window to Default Position	Sends the active window to its default position
Cascade from Main Window	Cascades the function windows, starting from the main window
Cascade from Active Window	Cascades the function windows, starting from the active window
Cascade from Upper Left	Cascades the function windows, starting from the upper left corner of the screen
Tiled	Tiles the function windows
Horizontal	Tiles the function windows in horizontal tiles
Vertical	Tiles the function windows in vertical tiles

After you choose an arrange operation, NeoTicker® will prompt you:

- **Partial fit only, do not expand individual window** - NeoTicker® resize windows conservatively.
- **Complete resize, try to fit windows to fill the screen** - NeoTicker® resize the windows aggressively to fill screen.

Multiple Monitor Window Tiling

This subsection is applicable only if you use NeoTicker® in an multiple monitors environment.

Some video card manufacturer uses proprietary software to handle multiple monitors. In this case, Microsoft Windows and subsequently NeoTicker® do not know more than one monitor exists and the multiple monitor code in NeoTicker® will not be invoked.

Window arrangement in NeoTicker® supports multiple monitors. When you perform a window tiling, horizontal tiling or vertical tiling, NeoTicker® opens a dialog to let you control how you want the windows to be tiled.

You can choose the following actions:

Tile all windows, do not move windows across monitors
All windows are tiled. Each window will stay at the monitor where located.

Tile all windows, move window across monitors when needed
All windows are tiled. The windows are tiled from the first monitor third monitor and so on. Windows are moved across monitor when

Tile only the windows of the monitor that contains the active window
Only tile the the windows that are in the same window as the active will not be moved to another monitor.

Function Window Templates

Function window templates let you reuse a function window quickly. NeoTicker® provides a set of pre-built templates. You can also create your own template by saving a function window as a template.

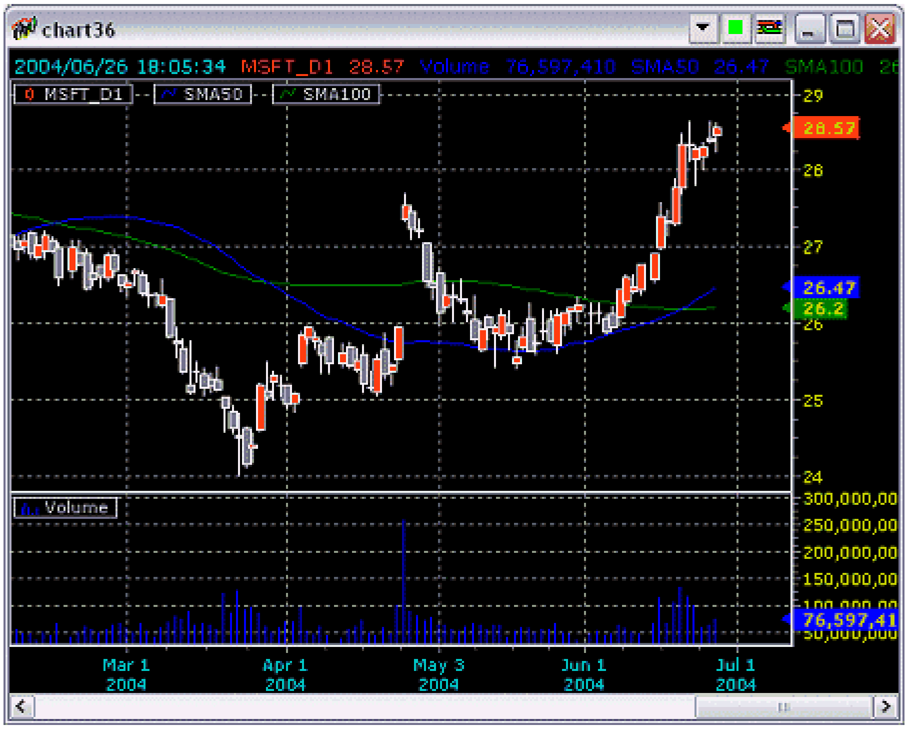
Opening a Template

We will use a chart template as the example. You can open templates of other window in a similar way.

Open a template by **Window>Open Template>Time Chart>50-100-Moving Average Dark**. You can also right click on the time chart tool button to open a chart with a template.

NeoTicker® will ask you to set up the data source for a symbol. Set up the data source as you normally would, and press the **Apply** button.

After you choose the symbol, NeoTicker® will use the template to open a chart. The 50-100-Moving-Average Dark template will create a dark background chart that contains a 50-bar moving average (SMA50 in the chart), a 100-bar moving average (SMA100 in the chart) and volume for MSFT.



Opening Template from Another Window

You can also open a template from any window that works on a symbol list (e.g. quote window).

Right click on a symbol in the window to open the pop up menu, choose **Send To>Template>Time Chart**, and choose one of the templates.

When you open a template from another window, the symbol in the you right click on will be sent to the template automatically.

Group and Function Window Files Cleanup

NeoTicker® creates group and function window files as you use it. Overtime, these files can buildup, making choosing files difficult.

We strongly recommend you make a backup of the group and window directory in the NeoTicker® installation before attempting the maintenance routine below.

To remove empty groups:

1 Choose **Program>User Preference** in the main window. Press the **Maintenance** tab.

2 Press the **Remove Empty Groups** button.

To remove unused function windows (windows that are created, but you have never save to the window):

1 Choose **Program>User Preference** in the main window. Press the **Maintenance** tab.

2 Press the **Remove Unused Windows** button.

Hour Bars

Hour bars is introduced in version 3.2.

With hour bars, you can construct hourly charts such as 1-hour, 2-hour, 4-hour charts. Hour bars are fully integrated into every aspect in NeoTicker®: time charts, quote window formula, scanning, trading systems and more.

Time Charts

In a time chart, you can simply create hourly chart just like minute charts. When you add a data series, you will have the hour option to choose from.

Hour bars are stamped at the 00 minute. For example, the 11:00am 1-hour bar consists of trading data from 10:00:00am to 10:59:59am.

In charts, bar space will be even if the trading hour is from 9:30am to 4:00pm. Specially the space between 4:00pm bar and the 10:00am bar occupies only a 30-minute interval. If your analysis calls for even bar space, you can either adjust the trading hours to land on 00 minute, or use bar driven time chart for hour bar charts.

Trading Time

When a hour bar starts depends on the start trading time.

For example, if trading starts at 9:30am, then the first 1-hour bar will be at 10:00am. It consists of trading data from 9:30:00am to 9:59:59am. The first 2-hour bar will be at 11:00am. It consists of trading data from 9:30:00am to 10:59:59am.

For a 24-hour chart, the counting starts at midnight. So the first 1-hour bar is at 1:00am, consists of trading data from 12:00:00am to 12:59:59am.

Formula

For formula that requires a time frame (quote window formula, cluster, dynamic grid), hour time frame is represented by the letter H. For example, the following quote window formula evaluates the RSI indicator on 2-hour bars.

```
RSIndex(0, H2, 10)
```

Cache Implication

Hour bars do not come from data vendor directly. Instead, they are synthesized from minute data, very much the same way second bars are synthesized from tick data. Therefore, disk cache and RAM Cache minute bar settings affect hour bar.

For users, the most important implication will be for indicator calculation in formula. The default minimum minute settings for RAM Cache's Bar Per Series setting is 250. So a formula starts only with 4 1-hour bars. The default maximum minute settings is 2000. So you will have at most 33 1-hour bars available.

If your formula recalls more than 250 minute of data, you will need to increase the minimum and maximum settings accordingly.

Scripts/IDL

The `ppHour` enumeration is introduced for any methods that expects/returns time frame information.

How to Override an Internal Indicator with Your Custom Version

NeoTicker® has many built-in indicators. These indicators can be overridden with your customized version. In general, it is safe to override internal indicators (majority of the built-in indicators), but not built-in indicators that are script or formula-based.

Here is an example of overriding the Exponential Moving Average Indicator.

1. Create a new formula indicator called "xaverage", which is the exact same function name of the built-in exponential moving average indicator. You need to use the script editor to create this indicator. For information on how to create formula indicator, refer to ***Tutorial: Making an Indicator with Formula Language, Example 1*** (on page 364) as an example. For list of function names, you can look it up in the main window under **Help>Indicator Quick Reference**. The function names are listed under the **Function** column.
2. Set up this new formula indicator to have 1 parameter called Period, and having the following line of formula:

   ```
   plot1 := qc_average(data1, param1)
   ```
3. Install this indicator and save it as `xaverage_override.for`.
4. Choose from main window, **Program>Indicator Manager** to open Indicator Manager.
5. In Indicator Manager, locate the item Exponential Moving Average.
6. You will find that the current in use version is based on the one we have just installed as oppose to the built-in one.
7. To make the override in effect for next start up of NeoTicker®, check the **Startup Load Script** box in Indicator Manager. Then click the **Save Startup Settings** button.

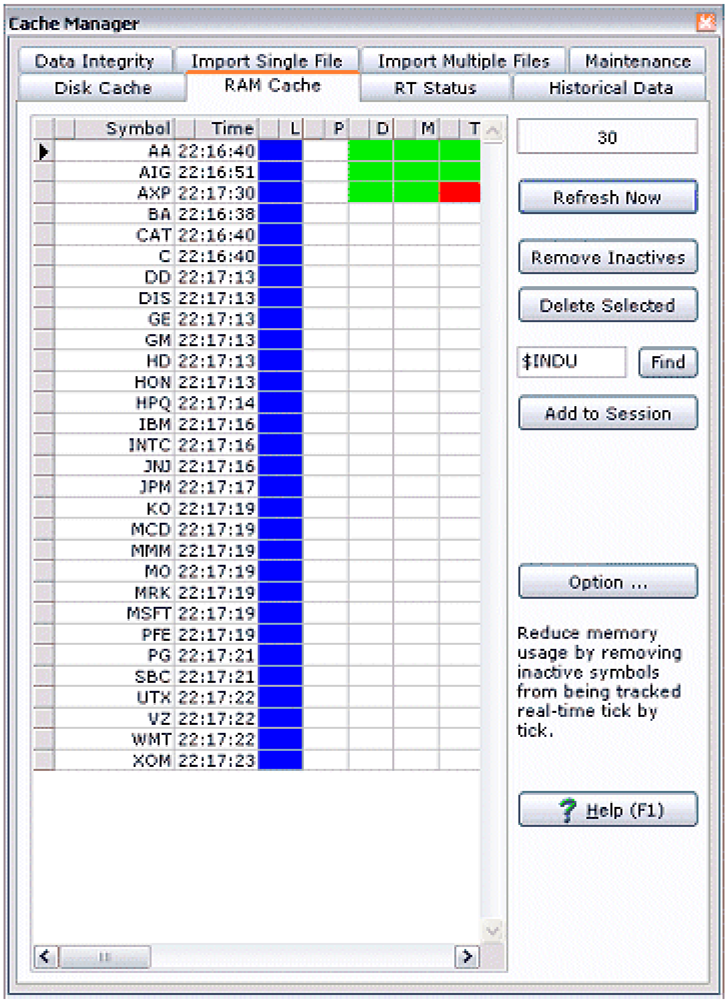
Monitor Data Status in RAM Cache

To monitor data status in RAM Cache:

1 Choose **manager>cache** from the main window to open the cache manager.

2 Press the **RAM Cache** tab.

3 You need to press the **Refresh Now** button to refresh the display.



Symbol	Time	L	P	D	M	T
AA	22:16:40	🟦		🟩	🟩	🟩
AIG	22:16:51	🟦		🟩	🟩	🟩
AXP	22:17:30	🟦		🟩	🟩	🟥
BA	22:16:38	🟦				
CAT	22:16:40	🟦				
C	22:16:40	🟦				
DD	22:17:13	🟦				
DIS	22:17:13	🟦				
GE	22:17:13	🟦				
GM	22:17:13	🟦				
HD	22:17:13	🟦				
HON	22:17:13	🟦				
HPQ	22:17:14	🟦				
IBM	22:17:16	🟦				
INTC	22:17:16	🟦				
JNJ	22:17:16	🟦				
JPM	22:17:17	🟦				
KO	22:17:19	🟦				
MCD	22:17:19	🟦				
MMM	22:17:19	🟦				
MO	22:17:19	🟦				
MRK	22:17:19	🟦				
MSFT	22:17:19	🟦				
PFE	22:17:19	🟦				
PG	22:17:21	🟦				
SBC	22:17:21	🟦				
UTX	22:17:22	🟦				
VZ	22:17:22	🟦				
WMT	22:17:22	🟦				
XOM	22:17:23	🟦				

Reduce memory usage by removing inactive symbols from being tracked real-time tick by tick.

D, M, T Columns

D, M, T stands for Daily, Minute and Tick. They are the raw data that is requested from the data feed. Note that second data is constructed from tick.

A green cell indicates the data is ready for use. A red cell indicates data is currently being loaded. A white cell indicates the data has not started loading.

DB, MB, TB Columns

DB, MB, TB stands for Daily Buffer, Minute Buffer and Tick Buffer. The numbers under this column tells you how many 1-day bars, 1-minute bars and 1-tick driven bars are available in the RAM Cache.

When a data series is constructed, if the amount of data needed can be constructed with what is available in the RAM Cache, speed is very fast. For example, a data series of 100 5-minute bar will be covered by a Minute Buffer of the size 500 or more.

If there are not enough data in the RAM Cache, the disk cache will be queried for data. If there are not enough data in the disk cache, the data feed will be queried.

DC, IC Columns

DC stands for Daily Series Count and IC stands for Intraday Series Count. They are the number of data series that are currently inside the RAM Cache. For example, a 5-minute data series and a 1-minute data series are counted as two data series.

The numbers shown are data series in use vs. data series available. For example, 2/5 stands for 5 data series and 2 of them are actively in use.

Packaging Groups into Single File

You can package multiple groups and their associate windows into a single file. Possible usages:

- Archiving of a particular setup.
- Sharing setup with other users.
- Sending a setup to TickQuest for diagnostic purposes.

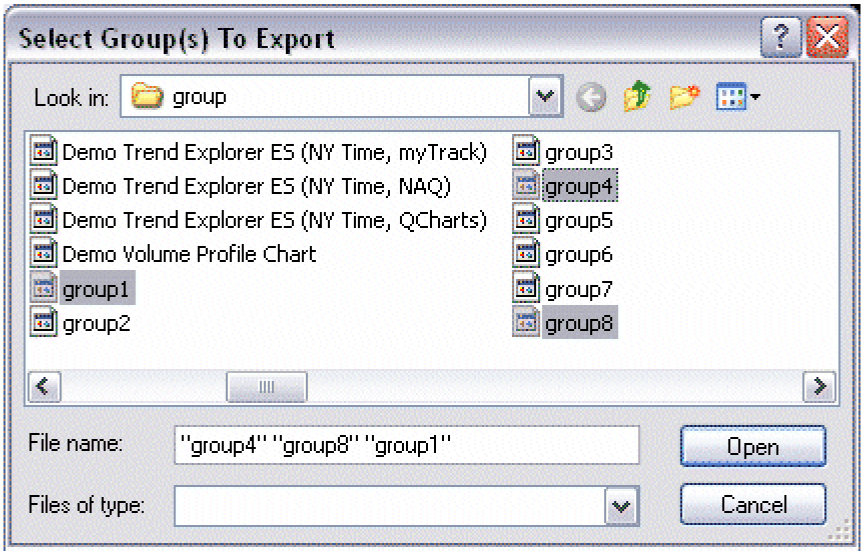
Packaged file as a default .TXT extension.

Export to a Package File

1 Save the groups and windows you want to package.

2 Choose **Group>Export Package**.

3 The **Select Groups(s) to Export** dialog will open to let you select groups to export. You can select multiple group by holding the CTRL key while selecting the group. Press the **Open** button after you select the groups.

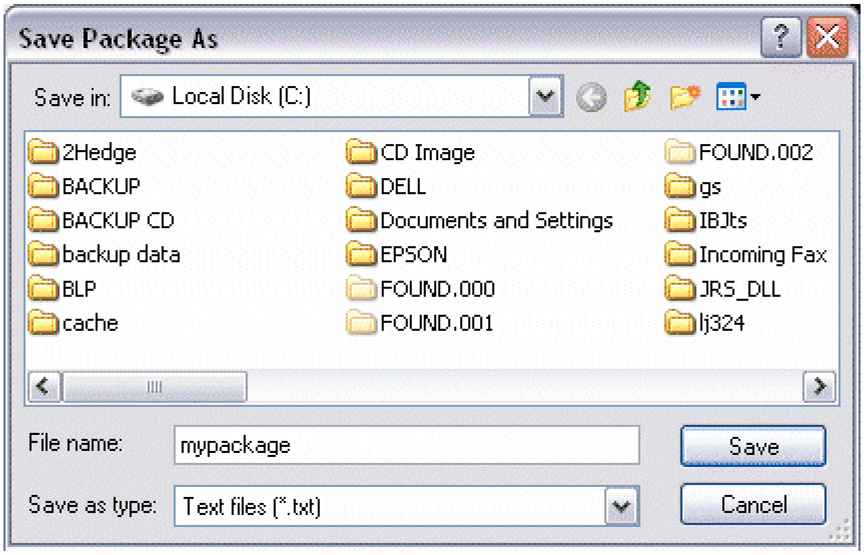

Select Group(s) To Export
Demo Trend Explorer ES (NY Time, myTrack)
Demo Trend Explorer ES (NY Time, NAQ)
Demo Trend Explorer ES (NY Time, QCharts)
Demo Volume Profile Chart
group1
group2
group3
group4
group5
group6
group7
group8
Open
Cancel

Look in: group

File name: "group4" "group8" "group1"

Files of type:

4 The **Save Package As** dialog will open. Choose a file name for the package and the directory you want to save the package file to, and press the **Save** button.



Save in: Local Disk (C:)

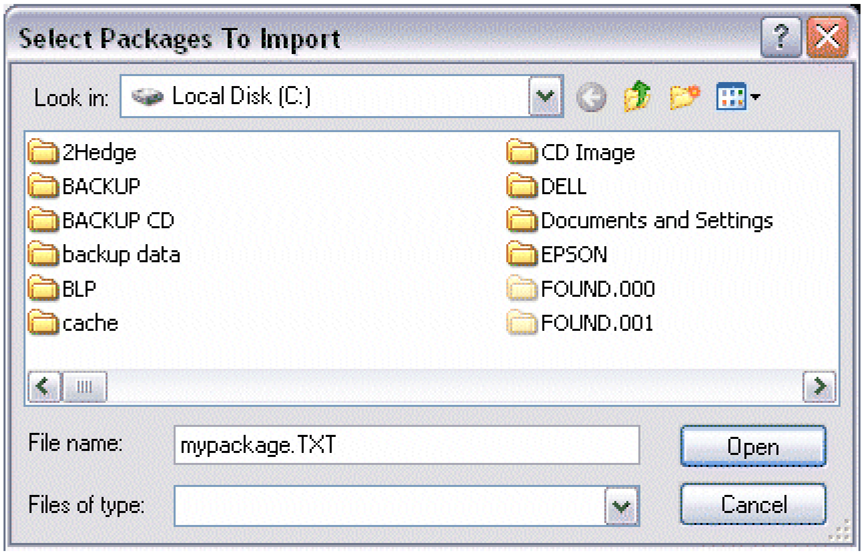
File name: mypackage

Save as type: Text files (*.txt)

Importing a Package File

1 Choose **Group>Import Package**.

2 A dialog will open to let you select package to import.

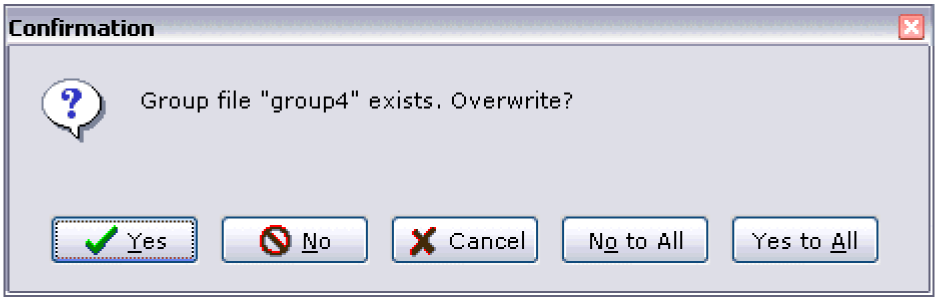


Look in: Local Disk (C:)

File name: mypackage.TXT

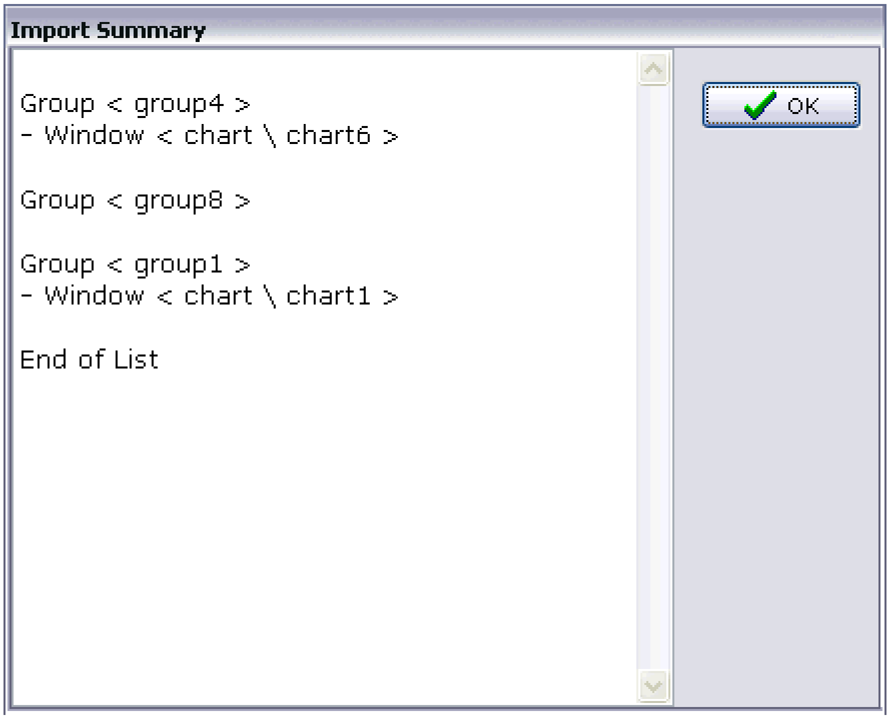
Files of type:

3 If the package contains groups and windows that have the same name as your current groups and windows, you will be prompted for action.



Group file "group4" exists. Overwrite?

4 A summary of imported groups and windows will be provided.



```
Group < group4 >
- Window < chart \ chart6 >

Group < group8 >

Group < group1 >
- Window < chart \ chart1 >

End of List
```

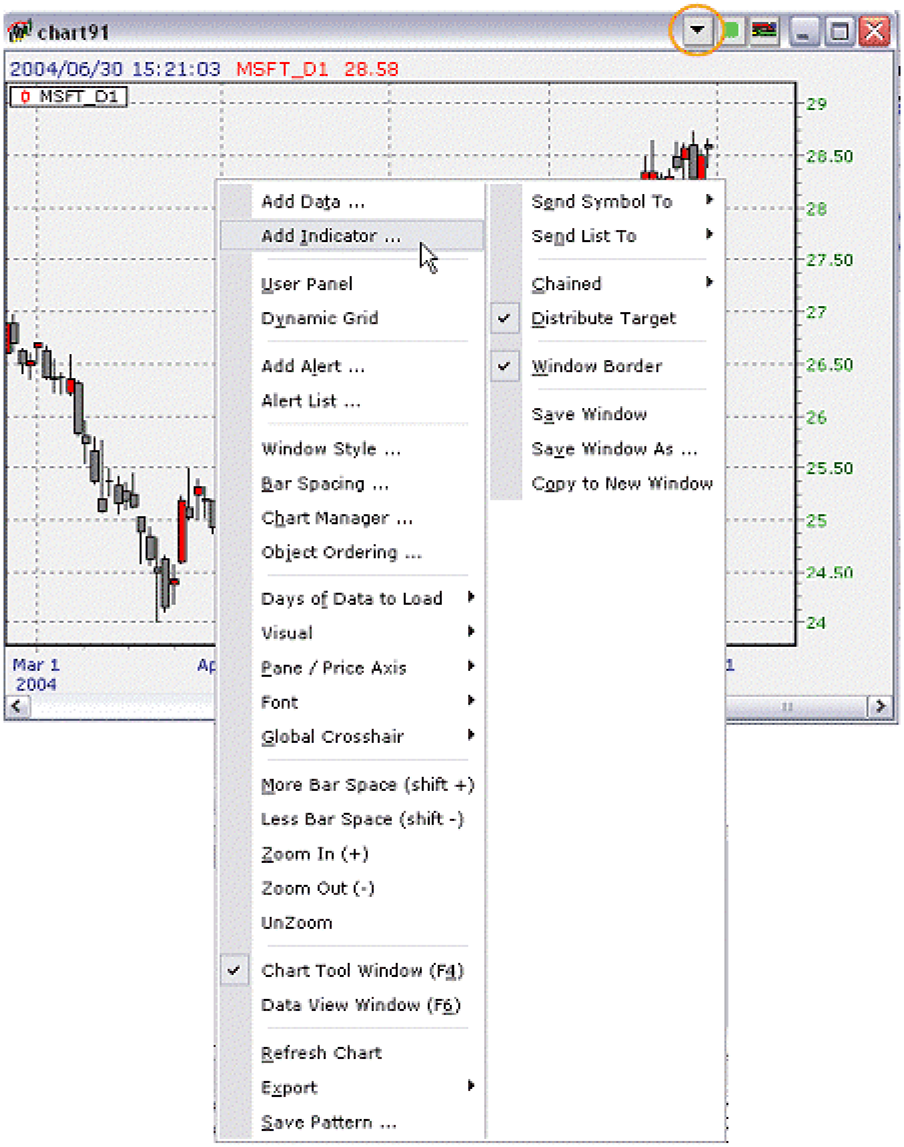
Once after the import is completed, you can access all these groups and function windows just like the ones you have created locally.

For function windows that depend on indicators, you will need to ensure that these indicators are installed. Otherwise the function window will not to able to locate the indicator.

Pop up Menu

You can access features of most function windows (charts, quote windows, etc) from their pop up menu.

To open pop up menu, right click on the window. Alternatively, you can use the menu button ▼ on the window's caption to access the menu.

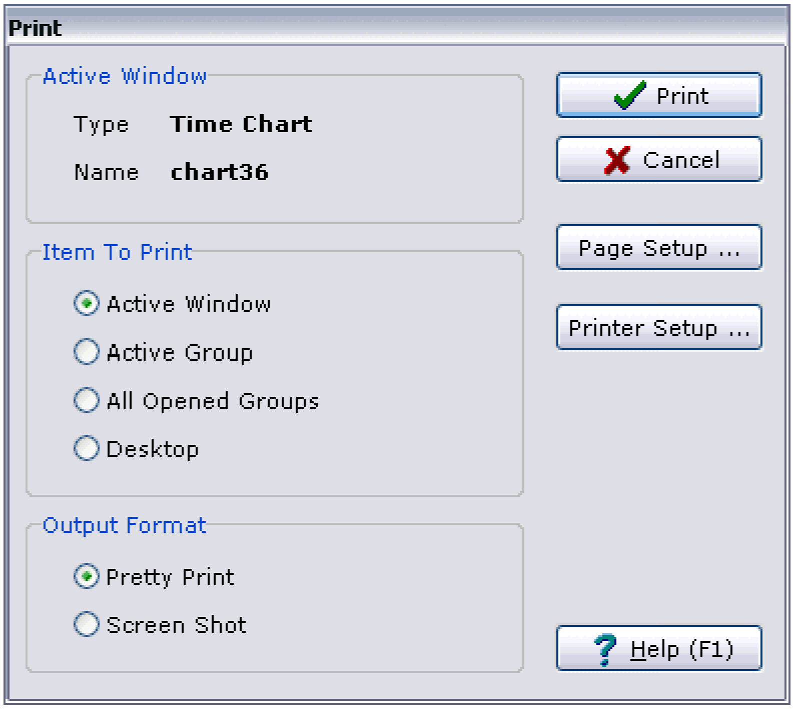


Menu Button and Time Chart

Most function windows have only one pop up menu. Time charts have several pop menus. When you use right click to open pop up menu, the menu opened depends on the current selection. On the other hand, menu button always open the main pop up menu, regardless of the current selection.

Printing

NeoTicker® offers a variety of printing options. From the Main Window, choose **Program>Print**. A dialog window is opened to let you choose printing options.


Print

Active Window

Type Time Chart

Name chart36

Item To Print

● Active Window

○ Active Group

○ All Opened Groups

○ Desktop

Output Format

● Pretty Print

○ Screen Shot

Item To Print

You can choose the item to print here.

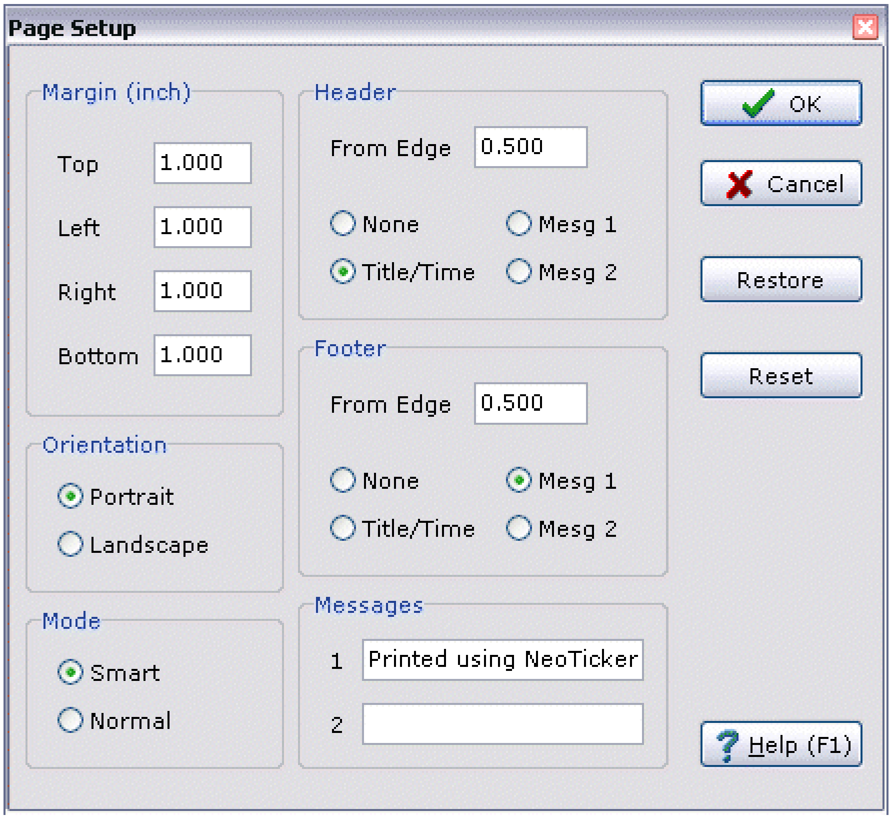
- **Active Window** - the active function window will be printed (default)
- **Active Group** - batch print the windows in the active group.
- **All Opened Groups** - batch print the windows in all opened groups
- **Desktop** - print the screen shot of each monitor on a separate page.

Output Format

- If you choose **Screen Shot**, the screen shot of the window will be printed.
- If you choose **Pretty Print**, the program will make the output looks good on a printed page. Further pretty print options are in Page Setup.

Page Setup

You can press the **Page Setup** button for more printing options. You can also launch Page Setup by choosing **Program>Page Setup** in the Main Window.



Page Setup

Margin (inch)

Top 1.000

Left 1.000

Right 1.000

Bottom 1.000

Orientation

● Portrait
○ Landscape

Mode

● Smart
○ Normal

Header

From Edge 0.500

○ None ○ Mesg 1
● Title/Time ○ Mesg 2

Footer

From Edge 0.500

○ None ● Mesg 1
○ Title/Time ○ Mesg 2

Messages

1 Printed using NeoTicker

2

Margin (inch) You can specify the print margins

Orientation You can specify the print orientations

Mode This value is used only when Pretty Print is on.

If **Smart** mode is chosen, NeoTicker® will adjust the color of the printed window to make the printout looks good.

If **Normal** mode is chosen, NeoTicker® will not make any adjustment to colors.

Header You can print a header message on the print out. Choices are **None**, **Title/Time**, **Mesg 1** and **Mesg 2**. Mesg 1 and Mesg 2 are specified in the Messages section

Footer Similar to **Header**, but it is for printing the footer

Messages Lets you specify optional printing messages in the header and footer

Restore Restores your saved printing default

Reset Resets to NeoTicker®'s printing default

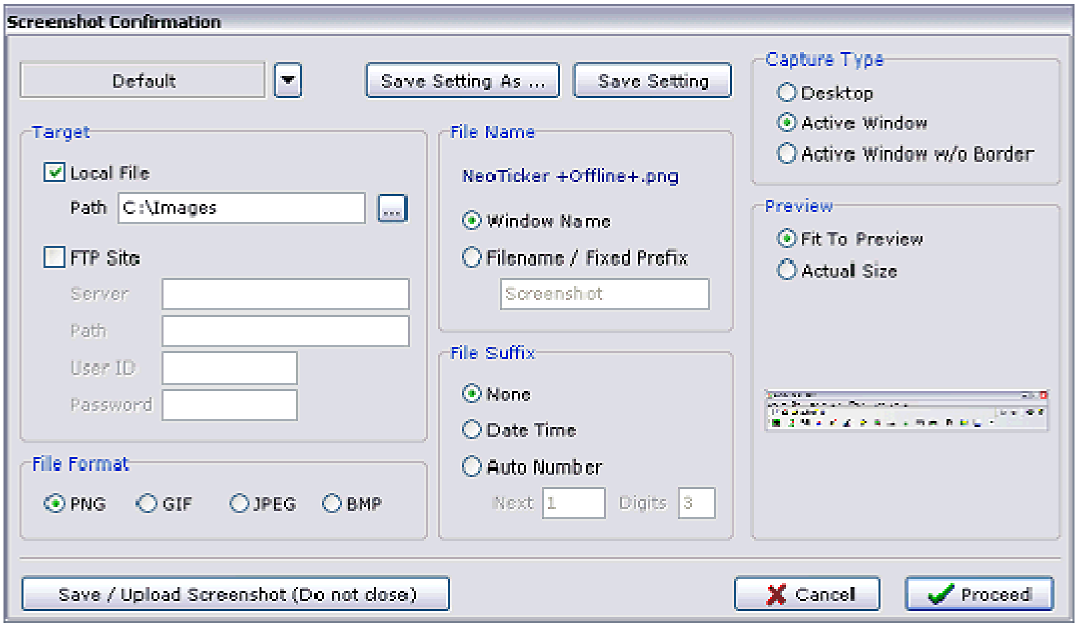
Printer Setup

In Main Window, choosing **Program>Printer Setup** will launch the standard Windows printer setup dialog, from which you can select the destination printer and adjust printer specific settings.

Screen Shots

You can take screen shots by pressing the CTRL-J. A dialog is opened to let you select destination, file name convention, etc.

Screen shot tool is also available under **Tool** menu in Grid Optimizer.



Screenshot Confirmation

Default

Target
☑ Local File
Path C:\Images
□ FTP Site
Server
Path
User ID
Password

File Format
● PNG ○ GIF ○ JPEG ○ BMP

File Name
NeoTicker +Offline+.png
● Window Name
○ Filename / Fixed Prefix
Screenshot

File Suffix
● None
○ Date Time
○ Auto Number
Next 1 Digits 3

Capture Type
○ Desktop
● Active Window
○ Active Window w/o Border

Preview
● Fit To Preview
○ Actual Size

Local File

To save screen shot to a local file on your hard drive:

1. Under **Target**, make sure **Local File** is checked. **Path** is a valid file path.
2. Under **File Name**, verify the file name shown.
3. Under **Capture Type**, check to see if screen shot is capturing what you want.
4. Press **Proceed** to save the screen shot.

FTP (Website)

To upload screen shot to a website:

1. Under **Target**, make sure **FTP Site** is checked. **Server** is the website you are uploading to. **Path** is a the path on the server to upload to. **User ID** and **Password** are set correctly.
2. Under **File Name**, verify the file name shown.
3. Under **Capture Type**, check to see if screen shot is capturing what you want.
4. Press **Proceed** to upload the screen shot.

File Format

The following formats are supported: PNG, GIF, JPEG, BMP.

For saving to your hard drive as a local file, we recommend PNG or BMP format.

For web usage, we recommend PNG or GIF format.

File Name Options

You have options to auto create file name for easy file management.

1 Under **File Name**, you can choose to use the window name, or a fixed file name.

2 Under **File Suffix**, you can choose to append a suffix to the file name/Window name. The suffix can be **date time** or Auto Number. For **Auto Number**, the suffix is auto increasing and your screen shot will have a unique file name.

Capture Type

You can capture desk top, active window or active window without border. The capture type can be set after you press CTRL-J. There is no need to retake screen shot when you switching between the different capture types.

Multiple Settings

You can create multiple screen shot settings. Multiple settings are useful for managing multiple targets. For example, if you have 2 websites you want to upload to, you can create two settings, one for each website.

To create a setting:

1 Edit the options.

2 Press **Save Setting As** button, choose a file name for the setting and save the settings.

Once saved, you can reuse the setting by pressing the down triangle button.

Sending Symbol and Symbol List Between Windows

Sending Single Symbol

You can send symbols between function windows.

Sending function window can be any function window that works on symbols. Here are the valid sending function windows:

- Quote window
- Time and sales
- Volume distribution
- Time chart
- Ticker
- Alert
- Pattern scanner
- Option chain
- Level 2
- Top list
- Cluster

The receiving function window must be a window that works on a single symbol, except for time chart. Here are the valid receiving function windows:

- Time and sales
- Volume distribution chart
- Time chart
- Option chain
- Level 2

To send a symbol, right click on the symbol to open the pop up menu, and choose **Send Symbol To**. You can send the symbol to an existing window, or to a new window.

Sending Symbol List

You can send symbol list between windows.

The sending and receiving function windows must be able to work on symbol list. Valid sending and receiving function windows are:

- Quote window
- Ticker
- Pattern scanner
- Cluster

- Time chart (send only)

To send a symbol list, right click on the window to open the pop up menu, and choose **Send List To**. You can send to an existing window or to a new window.

Trading Topics

Real-time Trade Simulation

You can perform paper trade in real-time with trade simulator. Trade simulator is a realistic broker simulator that can take orders from you and execute the trades.

For more information, see ***Trade Simulator*** (on page 755).

System Trading

If you intend to write mechanical trading systems, there are two routes you can go:

- Use Backtest EZ. You do not need any programming to use Backtest EZ. See ***Backtest EZ*** (on page 1275).
- Use `Trade` object. You can program in a scripting language or external programming language for maximum performance and flexibility. See ***Trade Object*** (on page 1577).

Analyzing Trading Performance

Trading performance from trade simulation, `Backtest EZ` and `Trade` object can all be analyzed using system performance viewer. System performance viewer is a sophisticate statistical analyzer that can help you locating issues with your trading.

For more information, see ***System Performance Viewer*** (on page 951).

Managing Past Trading Performance

Overtime, you will accumulate many trading performance analysis. These analysis can be saved into files and you can review them later. System performance manager is a tool to help you manage analysis files from the past.

For more information, see ***System Performance Manager*** (on page 947).

Turn Key Operation

This section describes techniques that will make running NeoTicker® a turn key operation, i.e. when your computer starts, it will run NeoTicker® .

Auto Load Groups

You can automate NeoTicker® to auto load groups when it starts.

1. In Windows, create a short cut to start NeoTicker®.
2. Right click on the short cut and choose **properties**.
3. Under the shortcut tab, the field target specifies how NeoTicker® starts, e.g. `C:\Program Files\TickQuest\NeoTicker3\NeoTicker.exe`
4. Append to the field the option -OG and the group name, e.g. `C:\Program Files\TickQuest\NeoTicker3\NeoTicker.exe" -OG group1`
5. Instead of -OG, you can use the -OPG argument (without parameter) to open groups from previous session.
6. Use the shortcut to start NeoTicker®.

The command `-OG group1` tells NeoTicker® to auto open the group when NeoTicker® starts. You can add as many `-OG` as you like and create as many shortcut as you like.

Auto Reboot Windows

You can schedule Windows to automatically reboot itself at a specific time, e.g. nightly. Nightly reboot can resolve many problems that is inherited in the operating system and application software.

1. Locate a shutdown utility. Microsoft provides one called `shutdown.exe` and many free shutdown utilities can be found in the Internet.
2. Use Windows' schedular to schedule the shutdown utility to run at a specific time.
3. To force NeoTicker® to shutdown, the shutdown utility should be configured to force shutdown, and reboot afterward.

Auto Login

Microsoft provides a utility called Tweak UI. This utility allows you to setup Windows to automatically login to a user's account when Windows starts.

Auto Start NeoTicker®

Each user has a startup folder where you can put a shortcut in it. The shortcut tells Windows to start certain programs when an user logs in. You can simply create a NeoTicker® shortcut in this folder.

The location of the startup folder is Windows and user dependent. You can use Windows' searching function to locate the startup folder.

Putting it Together

If you set up all the automatic procedure described above, the following will happen:

1. At a specified time, Windows will reboot.
2. After the reboot, Windows will login to a specific user.
3. The user's startup folder will dictate NeoTicker® to start.
4. The `-OG` option will tell NeoTicker® to open certain groups.

Window Chaining

Window chaining lets you connect function windows together so you can quickly change the symbol in a series of windows.

A tutorial is available to help you work through window chaining ***Tutorial: Windows Chaining*** (on page 165).

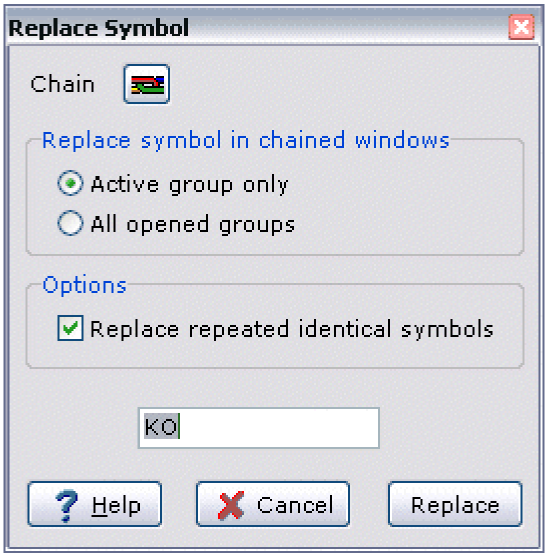
Directly Sending a Symbol to a Chain

Many window such as quote window that works on symbol list can send window directly to chain. You can simple double click on a symbol to send symbol to chain.

Sending a Symbol using a Dialog

To send symbol to a chain, choose **Replace Symbol in Chain** in the main window or press the F8 key.

A dialog is opened:



Replace Symbol

Chain

Replace symbol in chained windows

● Active group only

○ All opened groups

Options

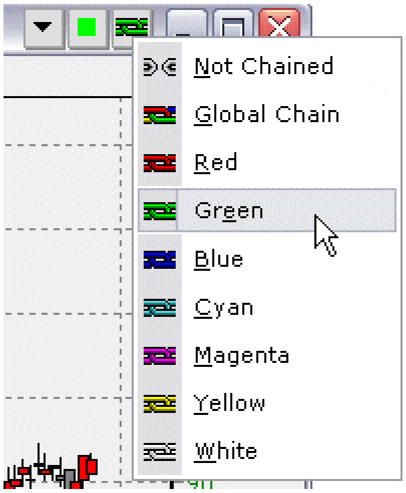
☑ Replace repeated identical symbols

KO

You can enter the symbol and press the **Replace** button.

Changing Chain, Global Chain and Un-Chaining

To turn on/off chaining or switch the chain for a window, press the chain button on the window's title bar. The chain button looks this or this , depending on the size of your window caption. A menu will be open:



You can choose the chain from the menu.

If you choose **Not Chained**, the window will not send or receive chain symbol.

If you choose **Global Chain**, the window will send to or receive from all chains.

You can also use the group and window list to change chains. To open group and window list, choose **Group>Group and Window List** from the main window.

Multiple Symbol Replacement

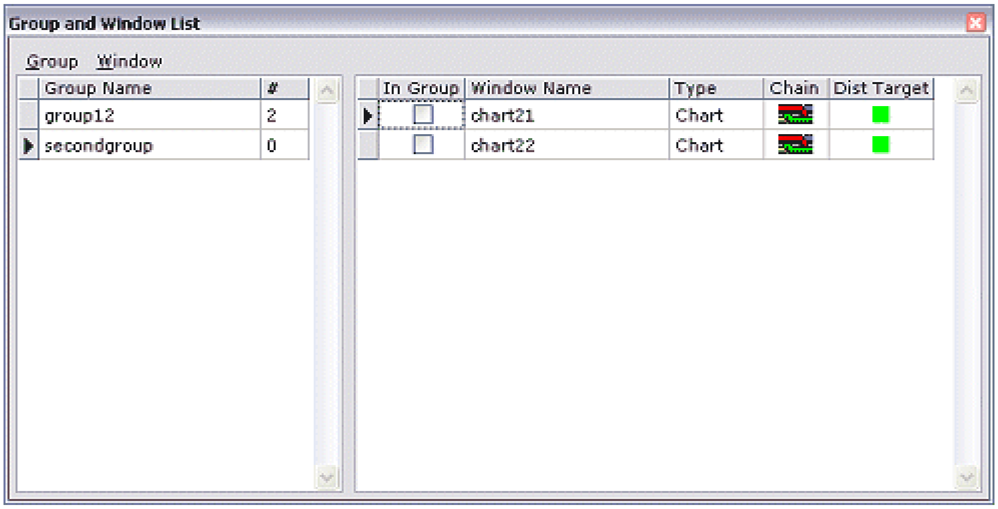
Chain symbol replacement (F8) allows you to replace multiple identical symbols in a time chart. To use this feature, check the **Replace repeated identical symbol** option in the dialog.

This feature is useful when you have a time chart that contains multiple time frames of an identical symbol, and need to replace them all by another symbol.

Group and Window List

Group and window list is a window that lets you manage groups and windows.

To open, choose **Group>Group and Window List** from the main window.



Group Name	#
group12	2
secondgroup	0

In Group	Window Name	Type	Chain	Dist Target
□	chart21	Chart		🟩
□	chart22	Chart		🟩

The left hand side of the group and window list is the group list. The group list shows you all the groups that are opened.

Item	Function
Group Name	The name of the opened groups.
#	Number of function windows in the group.
▶	Indicates the active group. You can click on a group to make it the active group.

The right hand side is the function window list. The function window list shows you all the opened windows.

Item	Function
In Group	Whether the function window belongs to the active group. You can click the check box to add or remove the function window to/from the active group.
Window Name	Name of the function window.
Type	Type of the function window.
Chain	Chaining information. You can click on this to change chain.
Dist Target	Whether the window is a distribution target of symbol list manager.
▶	Indicates the active window. You can click on a window to make it the active window.

Each function window must belong to one group. It can belong to multiple groups.

The menus in group and window list is exactly the same as the **Group** and **Window** menus in main window.

Indicator Management

Where the Indicators are Located

Internal NeoTicker® indicators are part of NeoTicker®.

External NeoTicker® indicators (indicator written with formula, scripting languages, and external programming language) are stored in the `indicator` directory in the NeoTicker® installation, e.g. `C:\Program Files\TickQuest\NeoTicker3\Indicator`.

Each external indicator is a single file in this directory.

How to Remove an Indicator

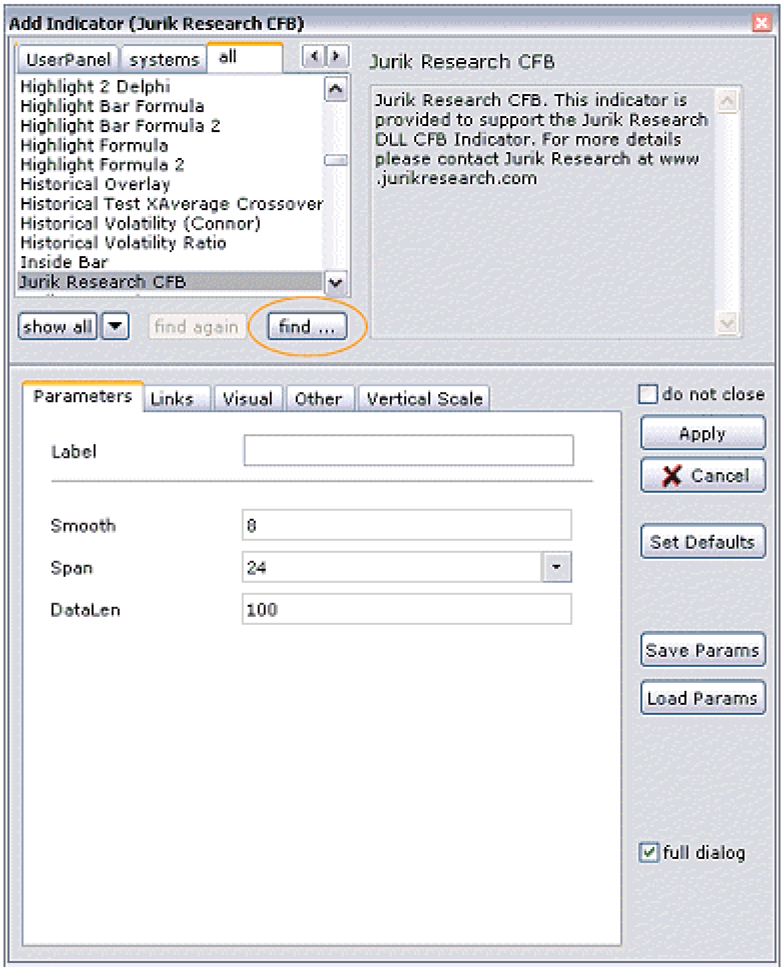
To remove an external indicator:

1. Make sure NeoTicker® is closed.
2. Locate the indicator in the indicator directory. NeoTicker® convention is the file name of the indicator is the same as the short name of the indictor, plus a type suffix (e.g. `scan_sample_daily_report.pas`).
3. Remove the file from the indicator directory.

You cannot remove internal indicators. Also, it is important NeoTicker® is closed before you remove the indicator file.

Indicator Searching

NeoTicker® provides searching capability to help locate the indicator in your mind.



Add Indicator (Jurik Research CFB)

Jurik Research CFB

Jurik Research CFB. This indicator is provided to support the Jurik Research DLL CFB Indicator. For more details please contact Jurik Research at www .jurikresearch.com

Label

Smooth 8

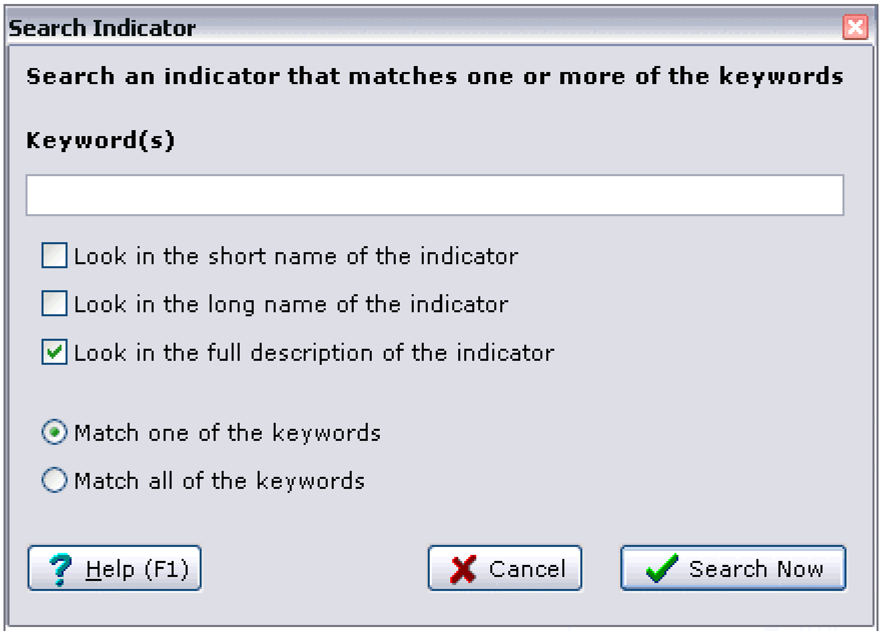
Span 24

DataLen 100

☐ do not close

☑ full dialog

When you add an indicator, you can press the **find** button. A search dialog will be opened. You can specify key words to help you find the indicator. You can specify where the key words are matched.



Search Indicator

Search an indicator that matches one or more of the keywords

Keyword(s)

□ Look in the short name of the indicator

□ Look in the long name of the indicator

☑ Look in the full description of the indicator

● Match one of the keywords

○ Match all of the keywords

The short name is the function name of the indicator. This is the same name that is displayed in the chart legend.

The long name is the name you see in the Add Indicator Window.

The full description is the text you see in the Add Indicator Window that describes what the indicator does.

If you do not see the indicator you seek for in the first search, simply press the **find again** button to search for the next indicator that matches the searching criteria.

Searching is not in any particular order. Using the **find again** button is the only way to go through all indicators that match the description.

Indicator List Manager

Overview

The indicator list manager is the place where you organize the indicators into indicator lists.

Indicator list is just a way of organizing indicators. There is a special indicator list called **all**. The **all** list is always available and lists all the indicators. You cannot add or remove indicators from the **all** list. You cannot remove the **all** list.

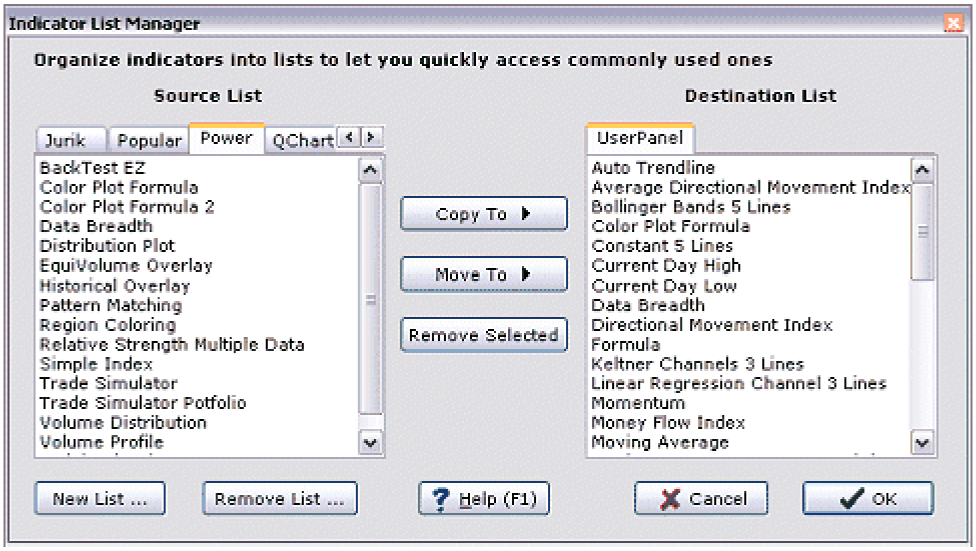
Other indicator list either comes with NeoTicker®, or are user created. The list are for you to organize indicators into different lists. You can freely create and remove these indicator list. Indicators can be added and remove from the indicator list and indicators can belong to more than one indicator list.

Indicator list is just a management tool. Indicator list does not affect the actual indicators.

Opening Indicator List Manager

To open indicator list manager:

- Choose **Manage>Indicator List** in the main window, or
- Press the **Manage** button. The **Manage** button is visible almost always when a list of indicators are shown.



Organize indicators into lists to let you quickly access commonly used ones

BackTest EZ
Color Plot Formula
Color Plot Formula 2
Data Breadth
Distribution Plot
EquiVolume Overlay
Historical Overlay
Pattern Matching
Region Coloring
Relative Strength Multiple Data
Simple Index
Trade Simulator
Trade Simulator Potfolio
Volume Distribution
Volume Profile

Auto Trendline
Average Directional Movement Index
Bollinger Bands 5 Lines
Color Plot Formula
Constant 5 Lines
Current Day High
Current Day Low
Data Breadth
Directional Movement Index
Formula
Keltner Channels 3 Lines
Linear Regression Channel 3 Lines
Momentum
Money Flow Index
Moving Average

Creating Indicator List

Press the **New List** button in indicator list manager.

Copying and Moving Indicators between Indicator Lists

In the indicator list manager, the left side is the source indicator list and the right side is the destination indicator list. When you copy or move indicators, the indicators are always moved from the left to the right.

To copy or move an indicator, select the indicator and press the **Copy To** or **Move To** buttons.

Removing Indicator List

To remove an indicator, select the indicator at the left and press the **Remove List** button.

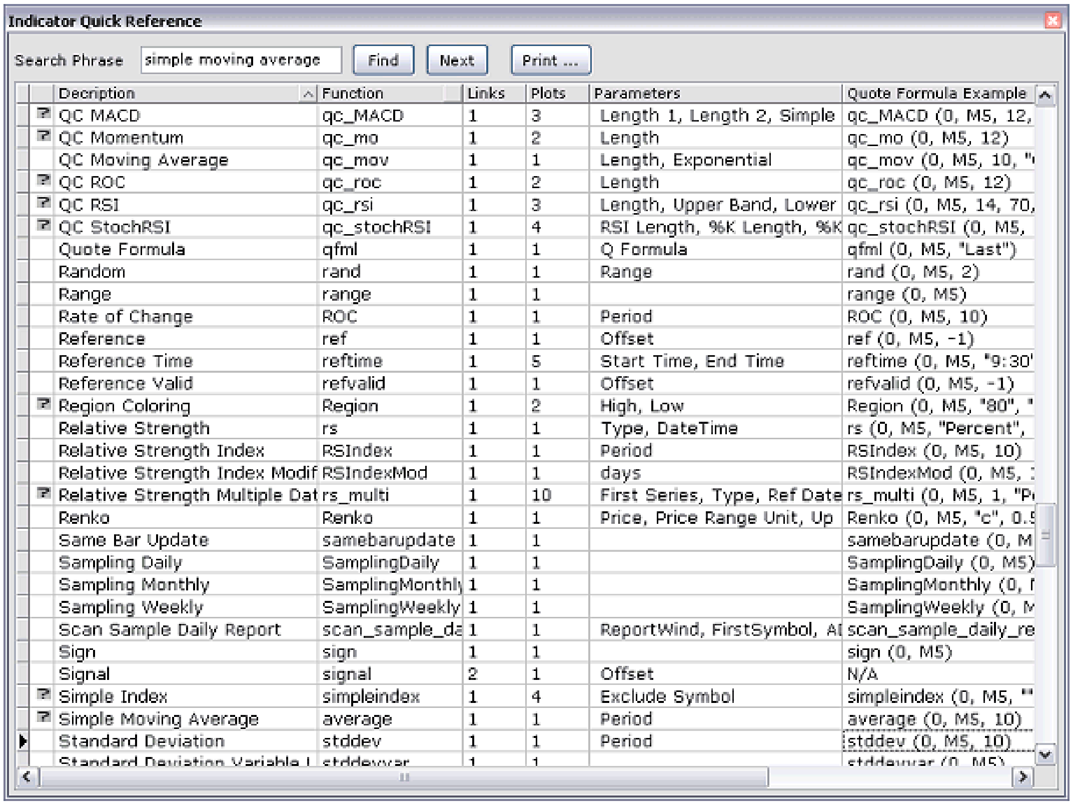
Removing an indicator from an indicator list does not delete the indicator from NeoTicker®. The operation only makes the indicator unavailable to the indicator list.

Because you cannot add or remove indicators into the **all** indicator list. Therefore the **all** list is not shown at the right side.

Indicator Quick Reference

Indicator Quick Reference lets you quickly browse the available indicators. To open, choose **Help>Indicator Quick Reference** from main window.

You can view indicators by description or short name, and you can search indicator by using the **Find** button.



Search Phrase: simple moving average

Decription	Function	Links	Plots	Parameters	Quote Formula Example
QC MACD	qc_MACD	1	3	Length 1, Length 2, Simple	qc_MACD (0, M5, 12,
QC Momentum	qc_mo	1	2	Length	qc_mo (0, M5, 12)
QC Moving Average	qc_mov	1	1	Length, Exponential	qc_mov (0, M5, 10, "
QC ROC	qc_roc	1	2	Length	qc_roc (0, M5, 12)
QC RSI	qc_rsi	1	3	Length, Upper Band, Lower	qc_rsi (0, M5, 14, 70,
QC StochRSI	qc_stochRSI	1	4	RSI Length, %K Length, %K	qc_stochRSI (0, M5,
Quote Formula	qfml	1	1	Q Formula	qfml (0, M5, "Last")
Random	rand	1	1	Range	rand (0, M5, 2)
Range	range	1	1		range (0, M5)
Rate of Change	ROC	1	1	Period	ROC (0, M5, 10)
Reference	ref	1	1	Offset	ref (0, M5, -1)
Reference Time	reftime	1	5	Start Time, End Time	reftime (0, M5, "9:30'
Reference Valid	refvalid	1	1	Offset	refvalid (0, M5, -1)
Region Coloring	Region	1	2	High, Low	Region (0, M5, "80", "
Relative Strength	rs	1	1	Type, DateTime	rs (0, M5, "Percent",
Relative Strength Index	RSIndex	1	1	Period	RSIndex (0, M5, 10)
Relative Strength Index Modif	RSIndexMod	1	1	days	RSIndexMod (0, M5,
Relative Strength Multiple Dat	rs_multi	1	10	First Series, Type, Ref Date	rs_multi (0, M5, 1, "P
Renko	Renko	1	1	Price, Price Range Unit, Up	Renko (0, M5, "c", 0.
Same Bar Update	samebarupdate	1	1		samebarupdate (0, M
Sampling Daily	SamplingDaily	1	1		SamplingDaily (0, M5)
Sampling Monthly	SamplingMonthl	1	1		SamplingMonthly (0,
Sampling Weekly	SamplingWeekly	1	1		SamplingWeekly (0, M
Scan Sample Daily Report	scan_sample_da	1	1	ReportWind, FirstSymbol, Al	scan_sample_daily_re
Sign	sign	1	1		sign (0, M5)
Signal	signal	2	1	Offset	N/A
Simple Index	simpleindex	1	4	Exclude Symbol	simpleindex (0, M5, ""
Simple Moving Average	average	1	1	Period	average (0, M5, 10)
Standard Deviation	stddev	1	1	Period	stddev (0, M5, 10)
Standard Deviation Variable I	stddevvar	1	1		stddevvar (0, M5)

Internet EOD Data Manager Operation Guide

Intended Audience

Internet EOD Data Manager lets you manage EOD data downloaded from the Internet (i.e. when you create a chart with data source set to Internet EOD Data).

The Internet setting is also used if you use LiveCharts as a back fill service. In this case, the data management features are not used.

If you use other EOD data source (Quotes Plus, TC2005, files) or if you use a real-time data server, you do not need to know anything about Internet EOD Data Manager.

You should read the tutorial: ***Tutorial: EOD Data Speed Improvement*** (on page 209) if you have never use Internet EOD Data Manager.

Requesting Historical Data for Internet EOD Data

If you use other EOD data source (Quotes Plus, TC2005, files) or you use a real-time data server, this section is not applicable.

To request historical data:

1. Choose **Manager>Internet EOD Data** from main window to open Internet EOD Data Manager.
2. Press the **Historical Data** tab.
3. Make sure the symbol for the data you want to request is listed in the window. If not, press **Edit Symbols** button to add it.
4. Under **Request data for**, press one of the buttons **All checked**, **All checked at/below selected**, **Selected only**.
5. If you want to stop downloading data, press **Abort** button.

All checked

Data is requested for all symbols which are checked under the **C** column.

All checked at/below selected

Data is requested for all symbols which are checked under the **C** column and are listed at/below the current row.

Selected only

Data is requested for the single symbol at the current row.

Sending Symbol from Internet EOD Data Manager to Other Windows

If you use other EOD data source (Quotes Plus, TC2005, files) or you use a real-time data server, this section is not applicable.

To send symbol from Internet EOD Data Manager to other windows (e.g. chart):

1. Choose **Manager>Internet EOD Data** from main window to open Internet EOD Data Manager.
2. Press the **Historical Data** tab.
3. Double click on a symbol to send symbol to other windows, or you can press the Enter key to send symbol..

Sending symbol follows the chaining rule where Internet EOD Data Manager acts as the source of a chain. You can press the chain button to configure the intended destination. See ***Window Chaining*** (on page 596) for more information.

Symbols for Internet EOD Data Manager

If you use other EOD data source (Quotes Plus, TC2005, files) or you use a real-time data server, this section is not applicable.

To request historical data:

1. Choose **Manager>Internet EOD Data** from main window to open Internet EOD Data Manager.
2. Press the **Historical Data** tab.

3 Press **Edit Symbols** button to add/edit/delete symbols. When you edit, type one line per symbol.

Reset Update

If you have remove cache data files manually, the update status may not updated. If this is the case, you can press the **Reset Update** button. This button will check for presence of cache file. If the cache file is missing, the **Last Update** date will be reset to 1980/1/1. Resetting update allows Internet EOD Data Manager to retrieves long period of data rather than short data period and attempt to merge the data into the missing data file.

Internet EOD Data Connection Setting

If you use other EOD data source (Quotes Plus, TC2005, files) or you use a real-time data server, this section is not applicable.

NeoTicker® can load EOD data directly from the Internet. Whether you can download data successfully depends on many factors, including your connection, traffic on the Internet, etc. Generally it takes a few seconds for broadband connections (e.g. cable modem, ADSL and etc.) to download the data of a single stock. Commodity data download speed is similar.

> It is important for you to know that TickQuest does not guarantee the accuracy of the data NeoTicker® found on the Internet. You are responsible to ensure and verify the accuracy of such data.

For most users, Internet EOD Data simply work. No additional setup is required. However, if you encounter connection problem due to proxy server and firewall, you can set up the Internet connection by:

1 Choose **Manager>Internet EOD Data** from main window to open Internet EOD Data Manager.

2 In Internet EOD Data Manager, press the **Internet Options** tab.

3 Select one of **Direct Connection**, **Manual Proxy** or **Use IE Settings**. See the explanation for each of these options below.

Direction Connection

This is the default. Most Internet connections connect to the Internet through a direct connection. This option is the fastest if you can use it.

Manual Proxy

If you connect to the Internet through a proxy server, you will need to tell NeoTicker® before you can load data from the Internet successfully. Proxy servers are commonly used in large corporations and sometime with home connections.

First, you need to find out your proxy server address and port number. You can obtain these information from your web administrator, or by looking at the settings of your browser.

For Internet Explorer users:

1. Choose **Tools>Internet Options** in Internet Explorer. Explorer will open the **Internet Options** dialog.
2. Click on the **Connections** tab in the **Internet Options** dialog.
3. Press the **LAN Settings** button in the **Internet Options** dialog.
4. If you are using a proxy server, the **Use a proxy server** check box should be checked.
5. Copy the address and port settings to a piece of paper.
6. If you don't see an address and port settings, press the **Advance** button. A **Proxy Settings** dialog will be opened.
7. Copy the HTTP proxy address and port to a piece of paper.

For Netscape 4.X users:

1. Choose **Edit>Preference** in Netscape Communicator or Navigator
2. Open **Advanced** item under **Category**. Press the **Proxies** item.
3. If you use a manual proxy configuration, the **Manual proxy configuration** radio button should be checked.
4. Press the **View** button.
5. Copy the HTTP proxy address and port to a piece of paper.

Once you have the proxy information, you can fill these into manual proxy settings.

Use IE Settings

It is possible you cannot obtain a proxy address or port, for example if your web browser uses automatic proxy configuration or when there are some firewall issues.

If you can use Microsoft Internet Explorer for web browsing, you can use the **Use IE Settings** option. NeoTicker® will rely on Internet Explorer to retrieve data from the Internet. Data download speed will be slightly slower than **Direct Connection** and **Manual Proxy**, but using this option will allow NeoTicker® to pass through most firewalls.

Time Out

For direct and proxy connection, you can set a time out. If no data is received after a connection is made for the specified period of time, NeoTicker® will time out.

For fast Internet connections, you can set a lower time out.

Cleaning up Internet EOD Data

The maintenance routine described below removes all the Internet EOD data files created when you chart or pattern scan using **Internet EOD Data** as an EOD data source. This feature is useful when the Internet EOD data files are corrupted and are causing charting and scanning problems.

To clean up:

1 Backup the cache data in the `EODData` folder in the NeoTicker® installation.

2 Choose **Manager>Internet EOD Data** in the main window. Press the **Maintenance** tab.

3 Press the **Delete Cache** button.

Jurik Research Indicators

Jurik Research Indicators are separate products from NeoTicker®. You must purchase them separately. To purchase Jurik Research Indicators, you can visit their website at www.jurikresearch.com.

The following Jurik Research Indicators are supported: JMA, CFB, VEL, DMX, RSX

To use Jurik Research Indicators, you can simply add these indicators directly to a chart. These indicators are listed under the following names:

- Jurik Research JMA
- Jurik Research CFB
- Jurik Research VEL
- Jurik Research DMX
- Jurik Research RSX

Users of Obsolete Wrappers

It used to be that in order to use Jurik Research Indicators in NeoTicker®, you need to install a set of wrapper scripts. This is no longer necessary. The wrapper scripts are now considered to be obsolete.

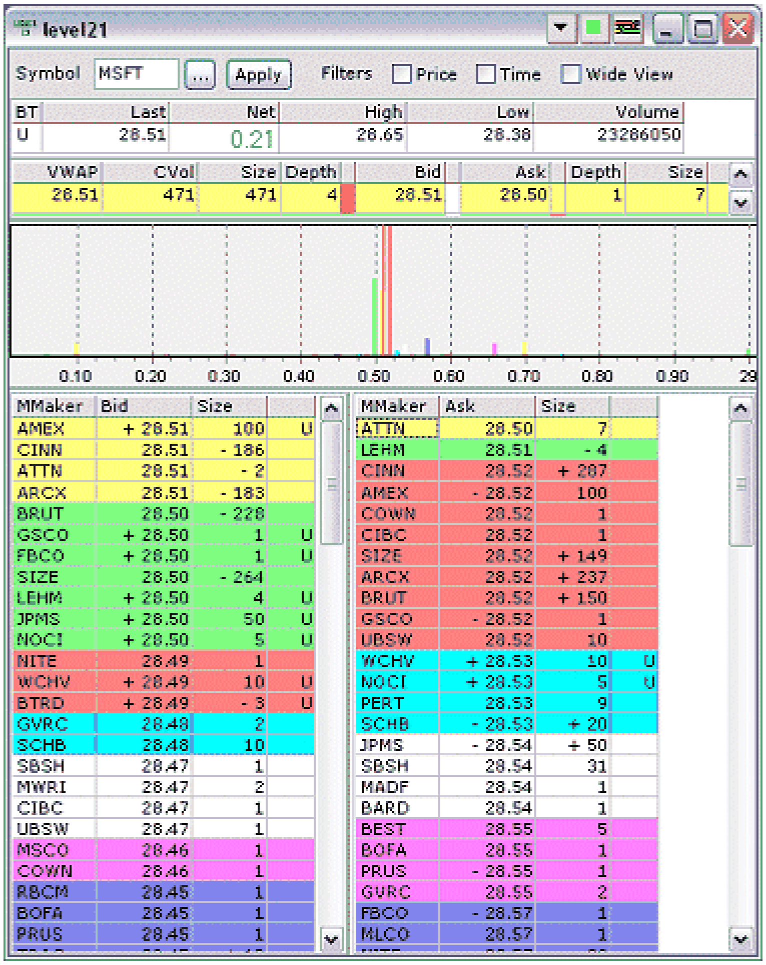
You should delete these wrapper scripts from your indicator directory (e.g. `C:\Program Files\TickQuest\NeoTicker\indicator`) and use the new way to directly call Jurik Research Indicators. Benefits:

- All Jurik Research Parameters are fully supported
- DMX support
- Runs Faster

Level 2 Window Operation Guide

Basic Operations

We will go over the items in the Level 2 window, from top to bottom.



Symbol MSFT

Filters □ Price □ Time □ Wide View

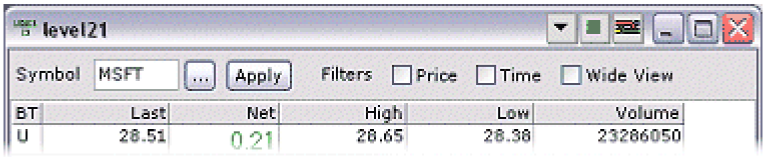
BT	Last	Net	High	Low	Volume
U	28.51	0.21	28.65	28.38	23286050

VWAP	CVol	Size	Depth	Bid	Ask	Depth	Size
28.51	471	471	4	28.51	28.50	1	7

MMaker	Bid	Size	
AMEX	+ 28.51	100	U
CINN	28.51	- 186	
ATTN	28.51	- 2	
ARCX	28.51	- 183	
BRUT	28.50	- 228	
GSCO	+ 28.50	1	U
FBCO	+ 28.50	1	U
SIZE	28.50	- 264	
LEHM	+ 28.50	4	U
JPMS	+ 28.50	50	U
NOCI	+ 28.50	5	U
NITE	28.49	1	
WCHV	+ 28.49	10	U
BTRD	+ 28.49	- 3	U
GVRC	28.48	2	
SCHB	28.48	10	
SBSH	28.47	1	
MWRI	28.47	2	
CIBC	28.47	1	
UBSW	28.47	1	
MSCO	28.46	1	
COWN	28.46	1	
RBCM	28.45	1	
BOFA	28.45	1	
PRUS	28.45	1	

MMaker	Ask	Size	
ATTN	28.50	7	
LEHM	28.51	- 4	
CINN	28.52	+ 287	
AMEX	- 28.52	100	
COWN	28.52	1	
CIBC	28.52	1	
SIZE	28.52	+ 149	
ARCX	28.52	+ 237	
BRUT	28.52	+ 150	
GSCO	- 28.52	1	
UBSW	28.52	10	
WCHV	+ 28.53	10	U
NOCI	+ 28.53	5	U
PERT	28.53	9	
SCHB	- 28.53	+ 20	
JPMS	- 28.54	+ 50	
SBSH	28.54	31	
MADF	28.54	1	
BARD	28.54	1	
BEST	28.55	5	
BOFA	28.55	1	
PRUS	- 28.55	1	
GVRC	28.55	2	
FBCO	- 28.57	1	
MLCO	28.57	1	

Top Panel



Symbol MSFT

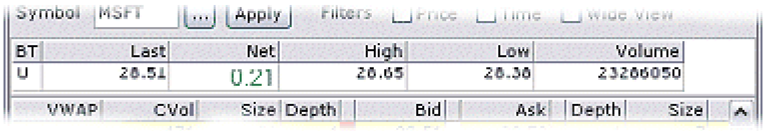
Filters □ Price □ Time □ Wide View

BT	Last	Net	High	Low	Volume
U	28.51	0.21	28.65	28.38	23286050

The top panel allows you to add symbol and controls the time and price filters.

At the top, there is a **Wide View** option. Toggling this option changes the zooming of the bid/ask chart.

Quote



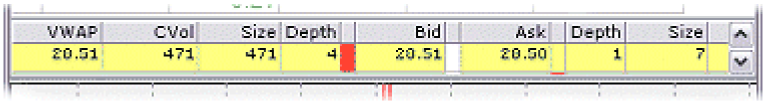
Symbol MSFT ... Apply Filters □ Price □ Time □ Wide View

BT	Last	Net	High	Low	Volume
U	28.51	0.21	28.65	28.38	23286050

VWAP	CVol	Size	Depth	Bid	Ask	Depth	Size

The Level 2 window displays the real-time quote of the stock, with the following fields: BT (Bid-tick), Last, Net, High, Low, Volume.

Bid/Ask Summary

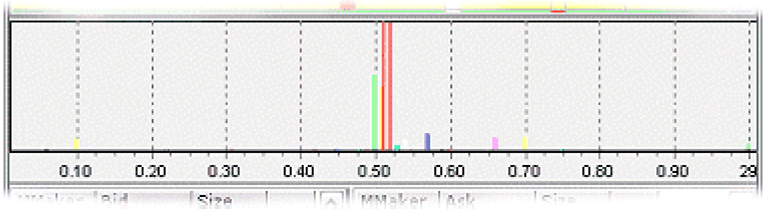


VWAP	CVol	Size	Depth	Bid	Ask	Depth	Size
20.51	471	471	4	20.51	20.50	1	7

Bid/ask summary summarizes the bid and ask information for all market makers for a particular bid/ask. Size is the total bid/ask size for all market makers for a particular bid/ask. Depth is the number of market makers for a particular bid/ask.

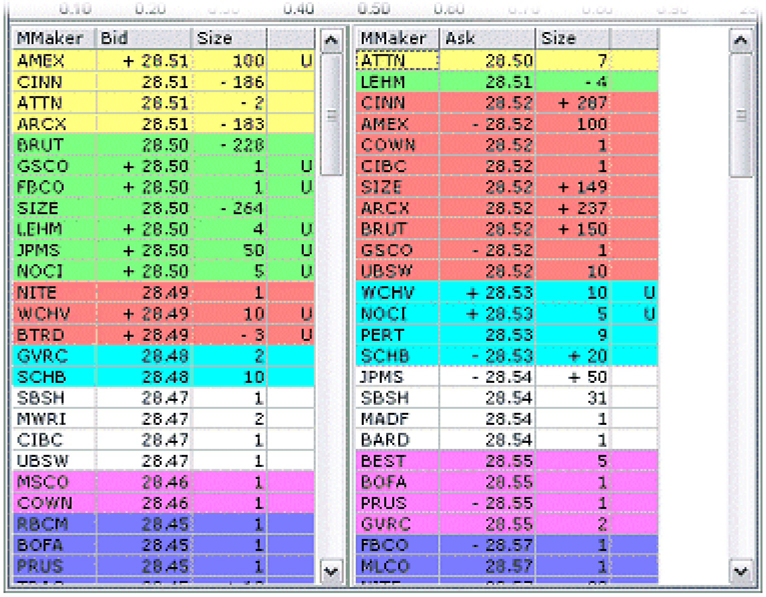
In the bid/ask summary, a red cell indicates the bid/ask is Nasdaq's official bid/ask price.

Bid/Ask Chart



Bid/ask chart displays the bid/ask size visually. The red line in the bid/ask chart is the price which the last trade happens.

Bid/Ask



MMaker	Bid	Size	
AMEX	+ 28.51	100	U
CINN	28.51	- 186	
ATTN	28.51	- 2	
ARCX	28.51	- 183	
BRUT	28.50	- 220	
GSCO	+ 28.50	1	U
FBCO	+ 28.50	1	U
SIZE	28.50	- 264	
LEHM	+ 28.50	4	U
JPMS	+ 28.50	50	U
NOCI	+ 28.50	5	U
NITE	28.49	1	
WCHV	+ 28.49	10	U
BTRD	+ 28.49	- 3	U
GVRC	28.48	2	
SCHB	28.48	10	
SBSH	28.47	1	
MWRI	28.47	2	
CIBC	28.47	1	
UBSW	28.47	1	
MSCO	28.46	1	
COWN	28.46	1	
RBCM	28.45	1	
BOFA	28.45	1	
PRUS	28.45	1	

MMaker	Ask	Size	
ATTN	28.50	7	
LEHM	28.51	- 4	
CINN	28.52	+ 287	
AMEX	- 28.52	100	
COWN	28.52	1	
CIBC	28.52	1	
SIZE	28.52	+ 149	
ARCX	28.52	+ 237	
BRUT	28.52	+ 150	
GSCO	- 28.52	1	
UBSW	28.52	10	
WCHV	+ 28.53	10	U
NOCI	+ 28.53	5	U
PERT	28.53	9	
SCHB	- 28.53	+ 20	
JPMS	- 28.54	+ 50	
SBSH	28.54	31	
MADF	28.54	1	
BARD	28.54	1	
BEST	28.55	5	
BOFA	28.55	1	
PRUS	- 28.55	1	
GVRC	28.55	2	
FBCO	- 28.57	1	
MLCO	28.57	1	

At the bottom is the bids and asks for all market makers. The bids and asks are sorted by the price. The Level 2 window displays all bid/ask price reported by your data vendor.

See ***Level 2 Window, Bid Ask Table*** (see "Bid Ask Table" on page 621) for explanation of the columns and configuration options.

CME/CBT Depth Data

See ***Level 2 Window, CME/CBT Depth Data*** (see "CME/CBT Depth Data" on page 622).

Adjusting Time and Price Filters

You can set the criteria for time and price filtering by:

1. Right clicking on the Level 2 window to open the pop up menu.
2. Choose **Setup.**
3. Press the **Filter** tab.
4. Make adjustments.
5. Press the **Apply** button.

The price filter allows you to filter by percentage and by value.

Adjusting Depth Chart

You can adjust the settings of Level 2 windows' depth chart by:

1 Right clicking on the Level 2 window to open the pop up menu.

2 Choose **Setup.**

3 Press the **Chart** tab.

4 Make adjustments.

5 Press the **Apply** button.

You can make the following adjustments:

- Center of the chart
- Narrow view settings, by percentage or value
- Wide view settings, by percentage or value

Depth Table Partition

The depth table can be partition by exact price or by a fixed increment. To set up:

1 Right clicking on the Level 2 window to open the pop up menu.

2 Choose **Setup.**

3 Press the **Depth** tab.

4 Make adjustments.

5 Press the **Apply** button.

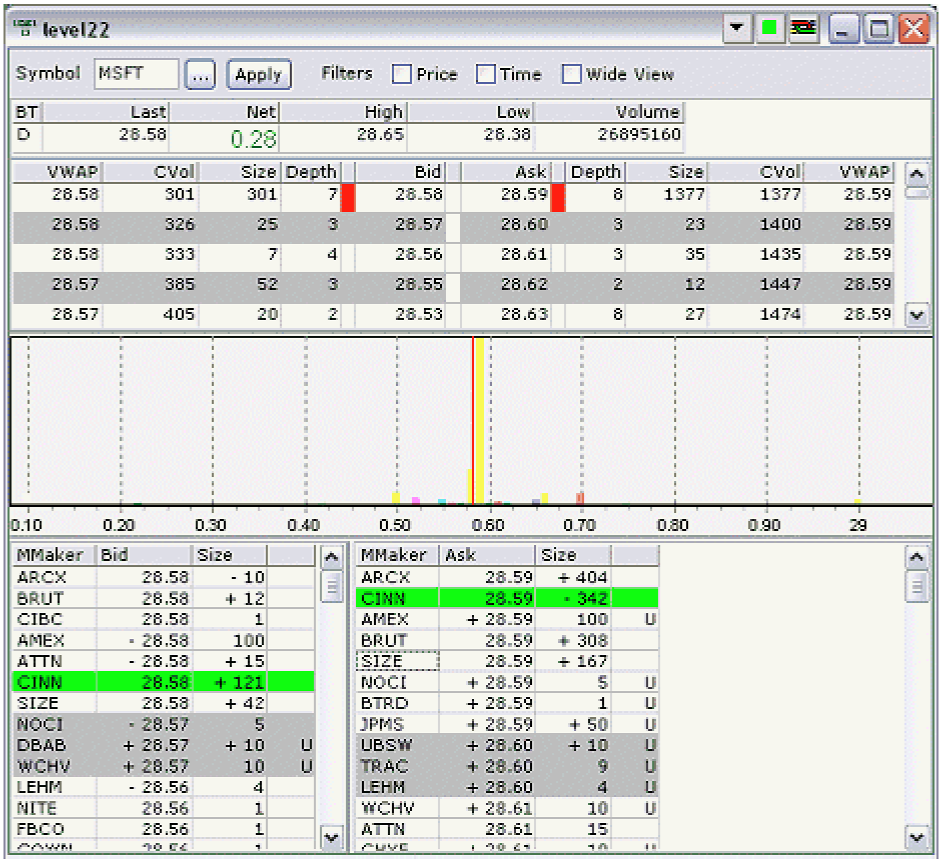
Tracking a Market Maker

You can track a market maker by marking their bid/ask with a special color.

Level 2 window is quite colorful, so color marking can be difficult to read. So the first thing we do is to make the Level 2 window plainer.

Right click to open the pop up menu, and choose **Color Style>2 Colors**. This reduces the color use in the Level 2 window to white and gray color except in the depth chart.

Then click on the name of the market maker you want to track under **MMaker** in the bid ask list. The market maker will be shown in a special color (light green by default).



Symbol MSFT

Filters □ Price □ Time □ Wide View

BT	Last	Net	High	Low	Volume
D	28.58	0.28	28.65	28.38	26895160

VWAP	CVol	Size	Depth	Bid	Ask	Depth	Size	CVol	VWAP
28.58	301	301	7	28.58	28.59	8	1377	1377	28.59
28.58	326	25	3	28.57	28.60	3	23	1400	28.59
28.58	333	7	4	28.56	28.61	3	35	1435	28.59
28.57	385	52	3	28.55	28.62	2	12	1447	28.59
28.57	405	20	2	28.53	28.63	8	27	1474	28.59

MMaker	Bid	Size	
ARCX	28.58	- 10	
BRUT	28.58	+ 12	
CIBC	28.58	1	
AMEX	- 28.58	100	
ATTN	- 28.58	+ 15	
CINN	28.58	+ 121	
SIZE	28.58	+ 42	
NOCI	- 28.57	5	
DBAB	+ 28.57	+ 10	U
WCHV	+ 28.57	10	U
LEHM	- 28.56	4	
NITE	28.56	1	
FBCO	28.56	1	

MMaker	Ask	Size	
ARCX	28.59	+ 404	
CINN	28.59	- 342	
AMEX	+ 28.59	100	U
BRUT	28.59	+ 308	
SIZE	28.59	+ 167	
NOCI	+ 28.59	5	U
BTRD	+ 28.59	1	U
JPMS	+ 28.59	+ 50	U
UBSW	+ 28.60	+ 10	U
TRAC	+ 28.60	9	U
LEHM	+ 28.60	4	U
WCHV	+ 28.61	10	U
ATTN	28.61	15	

To un-track, press on the name again.

Assigning Color to Market Makers

You can assign custom color to market makers. So you can watch the activities of the market makers that interest you.

1 Right clicking on the Level 2 window to open the pop up menu.

2 Choose **Setup.**

3 Press the **MM Colors** tab.

4 Choose **Customize** and assign colors to market makers.

5 Press the **Apply** button.

Color and Font Settings

To adjust color and font settings:

1 Right clicking on the Level 2 window to open the pop up menu.

2 Choose **Setup.**

3 Press the **Visual** tab.

4 Make color and font settings.

5 Press the **Apply** button.

Adjusting Level 2 Window Speed

Because Level 2 data are bid and ask data, the amount of data is huge, putting a heavy load on your computer's CPU. You can adjust the Level 2 window speed for a comfortable level for your computer.

1 Right clicking on the Level 2 window to open the pop up menu.

2 Choose **Setup.**

3 Press the **Speed** tab.

4 Adjust speed.

5 Press the **Apply** button.

Most Pentium 4 class computer can handle the fastest Level 2 window speed.

Quick Setup

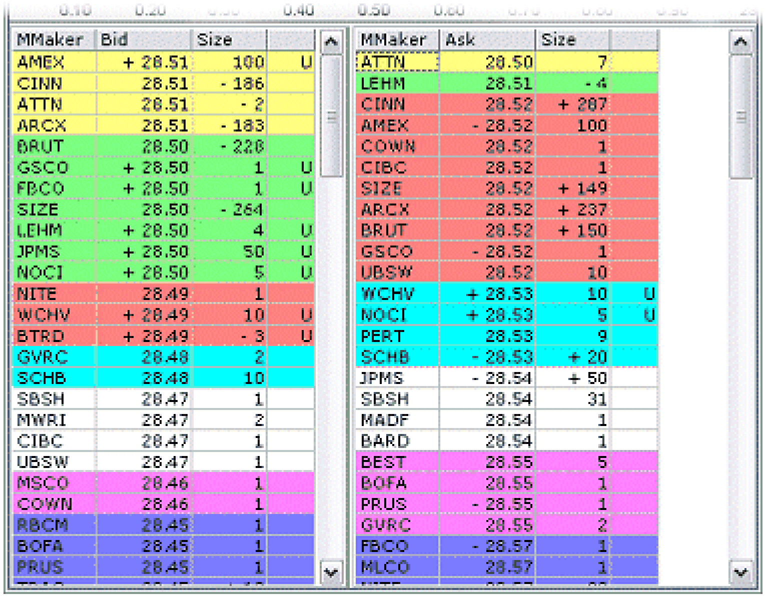
Quick Setup provides settings to let you quickly configure Level 2 Window. To access Quick Setup, right click on pop up menu, and choose one of the items under **Quick Setup**.

While the Quick Setups are designed with specific exchange in mind, the setup works with all exchanges, e.g. you can use CBT Depth Quick Setup to display Level 2 data.

Quick Setup	Description
NASD Level 2	Default Level 2 setup.
CME Depth	Setup designed for CME Depth data with an **Orders** column to display number of orders information.
CBT Depth	Similar to CME Depth but with **Orders** column hidden.

For more information on CME/CBT Depth, see ***Level 2 Window, CME/CBT Depth Data*** (see "CME/CBT Depth Data" on page 622).

Bid Ask Table



MMaker	Bid	Size	
AMEX	+ 28.51	100	U
CINN	28.51	- 186	
ATTN	28.51	- 2	
ARCX	28.51	- 183	
BRUT	28.50	- 220	
GSCO	+ 28.50	1	U
FBCO	+ 28.50	1	U
SIZE	28.50	- 264	
LEHM	+ 28.50	4	U
JPMS	+ 28.50	50	U
NOCI	+ 28.50	5	U
NITE	28.49	1	
WCHV	+ 28.49	10	U
BTRD	+ 28.49	- 3	U
GVRC	28.48	2	
SCHB	28.48	10	
SBSH	28.47	1	
MWRI	28.47	2	
CIBC	28.47	1	
UBSW	28.47	1	
MSCO	28.46	1	
COWN	28.46	1	
RBCM	28.45	1	
BOFA	28.45	1	
PRUS	28.45	1	

MMaker	Ask	Size	
ATTN	28.50	7	
LEHM	28.51	- 4	
CINN	28.52	+ 287	
AMEX	- 28.52	100	
COWN	28.52	1	
CIBC	28.52	1	
SIZE	28.52	+ 149	
ARCX	28.52	+ 237	
BRUT	28.52	+ 150	
GSCO	- 28.52	1	
UBSW	28.52	10	
WCHV	+ 28.53	10	U
NOCI	+ 28.53	5	U
PERT	28.53	9	
SCHB	- 28.53	+ 20	
JPMS	- 28.54	+ 50	
SBSH	28.54	31	
MADF	28.54	1	
BARD	28.54	1	
BEST	28.55	5	
BOFA	28.55	1	
PRUS	- 28.55	1	
GVRC	28.55	2	
FBCO	- 28.57	1	
MLCO	28.57	1	

At the bottom of Level 2 Window is bid ask information.

The following table provides a summary of the data displayed under each column:

Column	Data Displayed
MMaker	Market Maker (Level 2 only).
Bid/Ask	Bid and ask price. +/- sign indicates if price has increase/decrease from previous price.
Size	Bid and ask size. +/- sign indicates if the bid/ask size has increase/decrease from previous size.
Orders	Number of orders (CME Depth only). +/- sign indicates if the number of orders has increase/decrease from previous value.
Status	Exchange broadcasted status for the market maker (Level 2 only).

Configuring Bid/Ask Table

Right click on Level 2 Window to open pop up menu. You can have the following options to configure Bid/Ask Table:

- Select **Quick Setup** to display the default for the type of data you are displaying. See ***Level 2 Window, Quick Setup*** (see "Quick Setup" on page 621).
- Select **Bid / Ask Style** to display one of the pre-defined column arrangement.
- Select **Bid / Ask ScrollBar** for different scroll bar options
- Select **Bid / Ask Columns** to toggle the visibility of the columns.

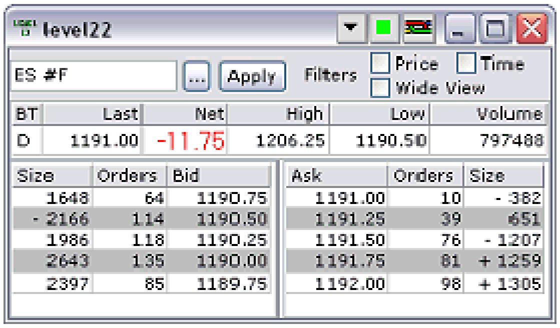
CME/CBT Depth Data

CME/CBT Depth Data requires additional subscription service. Contact your data vendor for availability and pricing for this type of data service

Level 2 Window can be configured to display CME/CBT depth data for contracts such as ES, NQ, ZB and YM. To configure:

1. Enter the symbol in Level 2 Window. Press **Apply** button.
2. Right click on Level 2 Window to open pop up menu, and choose **Quick Setup>CME Depth** or **Quick Setup>CBT Depth**.

Figures below are examples of CME/CBT Depth data.



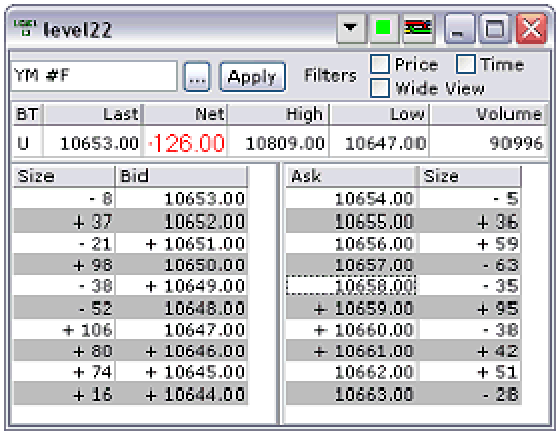
ES #F

Filters ☐ Price ☐ Time ☐ Wide View

BT	Last	Net	High	Low	Volume
D	1191.00	-11.75	1206.25	1190.50	797488

Size	Orders	Bid
1648	64	1190.75
- 2166	114	1190.50
1986	118	1190.25
2643	135	1190.00
2397	85	1189.75

Ask	Orders	Size
1191.00	10	- 382
1191.25	39	651
1191.50	76	- 1207
1191.75	81	+ 1259
1192.00	98	+ 1305



YM #F

Filters ☐ Price ☐ Time ☐ Wide View

BT	Last	Net	High	Low	Volume
U	10653.00	-126.00	10809.00	10647.00	90996

Size	Bid
- 8	10653.00
+ 37	10652.00
- 21	+ 10651.00
+ 98	10650.00
- 38	+ 10649.00
- 52	10648.00
+ 106	10647.00
+ 80	+ 10646.00
+ 74	+ 10645.00
+ 16	+ 10644.00

Ask	Size
10654.00	- 5
10655.00	+ 36
10656.00	+ 59
10657.00	- 63
10658.00	- 35
+ 10659.00	+ 95
+ 10660.00	- 38
+ 10661.00	+ 42
10662.00	+ 51
10663.00	- 28

Availability

CME/CBT Depth Data is a relatively new type of data service. Not all data vendors provide the appropriate tools for accessing this type of data. Currently NeoTicker® can display CME/CBT depth data from the following data vendor(s):

- eSignal
- MB Trading as data feed

Locales

Users

When you type a number or date time (e.g. as indicator parameter), you should use the format determined by the regional setting of your computer. Typically, regional setting is already set for you by the computer manufacturer. To determine your regional settings, in Windows, open Control Panel, double click on Regional and Language Options.

Examples below show numbers and date time if your computer's regional setting is set to English (United States):

```
13.50

100.2

12/25/2005

4:15:00 PM
```

Examples below show numbers and date time if your computer's regional setting is set to English (United Kingdom):

```
13.50

100.2

25/12/2005

16:15:00
```

Examples below show numbers and date time if your computer's regional setting is set to is set to German (Germany):

```
13,60

100,2

25.12.2005

16:50:00
```

Formula Writers/Programmers

Formulas and scripts uses a NeoTicker® specific format. This format is independent from your computer's regional settings. No matter what the computer's regional setting is, when you write formulas and scripts, you need to use the format specified here. Note also that when you use Script Editor to create indicator, the default parameter settings also use the format specified here.

Below are examples of specifying numbers and date time in formula and scripts.

```
13.50

100.2

2005/12/25

16:15:00

2005/12/25 16:15:00
```

Here is exact format:

Type	Format
Float number	Identical to numbers in English (United States), using period as decimal
Date	yyyy/mm/dd
Time	hh:mm:ss
Date Time	yyyy/mm/dd hh:mm:ss

Note that when you handle user parameters, the user of the indicator can specify numbers and date time in their native locale and the values will be translated automatically. For example, if a German user type 13,50 for a parameter when using your indicator, in your script `Param1.str` will appear as 13.50.

Functions in NTLib

The string conversion functions in NTLib are locale dependent. For example, date2str will translate the date value December 25, 2005 to 2005/12/25 if your computer's regional setting is English (United States).

Main Window Operation Guide

Main window is the window you see first when you open NeoTicker®.

Caption and Status

Main window's caption displays the status of the data server:

Status	Meaning
Ready	The data server is running and is receiving data.
No data for N minutes	There has been no tick coming in for N minutes. The server is probably off-line or there is a problem with the connection to the data vendor.
Offline Mode	NeoTicker® is running in off-line mode.
Offline	NeoTicker® is disconnected from the real time server
xxx.quote.com, Ping Time xxx secs	Server connected and ping time for Quote.com users
Server: Ready(DHTLN), Reception: OK, state OK	eSignal server status. DHTLN are the data server, historical server, tick server, Level 2 server and news server respectively
Server : K3-1, XXX msg/sec	Server connected and message per second for myTrack users.

Even if the data server is off-line, you can still use NeoTicker®. However, there will be no ticks coming into NeoTicker®. So the function windows will not update in real-time.

Shortcuts

Key	Function
CTRL-SPACE	Open the main window menu in the current monitor. Useful if you want to hide main window or work with many monitors.
F5	Brings the main window to front. If main window is the front most window, sends it to the back.
F8	Opens the replace symbol window
PAGE UP/PAGE DOWN	Switch between groups
CTRL-SHIFT-N	For Quote.com only, switches to next server
SHIFT	Aborts a connection attempt to an Internet data feed

Tool Buttons

Tool buttons let you invoke a NeoTicker® function quickly. Some tool button has its own drop down menu which can be opened by clicking on the triangle besides the tool button.

Tool buttons on the main window is user configurable. For more information, see ***Customizable Main Window Tool Bar*** (on page 498).

NeoBreadth® Operation Guide

Overview

History

NeoBreadth® is a feature based on TickQuest's research into market breadth since the late 1990's. Market breadth are widely recognized as useful trading tools. What TickQuest had done was systematizing the creation and calculation of market breadth. We paid particular attention to historical patterns to find ways to quantify performance.

Originally, NeoBreadth® was sold as a separate product from NeoTicker®. NeoBreadth® was an end-of-day product as personal computers at that time weren't capable to handle real-time breadth calculations. In 2003, we recognized that personal computers have advanced enough for real-time breadth calculations was possible. So we started integrating the two products.

Now NeoBreadth® is part of NeoTicker® that is capable of generating real-time breadth symbols.

Anatomy of a Breadth Symbol

Market breadth is a statistics of multiple instruments. For example, Nasdaq 100 Advanced Issues is a market breadth.

In NeoTicker®, market breadth are implemented as breadth symbols. Each breadth symbol can be used where you expect you can use a regular symbol. You can chart, quote, or scan breadth symbols.

There are 2 parts of a breadth symbol.

- Breadth definition (e.g. Advanced Issues, Tick16)
- Symbol list (Nasdaq 100, Dow Jones 30)

Breadth definition can be applied on any symbol list. For example, advanced issues definition can be applied the same way on Nasdaq 100 and on Dow Jones 30. By applying breadth definition on different symbol lists, you create different breadth symbols, e.g. Nasdaq 100 Advanced Issues, Dow Jones 30 Advanced Issues.

Breadth Definition

Breadth definition defines the statistics.

First, you need to define what to collect. For example, if you want to collect advanced issues, you need to define a formula like:

```
last > prev
```

You may notice this formula is a quote window formula. In general you can access any quote window fields, as well as using indicators. For example, you can use:

```
RSIndex(0,M1,10) < 30
```

to collect statistics about stocks which has an RSI value less than 30.

This formula is applied on each symbol in your symbol list. So you will have a collection of data for each symbol. The next step is to specify a way to summarize the data. NeoBreadth® provides you the following method for summarizing data: Sum, Avg, Min, Max.

For example, if you apply Sum to the RSI formula above, the breadth symbol created will collect statistics on the number of issues with an RSI value less than 30. This is a breadth symbol that can be used as a broad market oversold indicator.

Once a breadth symbol is defined, NeoTicker® treats it as a regular symbol. Data is collected so historical data will be available. You can simply access the breadth indicator as a regular symbol, e.g. you can chart it, quote it, scan it.

In this aspect, breadth symbols are similar to user defined symbols.

Generating Historical Data

NeoTicker® collects data for breadth symbols while it is running. So historical data for breadth symbols is available. However, it is not possible to directly recreate historical data because breadth symbol is created using quote window formula. As no data vendor provides historical data on all quote fields, recreating history is not generally possible.

NeoBreadth® provides a way to regenerate historical data using indicator formula (e.g. the type of formula used in fml, fml2, etc). This type of formula has access to OHLC data, but does not have access to the quote fields.

Therefore it is possible regenerate historical data only when the breadth symbol is based on OHLC data and the derived indicators. If the breadth symbols is based on other quote fields, regenerating history is not possible.

Batch Mode

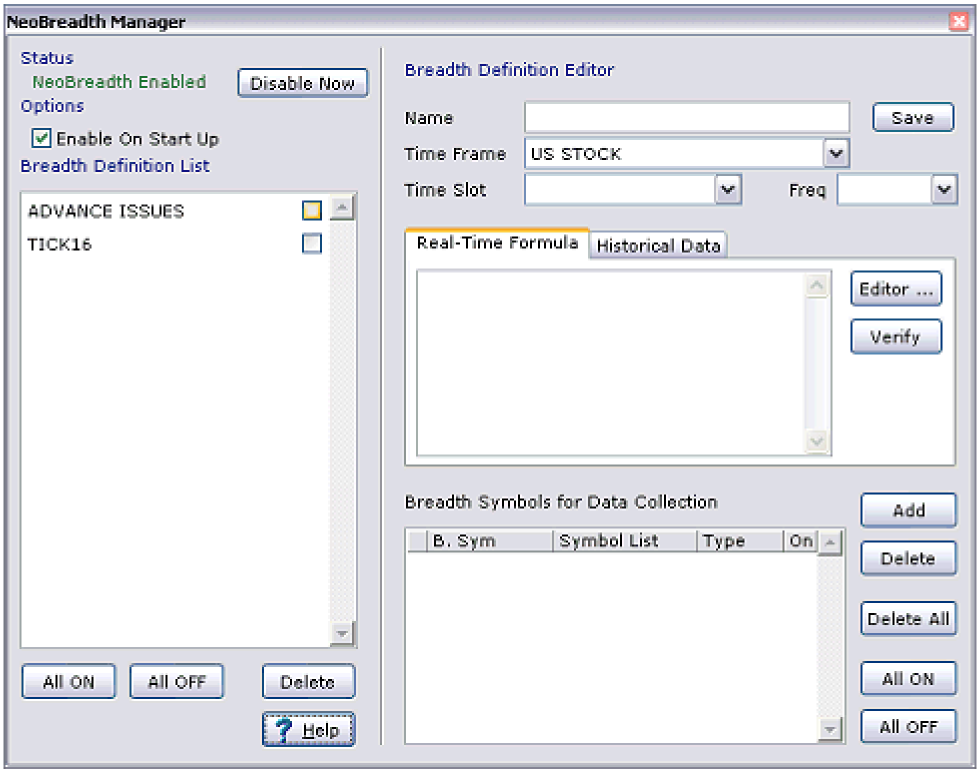
When you are defining a breadth, you can check the **Batch Gen** box. When you press the **Batch Gen** button in History Generator, it will recognize all the breadth definitions with **Batch Gen** enabled and generate history. Batch mode allows you to download historical data at night, then batch generate many batch symbols with a single click. See ***NeoBreadth, Reference*** (see "Reference" on page 642).

Tutorial: Creating an RSI Oversold Breadth Indicator

We will create a breadth symbol that reports the number of stocks that matches an RSI oversold condition out of a symbol list,

Opening NeoBreadth® Manager

First open NeoBreadth® Manager by choosing **Manager>NeoBreadth** from the main window.



NeoBreadth Manager

Status
NeoBreadth Enabled

Options
☑ Enable On Start Up

Breadth Definition List

ADVANCE ISSUES	□
TICK16	□

Breadth Definition Editor

Name:
Time Frame: US STOCK
Time Slot:
Freq:

B. Sym	Symbol List	Type	On

By default, NeoBreadth® is enabled when NeoTicker® is connected to a data feed. You can check the status in the upper left hand corner of NeoBreadth® Manager to verify.

If status is disabled, verify that NeoTicker® is connecting to a real-time data feed and press the **Enable Now** button.

Breadth Definition

We will be creating a breadth definition called "RSI Oversold".

Make the following changes in the manager:

Name - RSI Oversold

Time Frame - US STOCK

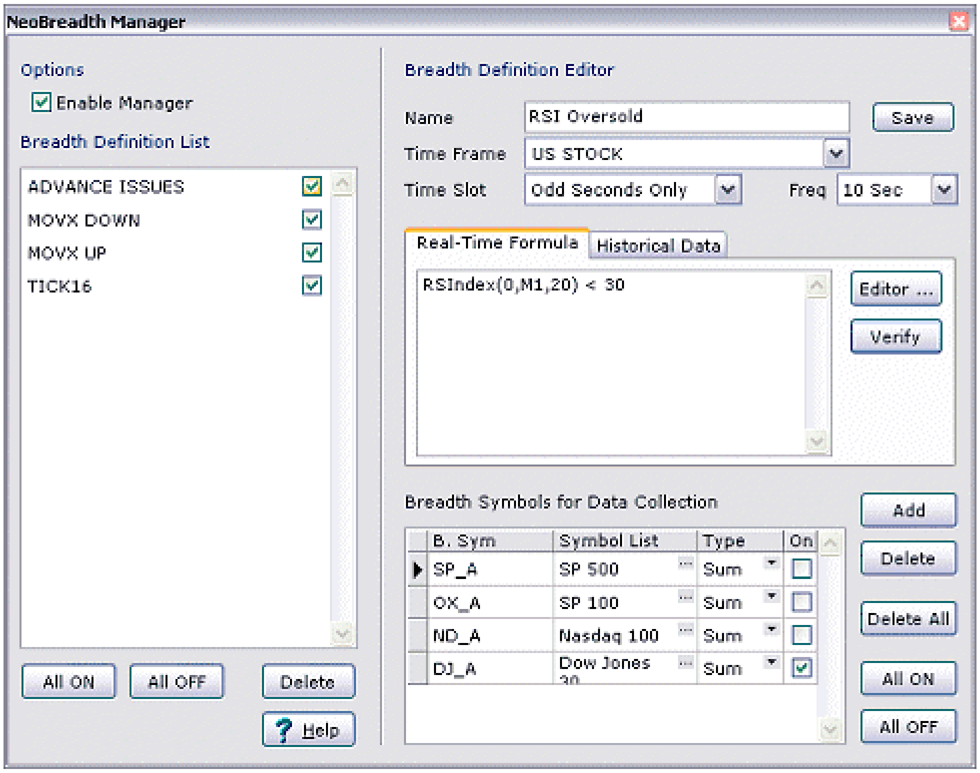
Time Slot - Odd Seconds Only

Freq - 10 Sec

Press the **Real-Time Formula** tab, and enter the formula for RSI oversold condition.

```
RSIndex(0,M1,20) < 30
```

Press the **Verify** button to make sure the formula is correct.



NeoBreadth Manager

Options

☑ Enable Manager

Breadth Definition List

ADVANCE ISSUES	☑
MOVX DOWN	☑
MOVX UP	☑
TICK16	☑

Breadth Definition Editor

Name: RSI Oversold

Time Frame: US STOCK

Time Slot: Odd Seconds Only

Freq: 10 Sec

Real-Time Formula

```
RSIndex(0,M1,20) < 30
```

Breadth Symbols for Data Collection

B. Sym	Symbol List	Type	On
SP_A	SP 500	Sum	□
OX_A	SP 100	Sum	□
ND_A	Nasdaq 100	Sum	□
DJ_A	Dow Jones 30	Sum	☑

Breadth Symbol

So far you have define a breadth definition. To use the breadth definition, you will need to apply the breadth definition to a symbol list to create a breadth symbol.

You can do this in the grid under **Breadth Symbols for Data Collection**.

Breadth Symbols for Data Collection may contains some breadth symbols from other definitions. Press the **Delete All** button to clear them.

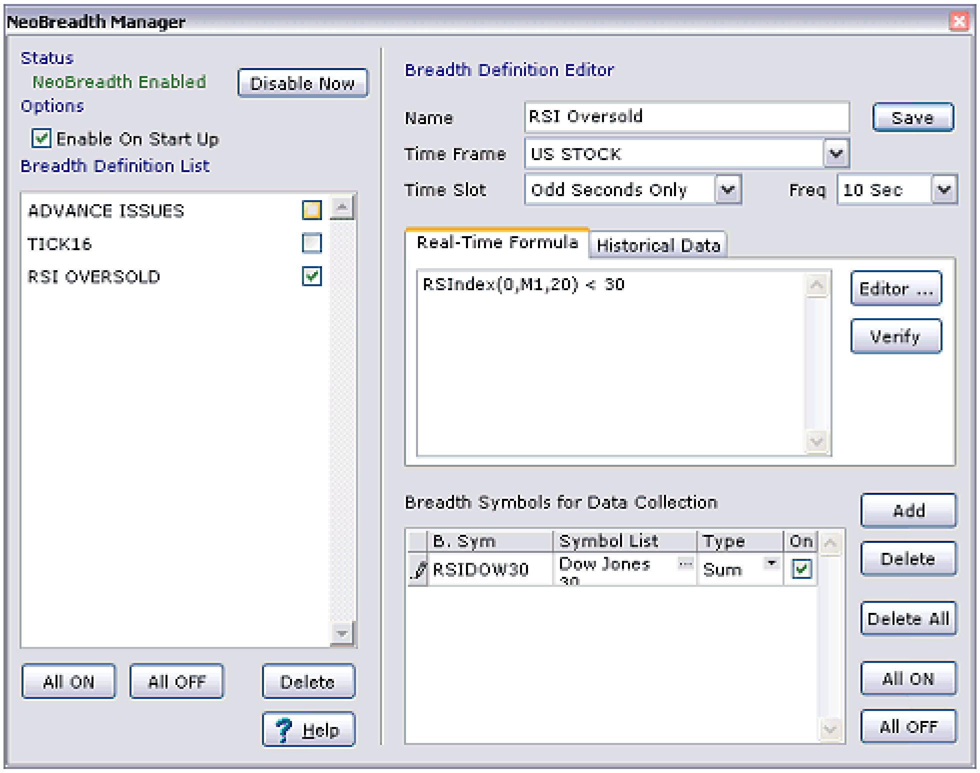
Press the **Add** button. A new breadth symbol is created.

Provide a name under **B. Sym**. We will call this symbol RSIDOW30. Select the Dow Jones 30 symbol list under **Symbol List**. Set **Type** to Sum. Make sure **On** is checked.

Press the **Save** button. A dialog will open to ask you if you want to collect data now. Press the **Yes** button.

When you choose the name for a breadth symbol, make sure it does not conflict with the name of a real symbol (e.g. MSFT, INTC). Conflicting symbol has undefined behavior. You can use special symbols as prefix to help distinguish between the two types of symbols, e.g. $$MSFT is a safe breadth symbol name.

The figure below shows NeoBreadth® manager with the RSIDOW30 breadth symbol defined.



☑ Enable On Start Up

ADVANCE ISSUES □
TICK16 □
RSI OVERSOLD ☑

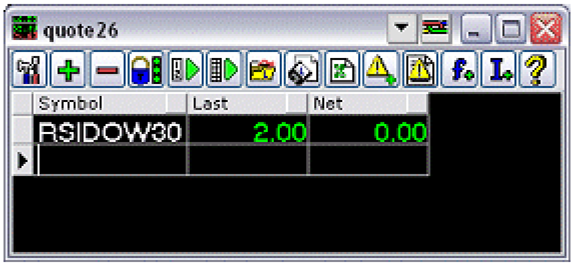
Name: RSI Oversold
Time Frame: US STOCK
Time Slot: Odd Seconds Only
Freq: 10 Sec

```
RSIndex(0,M1,20) < 30
```

Breadth Symbols for Data Collection

B. Sym	Symbol List	Type	On
RSIDOW30	Dow Jones 30	Sum	☑

Now open a quote window and enter a RSIDOW30 as a symbol. You will see the breadth symbol appear. It may take a couple of minutes for RSIDOW30 to stabilize because NeoTicker® needs to retrieve historical data for 30 stocks and apply indicators on them.



Symbol	Last	Net
RSIDOW30	2.00	0.00

Note that if you open a chart for RSIDOW30 now, the chart will not have much historical data until more data has been accumulated in real-time.

The steps below require you to have a symbol limit of 130 or higher. If you have a low symbol limit, skip to the next section.

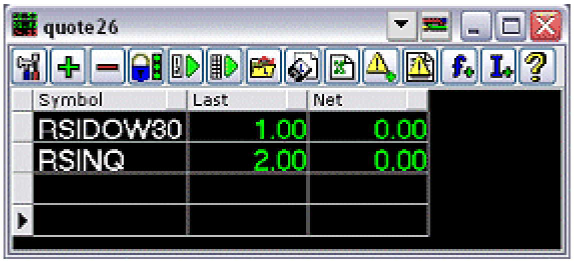
Going back to NeoBreadth® Manager. Press **Add** button to add another symbol.

Provide a name under **B. Sym**. We will call this symbol RSINQ. Select the Nasdaq 100 symbol list under **Symbol List**. Set **Type** to Sum. Make sure **On** is checked.

Press the **Save** button. A dialog will open to ask you if you want to collect data now. Press the **Yes** button.

You have created another breadth symbol based on the same breadth definition, but with a different symbol list.

Go back to the quote window, you can quote RSINQ now.



Symbol	Last	Net
RSIDOW30	1.00	0.00
RSINQ	2.00	0.00

Regenerating Historical Data

So far you have defined breadth symbols for real-time use. What if you need historical breadth data?

The steps below will show you how to regenerate historical data for the breadth symbols so you can chart breadth symbols right away.

In NeoBreadth® Manager, press the **Historical Data** tab.

Enter the following formula:

```
RSIndex(0,data1,20) < 30
```

Then set the parameters to:

Bar Type - Min

Bar Size - 1

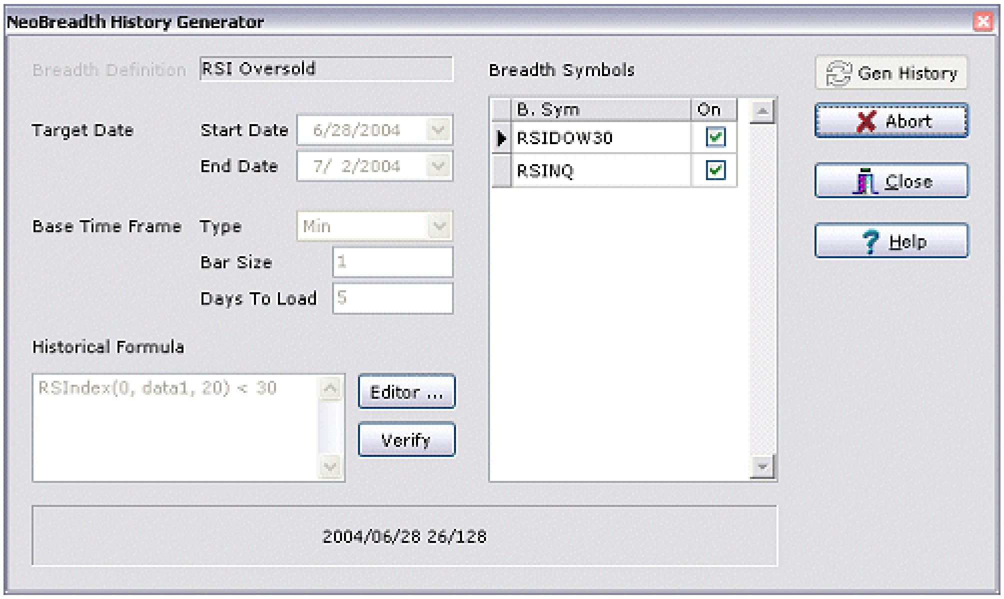
Days to Load - 2

Note that we are using indicator formula syntax here (the type of formula used in fml, fml2, etc). What we are doing is specifying the same RSI indicator, with the same time frame as the real-time quote formula.

Press the **Generate History** button to open NeoBreadth® History Generator.

Verify that the time frame and formula are correct. Specify a target date for generating history. For our example, make sure the **End Date** is today, and set the **Start Date** to a couple of days ago.

Press the **Gen History** button.



Breadth Definition: RSI Oversold

Target Date — Start Date: 6/28/2004; End Date: 7/ 2/2004

Base Time Frame — Type: Min; Bar Size: 1; Days To Load: 5

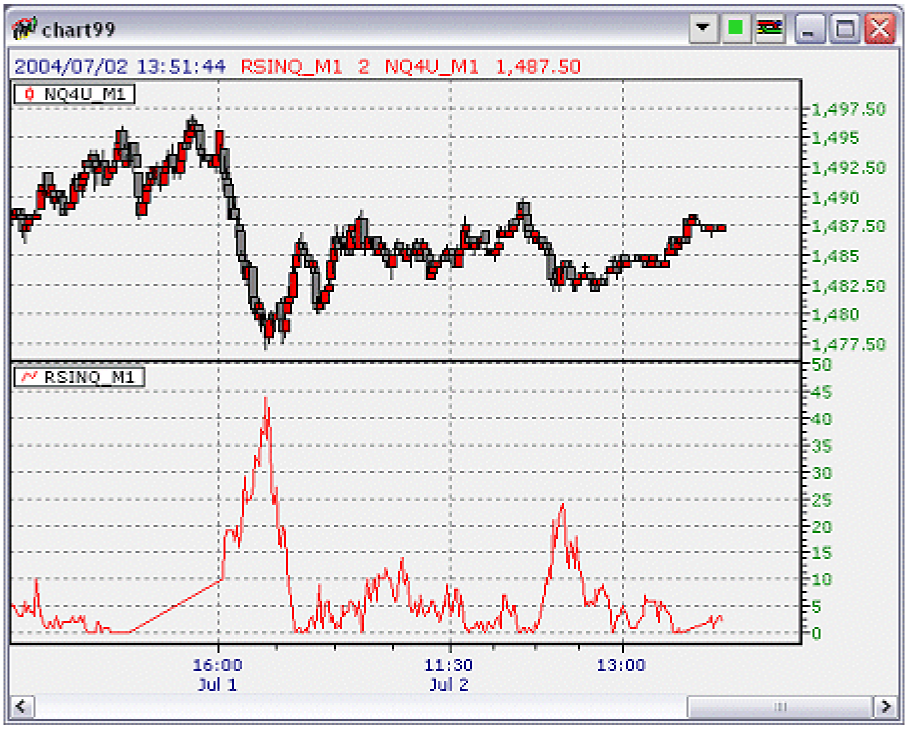
Historical Formula: RSIndex(0, data1, 20) < 30

Breadth Symbols

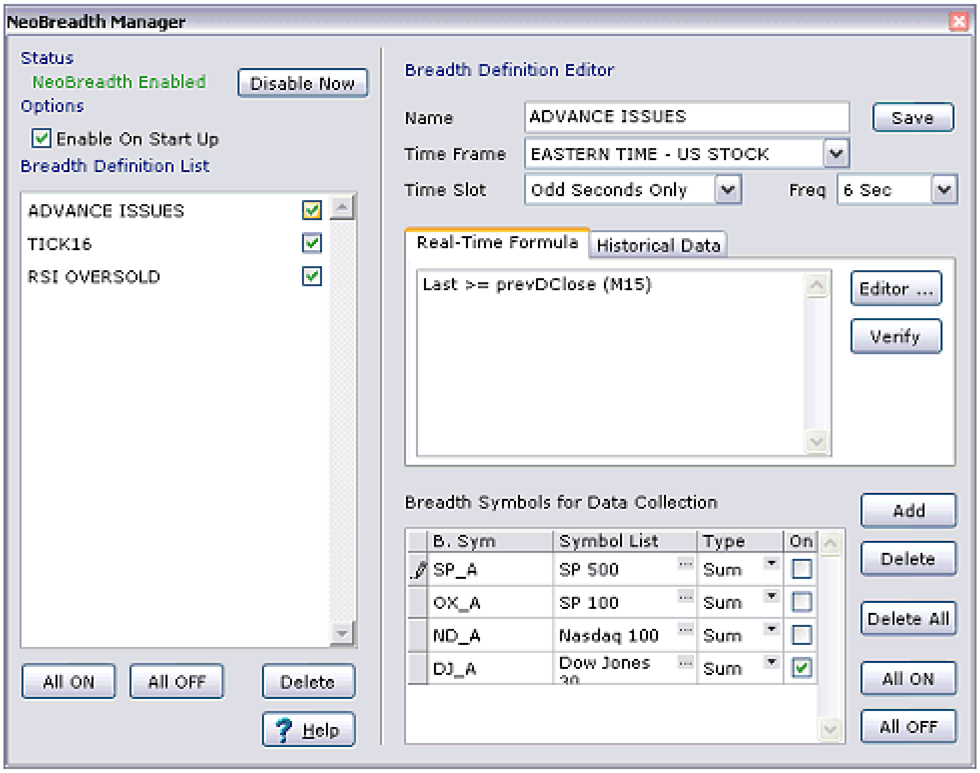
B. Sym	On
RSIDOW30	☑
RSINQ	☑

2004/06/28 26/128

Open a chart and chart RSIDOW30 and RSINQ. You can see that historical data is now available.



Example: Advanced Issues



☑ Enable On Start Up

ADVANCE ISSUES	☑
TICK16	☑
RSI OVERSOLD	☑

Name: ADVANCE ISSUES
Time Frame: EASTERN TIME - US STOCK
Time Slot: Odd Seconds Only
Freq: 6 Sec

```
Last >= prevDClose (M15)
```

B. Sym	Symbol List	Type	On
SP_A	SP 500	Sum	☐
OX_A	SP 100	Sum	☐
ND_A	Nasdaq 100	Sum	☐
DJ_A	Dow Jones 30	Sum	☑

Advance Issues is an example breadth definition. It calculates advancing issues of a basket. To enable Advance Issues, check the box besides ADVANCE ISSUES in NeoBreadth® Manager. By default, only the Advance Issues for Dow 30 is collected. If you have a high symbol limit, you can enable the other Advance Issues breadth symbols.

Notes:

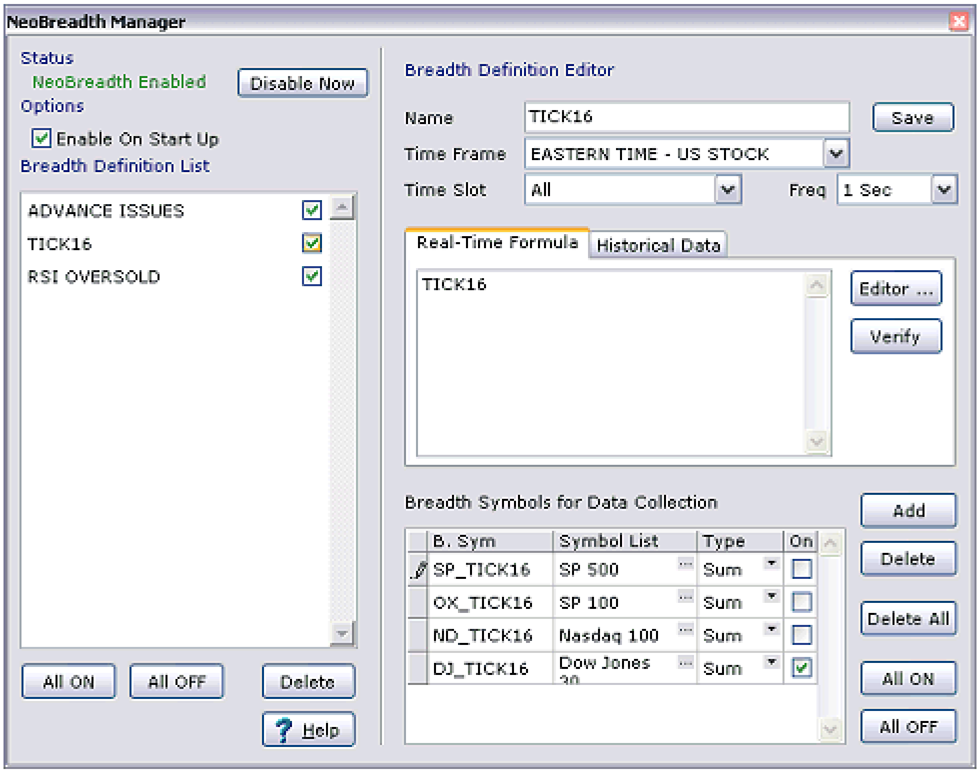
Advance Issues uses prevDClose(M15) to get the previous day close. This is not the only way to calculate advance issues. prevDClose(M15) can be replaced by the quote field "prev".

For regenerating history, Advance Issues uses the formula:

```
c >= prevDClose (data1)
```

with a 1-minute time frame. The low time frame enables an accurate regeneration of historical data.

Example: Tick16



☑ Enable On Start Up

ADVANCE ISSUES	☑
TICK16	☑
RSI OVERSOLD	☑

Name: TICK16

Time Frame: EASTERN TIME - US STOCK

Time Slot: All

Freq: 1 Sec

Real-Time Formula: TICK16

B. Sym	Symbol List	Type	On
SP_TICK16	SP 500	Sum	☐
OX_TICK16	SP 100	Sum	☐
ND_TICK16	Nasdaq 100	Sum	☐
DJ_TICK16	Dow Jones 30	Sum	☑

Tick16 is a breadth symbol based on the Tick16 quote field. Tick16 provides a tick summary of previous 16 ticks. When apply across a symbol list to create a breadth symbol, it provides a superior indicator to the exchange broadcast breadth symbol $TICK.

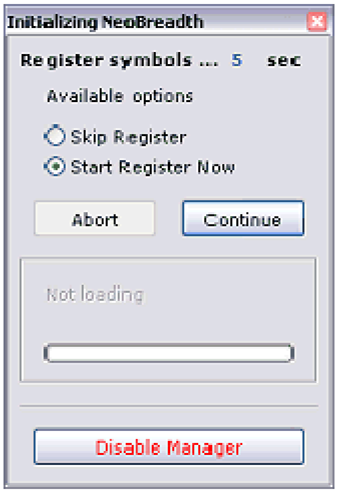
Tick16 breadth has the following advantage:

- When collected in real-time, you have historical data access
- You can control the update frequency
- Because it is a 16-tick summary, unlike the 1-tick summary of $TICK, Tick16 has less noise
- $TICK is NYSE only. With Tick16, you can use any symbol basket

Because Tick16 is a quote field, there is no easy way to regenerate historical data.

Interrupting NeoBreath® on Startup

When starting up with a real-time feed, you have a choice to interrupt NeoBreadth® initialization. The following dialog is opened automatically:



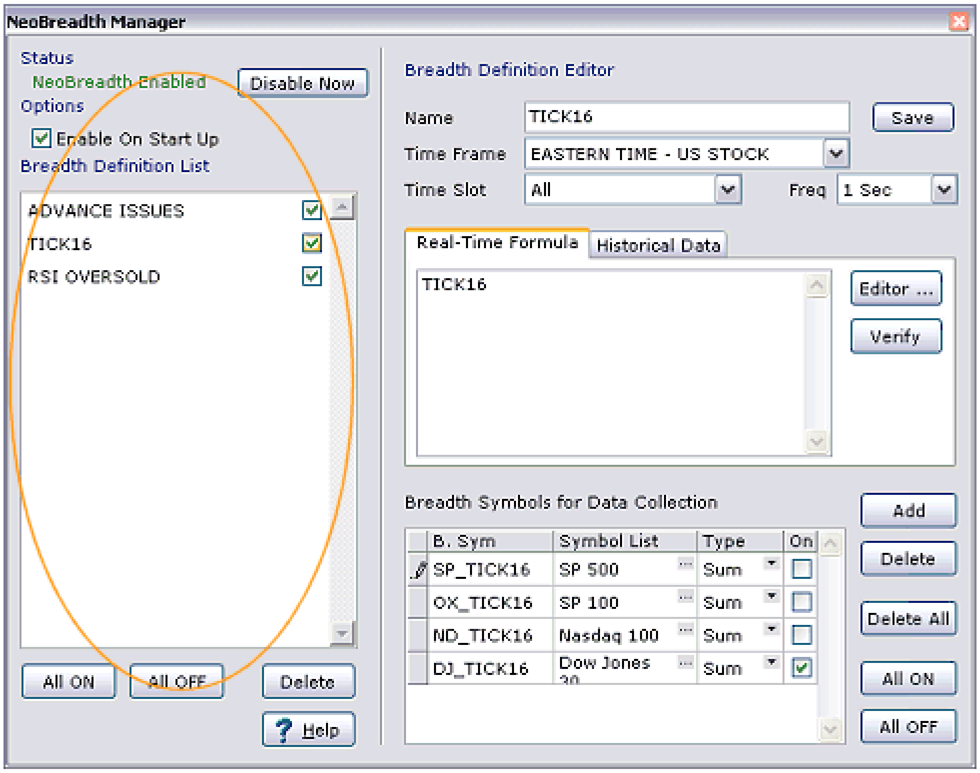
Actions you can take:

- Choose **Skip Register**, then press **Continue** button - Symbols required to generate NeoBreadth® symbols will not be registered (i.e. requested from your data vendor). NeoBreadth® symbols will still be generated, but their values will take longer to stabilize.
- Choose **Start Register Now**, then press **Continue** button - Symbols required to generate NeoBreadth® Symbols will be registered (i.e. requested from your data vendor). This is the default.
- Press **Disable Manager** button - NeoBreadth® Manager will be disabled and symbols will not be registered.

Reference

Breadth Definition Management

Left hand side of NeoBreadth® Manager is for managing breadth definitions.



Status
NeoBreadth Enabled

Options
☑ Enable On Start Up

Breadth Definition List

ADVANCE ISSUES	☑
TICK16	☑
RSI OVERSOLD	☑

Name: TICK16
Time Frame: EASTERN TIME - US STOCK
Time Slot: All
Freq: 1 Sec

TICK16

Breadth Symbols for Data Collection

B. Sym	Symbol List	Type	On
SP_TICK16	SP 500	Sum	☐
OX_TICK16	SP 100	Sum	☐
ND_TICK16	Nasdaq 100	Sum	☐
DJ_TICK16	Dow Jones 30	Sum	☑

Status - This item shows the current status of NeoBreadth®.

Enable Now / Disable Now - Turn on/off all breadth definitions/symbols at once.

Enable On Start Up - Whether NeoBreadth® will be turned on after NeoTicker® starts and connects to a real-time feed.

Breadth Definition List - List all the breadth definitions. If you disable a breadth definition, all breadth symbols using the breadth definition will stop updating. When you click on a name of a breadth definition, Breadth Definition Editor is also updated.

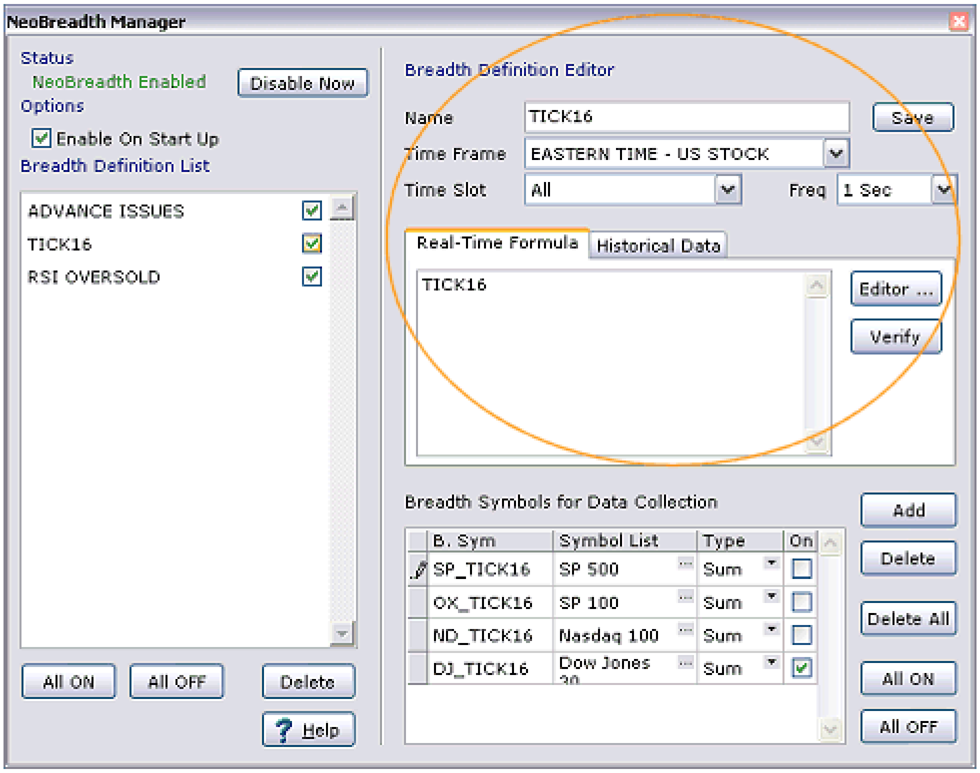
All ON - Turn all breadth definitions on.

All OFF - Turn all breadth definitions off.

Delete - Delete a breadth definition.

Breadth Definition Editor

Upper right hand side of NeoBreadth® Manager is for editing breadth definitions.



Status: NeoBreadth Enabled

☑ Enable On Start Up

Breadth Definition List

ADVANCE ISSUES	☑
TICK16	☑
RSI OVERSOLD	☑

Name: TICK16

Time Frame: EASTERN TIME - US STOCK

Time Slot: All

Freq: 1 Sec

Real-Time Formula: TICK16

B. Sym	Symbol List	Type	On
SP_TICK16	SP 500	Sum	☐
OX_TICK16	SP 100	Sum	☐
ND_TICK16	Nasdaq 100	Sum	☐
DJ_TICK16	Dow Jones 30	Sum	☑

Name - Name of the breadth definition you are working on.

Save - Saves the current breadth definition and symbols. Until you save, the breadth definition and symbols are not active.

Time Frame - Time frame when the breadth definition is updating, i.e. outside of trading time, breadth indicators using the breadth definition are not updated.

Time Slot - When the breadth symbols are updated. Breadth calculation can be CPU intensive. So you have many breadth definitions, you will want to set them to different time slots to reduce spikes in your computer's load.

Freq - Update frequency for the breadth definition.

Delay - If the data is from a delayed source, enter the delay parameter here (in number of minutes). The tick generated will have a delay time stamp, matching that of the source data.

Real-Time Formula - Formula for generating data for constructing breadth symbol. Use quote window formula here.

Historical Data - Formula and time frame for regenerating historical data. Use indicator formula here.

Editor - Launch formula editor to help edit formula.

Verify - Verify the syntax of formula.

The following options are available only under the **Historical Data** tab.

Bar Type - Bar type to use when generating historical data.

Bar Size - Bar size to use when generating historical data.

Days To Load - Number of days to load when generating historical data.

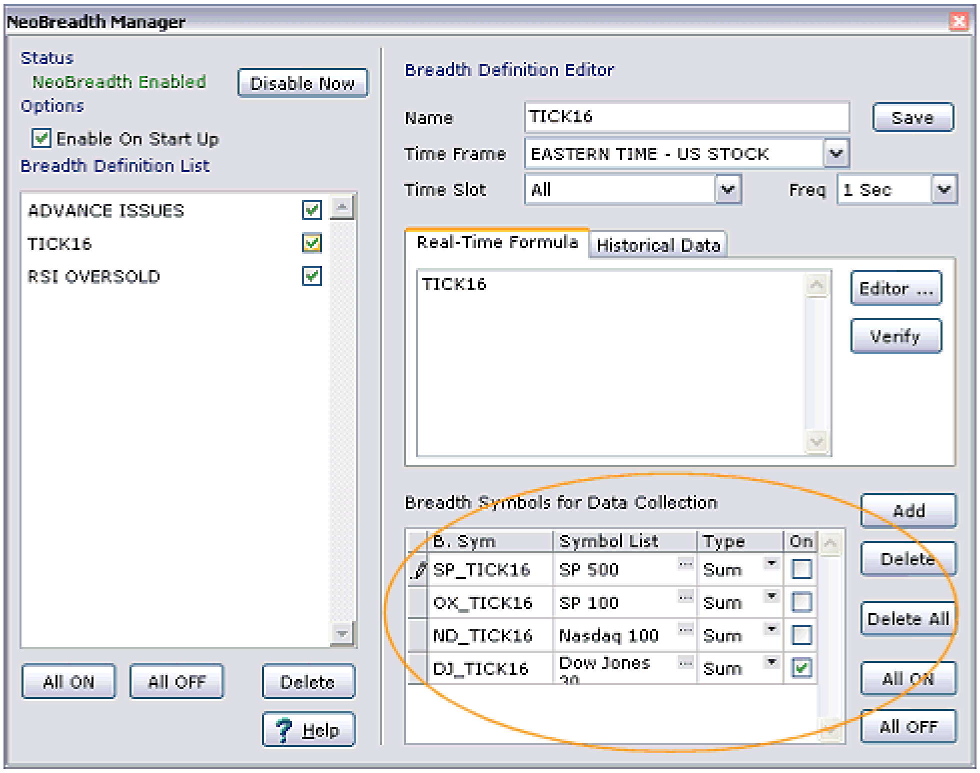
Tick Replay - To be implemented.

History - Open History Generator with the current breadth definition filled in.

Batch Gen - If checked, then this breadth definition will be recognized by batch mode in History Generator.

Breadth Symbols for Data Collection

This area is for defining breadth symbols based on the breadth definition that is currently in Breadth Definition Editor.



NeoBreadth Manager

Status

NeoBreadth Enabled

Options

☑ Enable On Start Up

Breadth Definition List

ADVANCE ISSUES	☑
TICK16	☑
RSI OVERSOLD	☑

Breadth Definition Editor

Name: TICK16

Time Frame: EASTERN TIME - US STOCK

Time Slot: All

Freq: 1 Sec

Real-Time Formula | Historical Data

TICK16

Breadth Symbols for Data Collection

B. Sym	Symbol List	Type	On
SP_TICK16	SP 500	Sum	□
OX_TICK16	SP 100	Sum	□
ND_TICK16	Nasdaq 100	Sum	□
DJ_TICK16	Dow Jones 30	Sum	☑

B. Sym - Breadth symbol. You muse symbol names that are distinct from real symbols.

Symbol List - Symbol list that breadth definition will be applied on to create the breadth symbol.

Type - How to summarize the data collected for each symbol. Choice of Sum, Avg, Min, Max, and Pct.

On - Toggle updating for a breadth symbol.

Add - Add breadth symbol.

Delete - Delete breadth symbol.

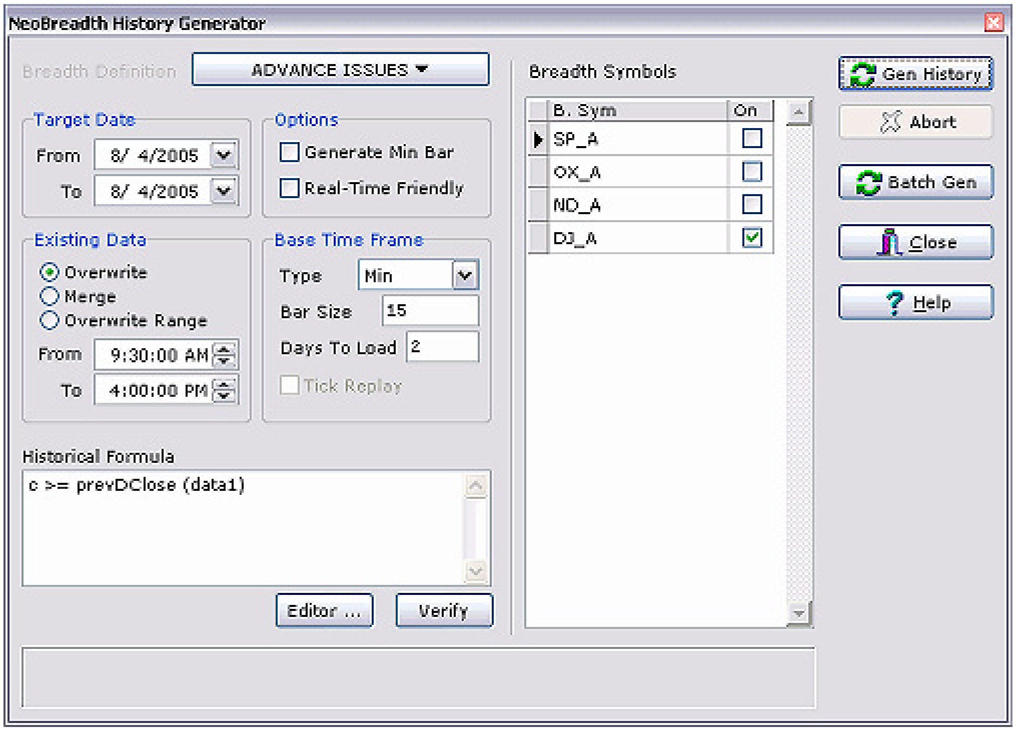
Delete All - Delete all breadth symbols for current breadth definition.

All ON - Turn all breadth symbols on for current breadth definition.

All OFF - Turn all breadth symbols off for current breadth definition.

NeoBreadth® History Generator

This dialog is launch when you press the **Generate History** button under the **Historical Data** tab for regenerating historical data.



NeoBreadth History Generator

Breadth Definition: ADVANCE ISSUES

Target Date

From 8/ 4/2005

To 8/ 4/2005

Options

□ Generate Min Bar

□ Real-Time Friendly

Existing Data

● Overwrite

○ Merge

○ Overwrite Range

From 9:30:00 AM

To 4:00:00 PM

Base Time Frame

Type Min

Bar Size 15

Days To Load 2

□ Tick Replay

Historical Formula

```
c >= prevDClose (data1)
```

Breadth Symbols

B. Sym	On
SP_A	□
OX_A	□
ND_A	□
DJ_A	☑

Breadth Definition - Shows the current breadth definition

Target Date - Date range when historical breadth data is regenerated.

Options - Generate Min Bar - To be implemented

Options - Real-Time Friendly - Default is on. When this option is on, History Generator can generate history data close to real-time and can be used during market hours. When this option is off, History Generator will buffer disk cache data (the **Buffer** field controls number of symbols to be buffered) for maximum performance. You can only turn off **Real-Time Friendly** in off line mode.

Existing Data - To be implemented.

Base Time Frame - Time frame which the regenerating is based on.

Historical Formula - formula used to generate historical data.

Breadth Symbols - Check this list for breadth symbols that will have historical data generated.

Gen History - Press to start generating historical data.

Batch Gen - Batch generate all breadth definitions that have the batch generation option enabled.

News Windows Operation Guide

NeoTicker®'s news window is designed to receive streaming news from the data feed provider. The ability to read news depends on whether you have subscribed to any streaming news service from your data vendor.

Quote.com provides web-based news, not streaming news. Therefore you cannot use the news window with Quote.com.

Enabling News Service

Before you can read news, you must enable the data server to receive news. This is because news data are separated from tick data.

To enable news service:

1 Choose **Program>Server Setup** from the main window.

2 Click on the **Data** tab.

3 Check the **Use News Server** option.

4 Press the **OK** button.

Creating News Window

To create a news window:

- Choose **Window>New>News** from the main window, or
- You can also use the news window button NEWS to create a news window.

By default, all news articles coming from your data service are shown.

Click on a heading to read the news body. If the news body is in hypertext format, a web browser will be opened to let you read the article.

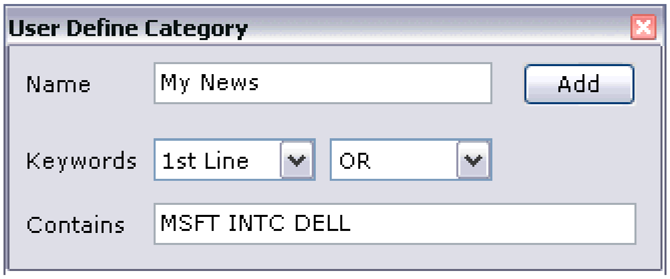
Basic Operations

By default, all news headings from your data vendor are listed. You can click on a news heading to read the news body.

News window provides two additional ways to organize news articles: you can organize the news articles by the news service agency, or by user defined news categories.

News Filtering

You can create user defined news categories to filter news articles. To create a news category, click the **+** button on the news window, and the user define category window will be opened.



Name: My News

Keywords: 1st Line, OR

Contains: MSFT INTC DELL

The following is an example of creating a news category:

Change the name to `My News`. Change the search type to **OR**. Change the **Keywords** to `MSFT INTC DELL`. Then press the **Add** button.

What you have created is a user defined news category that reports all the news that contains MSFT, INTC or DELL in the first line of the news.

Here is an explanation of the items in the user defined category window:

Setting	Meaning
Name	A name you choose for the news category
Keywords	You can filter the news articles by searching keywords in the first line only, or in the whole news story
Search type	Search type can be AND, OR, NOT. If AND is the search type, you will find news articles that contain all of the keywords you provided. If OR is the search type, you will find news articles that contain any of the keywords you provided. If NOT is the search type, you will find news articles that contain none of the keywords you provided
Contains	Enter the keywords for searching here
Add button	Press to add the news category

Organizing News by News Service Agency

Click on the **+** sign besides the **By Service** item to view a list of news service agencies. Click on a news service agency to view the headings of news from that agency.

Deleting User Define News Category

You can delete a user defined news category by selecting the category and press the minus (**-**) button.

Editing User Define News Category

You can edit a user defined news category by selecting the category and press the **Edit** button. After you have pressed the **Edit** button, the user defined category window will be opened with the selected category.

Editing a category is similar to adding a category.

OLE Automation

Overview

NeoTicker® fully supports ActiveX / OLE automation that allows you to control NeoTicker® using VBScript, JavaScript, Win32 programs written in programming languages like VC++, VB, or Delphi.

There are many ways in utilizing automation capabilities of NeoTicker® . For example, you can create scripts to drive NeoTicker® to handle batch jobs like scanning at specific time after market hours, then print each symbol in the scan results on your designated chart window with your favorite indicator setup. All these can be done without your attention, thus saving your time for other task.

The central concept behind automation is dynamic object creation. In order to drive NeoTicker® to automate its tasks, you will create objects based on the pre-defined classes that has the necessary services you want. You will need to get yourself familiar with the NeoTicker® object model before you can fully utilize the automation power.

The underlying architecture of the ActiveX services will be outlined in the following sections with examples.

This section is for automating NeoTicker® from an external program. NeoTicker® itself can call other ActiveX enabled program in scripts/IDL. Do not get the two concepts confused.

Round Brackets vs. Square Brackets

In the reference section, we use a combination of round and square brackets, e.g.

RecalcIndicator(Index, ParamList)

IndicatorName[Index]

Round brackets are used for parameters. Square brackets are used for property indexing. This notation is important only if you program in a language that makes such a distinction, such as Delphi or Visual C++.

If you use Visual Basic and VBScript, you should replace square brackets with round brackets in your code.

OLE Automation Examples

The following examples are written in VBScript. In Windows, you can enter script into a text file using Notepad, then rename the file extension to vbs (e.g. `printgroup.vbs`). When you double click on the script in Windows, the script will drive NeoTicker® to perform specific actions.

OLE Automation requires the application to be running. So you need to start NeoTicker® before running automation script.

Example 1 - Print all Windows

This example opens a group, then print each windows one by one.

```
set nt = createobject ("NeoTicker.App")
nt.opengroup ("mygroup")
nt.printallwindows
set nt = nothing
```

To access specific functionality, you need to create the specific class of ActiveX object in your code.

In this example we have created an instance of the object class `NeoTicker.App.` In the script, you will access NeoTicker® functionality through this object. In this example, the object is used to open a group and print all windows.

When the script is finished, it should set the object to `nothing`. This will stop the communication between NeoTicker® and the script.

Example 2 - Printing Charts

This example will print all the charts, but not other windows such as quote window.

```
set nt = createobject ("NeoTicker.App")

firstwindow = ""

do while true
   ' Get the current active window, and check if it is a chart
   set fw = nt.activewindow

   if fw.Active and (fw.ClassName = "Chart") then

      ' Save the name of the first window
      ' Exit loop when all charts are printed
      if firstwindow = "" then
         firstwindow = fw.Name
      elseif firstwindow = fw.Name then
         exit do
      end if
      nt.PrintActiveWindow

   end if

   ' Make next window active
   nt.NextWindow
loop

set fw = nothing
set nt = nothing
```

In this example, the script loops through all windows in NeoTicker®. An object (`fw`) is created for the window. Window type is checked and if it is a chart, the window is printed.

Example 3 - Pattern Scanner Slide Show

This example requires real-time version of NeoTicker®.

This example requires a pattern scanner and a chart. The pattern scanner should have some scanning result in it. Every 5 seconds, the slide show script will send one symbol from the scanning result from the pattern scanner to the chart.

```
mychart = "chart854"
mypattern = "pattern24"
sleepduration = 5000

set nt = createobject ("NeoTicker.App")

' look for chart

found_chart = false

do while true
   set fw = nt.activewindow

   if fw.Active and (fw.ClassName = "Chart") then

      if firstwindow = "" then
         firstwindow = fw.Name
      elseif firstwindow = fw.Name then
         exit do
      elseif fw.Name = mychart then
         found_chart = true
         exit do
      end if

   end if

   nt.NextWindow
loop

if found_chart then
   set fwchart = nt.activewindow
end if
```

```
' look for pattern scanner

found_pattern = false

do while true
   set fw = nt.activewindow

   if fw.Active and (fw.ClassName = "Pattern") then

      if firstwindow = "" then
         firstwindow = fw.Name
      elseif firstwindow = fw.Name then
         exit do
      elseif fw.Name = mypattern then
         found_pattern = true
         exit do
      end if

   end if

   nt.NextWindow
loop

' slideshow

if found_chart and found_pattern then
   fw.ResultSymbols symbollist

   lb = lbound (symbollist)
   ub = ubound (symbollist)

   for i = lb to ub
      fwchart.setsymbol (symbollist (i))
      wscript.sleep sleepduration
   next
end if

set fw = nothing
set fwchart = nothing
set nt = nothing
```

First, the script will try to find the specified chart and pattern scanner. If they are found, the script will read the pattern scanner result into an OLE variant array.

In a for loop, each symbol from the OLE variant array will be iterated. The symbol is then set to the chart.

This script also uses the `wscript.sleep` method to sleep for 5 seconds between each symbol.

Automation Object Hierarchy

The following automation object classes are available for automation.

App

It is the object class responsible for application-wide services like printing and opening groups, etc.

UserDefineSymbol

It is the object class responsible for the management of User Define Symbol that you usually access through the User Define Symbol Manager.

It has one special method that is not accessible within the manager window, `Update`, which allows external programs to talk to NeoTicker® and update a user defined symbol belonging to the external class.

FunctionWindow

It is the object class responsible for generic control of all function windows.

FWTimeChart

It is descendent of `FunctionWindow` that provides specific control of Time Chart function window.

IDL Interface Objects

There are other object classes defined mainly for the purpose of indicator construction. Basically all the object classes that ends with the suffix "obj" are part of the indicator interface object definitions.

For now, automation cannot use these objects effectively.

App Object - Groups Related

OpenGroupsFromPreviousSession

Opens group from previous session.

NewGroup (GroupName)

Creates a new group called `GroupName`.

OpenGroup (GroupName)

Opens group named `GroupName`.

CloseActiveGroup

Closes the current active group.

SaveActiveGroup

Saves the current active group.

SaveActiveGroupAs (GroupName)

Save the current active group under a new name `GroupName.`

CloseAllGroups (DoSave)

Closes all opened groups and specify whether you are saving the changes.

SaveAllGroups

Saves all the opened groups and their windows.

ActiveGroupName

Returns the name of the current active group.

NextGroup

Makes the next opened group the active one.

IsGroupOpened(GroupName)

Returns true if group named `GroupName` is opened.

SetActiveGroup(GroupName)

Sets the group named `GroupName` to active.

App Object - Printing Related

Print

Same as the **Print** command under the main window **Program** menu.

PrintActiveWindow

Without prompt, prints the current active function window. If the window can be printed with special formatting (pretty print), it will be used automatically.

PrintActiveWindowScreenshot

Without prompt, prints the active window as is.

PrintActiveGroup

Prints each function windows in the current active group.

PrintAllGroups

Prints all opened groups one by one.

PrintDesktop

Prints the desktop as if you are taking a screen shot of it.

App Object - Function Window Related

ActiveWindowName

Returns the active function window name.

NextWindow

Makes the next function window the active one.

ActiveWindow

Returns a `FunctionWindow` object of the current active function window. If there is no active window, returns nil instead.

FunctionWindowExists(WName, WType)

Check if a function window called `WName` of `WType` already exists. `WName` is a string parameter. `WType` is an integer constant. `FunctionWindowsExists` return true or false depending on the existence of the function window.

See ***Numeric Constants for Function Type*** (on page 680) for more information on function window type.

FunctionWindowInActiveGroup(WName)

Returns a `FunctionWindow` object of the name `WName`. `WName` is a string parameter. If there is no matching function window, nil is returned.

GetFunctionWindowList(WType, WindowList)

Returns the list of function windows of window type WType. WindowList is an array of variant string.

See ***Numeric Constants for Function Type*** (on page 680) for more information on function window type.

NewFunctionWindow(WName, WType)

Creates and returns a function window of the name `WName` of type `WType`. `WName` is a string parameter. `WType` is an integer constant. If the function window already exists, it will be overwritten.

See ***Numeric Constants for Function Type*** (on page 680) for more information on function window type.

NewFunctionWindowByTemplate(Template, WType)

Creates and returns a function window using the template called `Template` of type `WType`. `Template` is a string parameter. `WType` is an integer constant. If the template does not exists, it will return nil.

See ***Numeric Constants for Function Type*** (on page 680) for more information on function window type.

NewFunctionWindowByTemplateEx(Template, WType, WName)

Creates and returns a function window of the name `WName` using the template called `Template` of type `WType`. `WName and Template` are string parameters. `WType` is an integer constant. If the template does not exists, it will return nil.

See ***Numeric Constants for Function Type*** (on page 680) for more information on function window type.

SetTemplateSymbol(ASymbol)

Call this method before calling `NewFunctionWindowTemplate` to set the symbol for templates that contains symbol. The symbol in the template will be replaced by `ASymbol`. `ASymbol` is a string parameter.

OpenFunctionWindow(WName, WType)

Opens function window of the name `WName` of type `WType` to the current active group. `WName` is a string parameter. `WType` is an integer constant.

See ***Numeric Constants for Function Type*** (on page 680) for more information on function window type.

FunctionWindowClassName[WType]

Returns the class name of `WType`. For example,

`App.FunctionWindowClassName[4]` returns 'Time Chart'.

See ***Numeric Constants for Function Type*** (on page 680) for more information on function window type.

App Object - Misc

ActivateIndicator (AFilename)

Activate the indicator under the file name `AFilename`. `AFilename` is a string.

ClearScriptErrorLog

Clear all messages in the script error log window.

DeactivateIndicator (AnIndicatorName)

Deactivate an indicator with name `AnIndicatorName`. `AnIndicatorName` is a string.

Debug (AValue)

Display a debug message based on `AValue`, a variant value, in the script error log window.

ExitNow (Prompt)

Tells NeoTicker® to exit. `Prompt` is a boolean value that determines whether to prompt user before exit.

InstallSuperpositionDLL (Filename)

Tell NeoTicker® to install the DLL file specified by `Filename` as a new superposition style

IsMinimized

Returns whether the application is minimized.

IsMaximized

Returns whether the application is minimized.

IsNormal

Returns whether the application is in normal window state.

Online

Read-only boolean. Returns true when NeoTicker® is connected to a real-time server (data feed or Trade Simulator).

ReplaceChainSymbolPrompt

Same as the **Replace Symbol in Chain** command under the main window **Program** menu.

ReplaceChainSymbol (NewSymbol)

Without prompt, replaces the active group chained function windows with the symbol `NewSymbol.`

Report

Read-only property. Returns the `IReportObj` for access to all report function windows.

TakeScreenshot(TargetSetting : string; Fullscreen : boolean; WithBorder : boolean)

Take screenshot.

`TargetSetting` is the name of the target. Targets are defined in Screenshot tool ***(see Screen*** Shots (on page 588)).

If `Fullscreen` is true, a full screen screenshot is taken.

If `Fullscreen` is false, screenshot of active window is taken. `WithBorder` controls whether the border is included in the screenshot.

TakeScreenshotEx (TargetSetting : string; CaptureType :integer; UpdateMessage : boolean; Message : string; CustomFilename : boolean; Filename : string)

Take screenshot.

`TargetSetting` is the name of the target. Targets are defined in Screenshot tool ***(see Screen*** Shots (on page 588)).

`CaptureType` equals to 0 for last used; 1 for desktop; 2 for window with border; 3 for window without border; and negative value (i.e. -N) for monitor N.

Set `UpdateMessage` to true to change the user message to `Message` within the `TargetSetting`.

Set `CustomFilename` to true to use a custom `Filename` over the predefined one set within the `TargetSetting`.

UninstallSuperpositionDLL (SuperpositionName)

Tell NeoTicker® to uninstall the superposition style with name `SuperpositionName`

UserID

Returns user id of user.

VersionNum

Returns version number as an integer. The last 2 digits is the minor version, e.g. for version 3.11, `VersionNum` returns 311; for version 3.2, `VersionNum` returns 320.

VersionInfo

Returns the version information about instance. The following codes are used

0 - EOD

1 - Real-time demo version

2 - Real-time lease version

3 - Real-time full version

4 - Reserved

FunctionWindow Object

Activate

Makes the function window the active window. This call works only if the window is in current active group.

Active

Returns the boolean state whether the window is within the active group.

Busy

Returns the boolean state whether the window is busy processing something.

ClassName

Returns the class name of the function window. It can be one of the following:

```
Quote
T&S
News
VolDist
Chart
Ticker
Alerts
Pattern
Option
Level II
Report
Sys Mon
TopList
DynamicTable
Cluster
```

Close

Closes the window. User will be prompt to confirm.

Chained : string (read/write property)

Chain property of a function window can be set through this property. Valid settings include,

```
Not Chained
Global Chain
Red
Green
Blue
Cyan
Magenta
Yellow
White
```

DistributeTarget : boolean (read/write property)

Equivalent to the function window property with same name that can be set through the general function window popup menu.

ExportChartToClipBoard

Exports the chart to clipboard. This allows other program to paste the chart from the clipboard. This method is applicable only to function windows that hosts a chart, e.g. time chart, volume distribution chart, cluster.

ExportChartToClipBoardCustomMessage (AMessage : string)

Exports the chart to clipboard with customized display message.

ExportChartToFile (Filename)

Exports the chart to a file. This method is applicable only to function windows that hosts a chart, e.g. time chart, volume distribution chart, cluster.

This method supports the following file formats: BMP, GIF and PNG. The format is determined by the file suffix, e.g. if you export to the file `mychart.gif`, then the graphics format of the file will be GIF.

ExportChartToFileCustomMessage (Filename, AMessage : string)

Exports the chart to a file with customized display message.

ForceClose

Closes the window. User will not be prompt to confirm.

IsMinimized

Returns whether the function window is minimized.

IsMaximized

Returns whether the function window is minimized.

IsNormal

Returns whether the function window is in normal window state.

Minimize

Minimizes the function window.

Maximize

Maximizes the function window.

Name

Returns the name of the function window.

Restore

Restores a minimized/maximized function window to its normal size.

Move(Left, Top)

Move the function window.

Resize(Width, Height)

Resize the function window.

Save

Saves the function window.

SaveAs(filename, overwrite)

Saves the function window. Filename is a string. Overwrite is an integer (0 - do not overwrite; 1 -prompt user before overwritting; 2 - force overwrite).

SetSymbol (ASymbol)

Changes the symbol of the function window to `Asymbol`. If the function window is busy, it will reject the change request.

WindowBorder : boolean (read/write property)

Equivalent to the function window property with same name that can be set through the general function window popup menu.

WriteProtected : boolean (read/write property)

Equivalent to the function window property with same name that can be set through the general function window popup menu.

FWPattern Object

ClearResults

Clears the scanning result.

ExportSymbolList

Opens the export symbol list dialog.

InputFromAllCurrentSymbol : boolean

Read/Write property. If true, enable the Input option All Current Symbol.

InputFromPatternScanner : boolean

Read/Write property. If true, enable the Input option Pattern Scanner.

InputFromPatternScannerName : string

Read/Write property. If Input option Pattern Scanner is enabled, then this property is the pattern scanner window name.

InputFromQuoteWindow : boolean

Read/Write property. If true, enable the Input option Quote Window.

InputFromQuoteWindowName : string

Read/Write property. If Input option Quote Window is enabled, then this property is the quote window name.

InputFromSymbolList : boolean

Read/Write property. If true, enable the Input option Symbol List.

InputFromSymbolListName : string

Read/Write property. If Input option Symbol List is enabled, then this property is the Symbol List name.

InputFromTopList : boolean

Read/Write property. If true, enable the Input option Top List.

InputFromTopListName : string

Read/Write property. If Input option Top List is enabled, then this property is the top list window name.

LastDayUseMostRecentData : boolean

Read/Write property. If true, set the Last Day data setting to Use Most Recent Data.

LastDaySpecificDate : double

Read/Write property. If Last Day data setting is set to Specific Date, then this property is the last date for the scanner. Value is in internal date format.

OutputToExcel : boolean

Read/Write property. If true, enable the Output option Excel.

OutputToExcelWorkbook : string

Read/Write property. If Output option Excel is enabled, then this property is the workbook name.

OutputToExcelWorksheet : string

Read/Write property. If Output option Excel is enabled, then this property is the worksheet name.

OutputToQuoteWindow : boolean

Read/Write property. If true, enable the Output option Quote Window.

OutputToQuoteWindowName : string

Read/Write property. If Output option Quote Window is enabled, then this property is the quote window name.

OutputToSymbolList : boolean

Read/Write property. If true, enable the Output option Symbol List.

OutputToSymbolListFilename : string

Read/Write property. If Output option Symbol List is enabled, then this property is the file name.

ResultSymbols (SymbolList : variant)

Reads scanning result symbols into `SymbolList`. `SymbolList` is an OLE variant array.

ResultTable (Table : variant)

Reads scanning result symbols into `Table`. `Table` is a 2-dimensional OLD variant array.

ScanningInProgress : boolean

Read-only property. Returns true if pattern scanner is scanning at the moment.

ScanNow

Starts scanning.

StartTimer

Starts timer for scheduled scanning.

StopNow

Stops scanning.

StopTimer

Stops timer for scheduled scanning.

ScanNowUsingSymbols (SymbolList : variant)

Starts scanning now with `SymbolList`. `SymbolList` is an OLE variant array.

TimerActive : boolean

Read-only property. Returns true if pattern scanner has its timer enabled.

FWQuote

AddSymbol(Symbol : string, IgnoreRepeat : boolean)

Adds `Symbol` to quote window. If `IgnoreRepeat` is true and if the symbol is a duplicate, i.e. quote window already has the symbol, the symbol is not added.

ColumnNames : Variant

Returns an variant array of all the currently visible columns.

ClearAllSymbols

Removes all symbols in quote window.

CurrentRowSymbol : string

Read/Write property. The symbol located at the current row.

ExportTableToCSV (Filename : string; Overwrite : boolean)

Export table to a text file in CSV format. `Filename` specifies the target location for exporting the data. `Overwrite` controls if NeoTicker can overwrite an existing file at the target location.

ExportTableToExcel

Export table to Excel.

JumpToFirstRow (SendToChain : boolean)

Jump to first row of quote window. `SendToChain` determines if the symbol located at the first row will be sent to the chain.

JumpToLastRow (SendToChain : boolean)

Jump to last row of quote window. `SendToChain` determines if the symbol located at the first row will be sent to the chain.

JumpToNextRow (SendToChain : boolean)

Jump to next row from the current row of quote window. `SendToChain` determines if the symbol located at the first row will be sent to the chain.

JumpToRowBySymbol (Symbol : string; SendToChain : boolean)

Jump to first row in the quote window with `Symbol`. `SendToChain` determines if the symbol located at the first row will be sent to the chain.

SetSymbolList(SymbolList)

Replaces all symbols in quote window with the ones in `SymbolList`. `SymbolList` is an OLE variant array.

SortByColumn (ColumnName : string; Ascending : boolean)

Sort quotes using values in the column specified by `ColumnName`. `Ascending` determines the sorting order is ascending or descending.

StartTimerSort

Enable automatic sorting.

StopTimerSort

Disable automatic sorting.

TimerSortActive : boolean

Read-only property. Returns true if timer sort is currently enabled.

FWTimeChart Object

AutoScroll : boolean (read/write property)

Equivalent to the auto scroll visual setting.

AutoScrollPercentFromRightEdge : integer (read/write property)

Equivalent to the auto scroll percent from right edge visual setting.

ChartAtRightEdge

Returns true if the chart is at the right edge, i.e. most recent bar of primary data series is visible in the chart.

ChartBegin

Scrolls to beginning of the chart. This is equivalent to pressing the HOME key interactively.

ChartEnd

Scrolls to end of the chart. This is equivalent to pressing the END key interactively.

CompleteReload

This method reload all the data from cache and recalculates all the indicator. This is the automation of the reload button on the tool bar. See ***Completely Reloading all Data*** (on page 1123).

CompleteRecalc

This method recalculates all indicators in the chart. This is the automation of the recalculate button on the tool bar. See ***Recalculating Indicator*** (on page 1107).

DataCount

This property returns the number of data series reside in the chart.

DataName[Index]

Returns the Data proper name (in olestring) of `Index` ranged from 0 to `DataCount - 1`.

DefaultZoomFitAllBars : boolean (read/write property)

Equivalent to the fit all bars visual setting.

DefaultZoomBarcount : integer (read/write property)

Equivalent to the default number of bars to show on screen visual setting.

DeleteAllDrawingObjects

This method deletes all drawing tools in the chart.

DeleteIndicator (Index : integer; Prompt : boolean)

This method deletes indicator at `Index` position in the chart. `Index` range from 0 to `IndicatorCount` - 1. `Prompt` when sets to true will allow user to finalize the delete decision.

GetChartDataRange(AStart, AEnd)

Returns the data range of the chart through the AStart, AEnd variables. AStart and AEnd are double representing the date time of the chart edges in internal format.

GetChartVisibleRange(AStart, AEnd)

Returns the visible range of the chart through the AStart, AEnd variables. AStart and AEnd are double representing the date time of the chart edges in internal format.

GetChartVisible returns true if the function succeed. GetChartVisible can fail and returns false if the chart does not have data or is in the middle of adjusting the chart edges.

GetPositionNetProfitList (Indicator, PosList)

GetPositionNetProfitList (Index, PosList)

GetPositionNetProfitList returns the net profit list of the trading system in an array of variant strings.

To identify the indicator, you can either supply the indicator name as a string, or the indicator index.

IndicatorCount

This property returns the number of indicators reside in the chart.

IndicatorName [Index]

This property returns the name of the indicator referenced by `Index`. `Index` is 0-based, and ranged between 0 to `IndicatorCount – 1`.

IndicatorParamCount [Index]

This property returns the number of parameters used by the indicator indexed by `Index`. `Index` is 0-based, and ranged between 0 to `IndicatorCount – 1`.

IndicatorParams [Index]

This property returns the parameter list of the indicator indexed by `Index`. `Index` is 0-based, and ranged between 0 to `IndicatorCount – 1`. The parameter list returned is an OLE array of string.

IndicatorPlotNames (Index, PlotList)

Returns `PlotList` as an variant array containing the plot names of the indicator.

IndicatorSystemStats [Index]

This property returns the trading system statistics of the indicator indexed by `Index`. `Index` is 0-based, and ranged between 0 to `IndicatorCount - 1`. The returned statistics is stored in a TradingSystemStats object (see ***TradingSystemStats Object*** (on page 677)).

The statistics is a snapshot. You must call `SnapshotSystemResult` first to collect the statistics before calling `IndicatorSystemStats`. The statistics value will not automatically change until you call `SnapshotSystemResult` again.

This property is safe even if the indicator is not intended to be a trading system.

JumpTo(HasEndTime, StartTime, EndTime)

Jumps to a specified start time in the chart. If end time is specified, the chart will scaled the right hand side to the end time.

`HasEndTime` is boolean.

`StartTime, EndTime` are double, using Windows date time.

LastValidValues (DataList, IndicatorList)

Returns the last valid values of all data and indicators.

`DataList` is a variant array of variant array <datetime, open, high, low, close, volume, tick, openinterest>.

`IndicatorList` is an variant array of variant array <plot1, plot2, ..., plotN>.

RecalcData(Index, Symbol, TimeFrame, BarSize)

Replaces data series with Symbol, TimeFrame, Barsize. Index ranges from 0 to `DataCount` - 1. Symbol is a string. TimeFrame is one of the pre-defined constants in type library: `ppTick, ppSec, ppMin, ppHour, ppDaily, ppWeekly, ppMonthly, ppQuarterly, ppYearly`. For script-based automation which does not use type library , use 0 for `ppTick`, 1 for `ppSec` and so on. Barsize is integer.

Index ranges from 0 to `DataCount` - 1.

RecalcIndicatorOneParamChange(Index, ParamIndex, NewParamValue)

Changes parameter to new value and recalculates the indicator.

Index ranges from 0 to `IndicatorCount - 1`.

ParamIndex ranges from 0 to `IndicatorParamCount[Index] - 1`

NewParamValue is the new parameter value, in string representation.

RedrawOnTimer : boolean (read/write property)

Equivalent to the redraw on timer visual setting.

RedrawOnTimerFrequency : integer (read/write property)

Equivalent to the redraw on timer frequency visual setting.

RightSideSpacing : integer (read/write property)

Equivalent to the right side spacing visual setting.

ScrollByBar(ARight, ABars)

Scroll the chart by bar. ARight is a boolean. It determines the direction of scrolling. If ARight is set to true, the chart is scrolling to the right, otherwise the chart scrolls to the left.

ABars is an integer representing the number of bars to be scrolled.

ScrollByPage(ARight, APages)

Scroll the chart by page. A page is the area of the chart that is currently visible. ARight is a boolean. It determines the direction of scrolling. If ARight is set to true, the chart is scrolling to the right, otherwise the chart scrolls to the left.

APages is a double representing the number of pages to be scrolled.

SetValue

Reserved for future functionality.

SnapshotSystemResult (Index)

This method takes a snapshot of the indicator's trading system statistics. The indicator is referenced by `Index`. `Index` is 0-based, and ranged between 0 to `IndicatorCount - 1`.

After you take a snapshot, you can use `IndicatorSystemStats` to obtain the statistics.

This method is safe even if the indicator is not intended to be a trading system.

SnapshotSystemResultByDateRange (Index, StartDate, EndDate)

This method is similar to `SnapshotSystemResult`, except `SnapshotSystemResultByDateRange` allows you to specify a date range. The statistics collected is ranged from the start date and end date, inclusive. `StartDate` and `EndDate` are in PC date format.

TickReplay(ReplayStyle, LastNDays : integer; FromDate : double)

Starts a tick replay.

`ReplayStyle` is an integer, controlling which part of the data is replayed:

0 - replays all data

1 - replays last N days of data

2 - replays from a specific data time

If `ReplayStyle` is 1, the `LastNDays` parameter specifies the number of days to replay.

If `ReplayStyle` is 2, the `FromDate` parameter specifies the starting date of the data to be replayed. `FromDate` is a double representing the date time in internal format.

ZoomIn

Zooms in to the chart. This is equivalent to pressing the + key interactively.

ZoomInSmall

Zooms in to the chart. This is equivalent to pressing the SHIFT+ keys interactively.

ZoomOut

Zooms out to the chart. This is equivalent to pressing the - key interactively.

ZoomOutSmall

Zooms out to the chart. This equivalent to pressing the SHIFT- keys interactively.

TradingSystemStats Object

Statistics

A TradingSystemStats object contains statistics of a trading system. This object is returned by calls such as `FWTimeChart.IndicatorSystemStats`. Statistics are read-only.

When requesting a statistics, you will need to supply an index value:

- Index = 0 returns combined statistics
- Index = 1 returns long side statistics
- Index = -1 returns short side statitics

The following calls are supported:

AccountSizeRequired [AnIndex : integer] : double

AnnualizedReturn [AnIndex : integer] : double

AnnualizedReturnPct [AnIndex : integer] : double

AverageDurationInBars [AnIndex : integer] : double

AverageLoser [AnIndex : integer] : double

AverageProfit [AnIndex : integer] : double

AverageWinner [AnIndex : integer] : double

AvgWinToAvgLoss [AnIndex : integer] : double

BestWinner [AnIndex : integer] : double

CAGR [AnIndex : integer] : double

ClosedNetProfit [AnIndex : integer] : double

ClosedNetProfitPct [AnIndex : integer] : double

Commission [AnIndex : integer] : double

DurationInMkt [AnIndex : integer] : integer

DurationInMktPct [AnIndex : integer] : double

GrossLoss [AnIndex : integer] : double

GrossProfit [AnIndex : integer] : double

LoserAverageDuration [AnIndex : integer] : double

Losers [AnIndex : integer] : integer

MARRatio [AnIndex : integer] : double

MaxConsecutiveLosers [AnIndex : integer] : integer

MaxConsecutiveWinners [AnIndex : integer] : integer

MaxDDSinglePosition [AnIndex : integer] : double

MaxDDEquity [AnIndex : integer] : double

MaxDDPctEquity [AnIndex : integer] : double

MaxPositionSize [AnIndex : integer] : integer

NonScratchPositions [AnIndex : integer] : integer

OpenPositionPL [AnIndex : integer] : double

PositionsTaken [AnIndex : integer] : integer

ProfitFactor [AnIndex : integer] : double

QualityScore [AnIndex : integer] : double

SharpeRatio [AnIndex : integer] : double

WinnerAverageDuration [AnIndex : integer] : double

WinnerPct [AnIndex : integer] : double

Winners [AnIndex : integer] : integer

WinnersToLosers [AnIndex : integer] : double

WorstLoser [AnIndex : integer] : double

Above are the same statistics displayed in System Performance Viewer. See ***Statistics Reference*** (on page 988)

GetStatValues(StatList)

Returns statistics to StatList. StatList is a 2-D OLE variant array. Each row is a 4-tuple: (stat name, stat value combined, stat value long, stat value short), representing the statistics' name, long/short combined statistics, long side statistics, short side statistics.

GetStatValues returns built-in and custom statistics.

Combined Statistics (Obsoleted)

Calls in this section are for backward compatibility only. For a complete list of statistics, see ***TradingSystemStats Object, Statistics*** (see "Statistics" on page 677).

The following properties are exposed for combined long and short statistics:

Combined_ClosedNetProfit

Returns net profit.

Combined_DurationInMarket

Returns duration in market in number of days. Fraction represents parts of a day.

Combined_AvgWinToAvgLoss

Returns average win to loss ratio.

Combined_SharpeRatio

Returns Sharpe ratio.

Combined_WinnerPercent

Returns winner pecentage.

Combined_ProfitFactor

Returns profit factor.

Combined_DurationInMarketPercent

Returns duration in market, in percentage.

Combined_PositionsTaken

Returns the number of positions taken.

UserDefineSymbol Object

Exists (Symbol)

Whether a symbol is a user define symbol.

Active (Symbol)

Checks whether a symbol is enabled.

Update (Symbol, Price)

Updates user define symbol with a price value.

This call is for updating user define symbol that are set as External type.

If the user define symbol is updated by tick, this call is equivalent to inserting a new tick to the user define symbol.

If the user define symbol is not updated by tick, this call will update the latest price of the user define symbol, while tick generation will rely on user define symbol's own timer.

UpdateWithVolume(Symbol, Price, Volume)

This call is identical to Update, with additional volume information.

Numeric Constants for Function Type

Some method calls such as `NewFunctionWindow` expects a function window type parameter call `WType`. `WType` is an integer and valid values are defined by constants in type library.

The constants are listed below:

```
      Quote
      TimeSales
      News
      VolDist
      TimeChart
      Ticker
      Alert
      Pattern
      Option
      Level2
      Report
      Monitor
      TopList
      Cluster
      QuoteMemo
```

VBScript Issue

VBScript lacks the ability to read type library constants. So you need to explicitly declare these constants in your VBScript. You can cut and paste the code segment provided below into your VBScript to declare these constants.

Most other scripting environment does not have this problem.

```
const
    Quote     = 0
    TimeSales = 1
    News      = 2
    VolDist   = 3
    TimeChart = 4
    Ticker    = 5
    Alert     = 7
    Pattern   = 8
    Option    = 9
    Level2    = 10
    Report    = 11
    Monitor   = 12
    TopList   = 15
    Cluster   = 17
    QuoteMemo = 18
```

Option Chain Window Operation Guide

NeoTicker®'s option chain window lets you quote stock option prices in real-time. It provides many convenient features to help you to trade options, such as retrieving option symbol information from TickQuest's website and highlighting the most recently traded option.

Option chain only works if you are a subscriber of eSignal or QCharts.

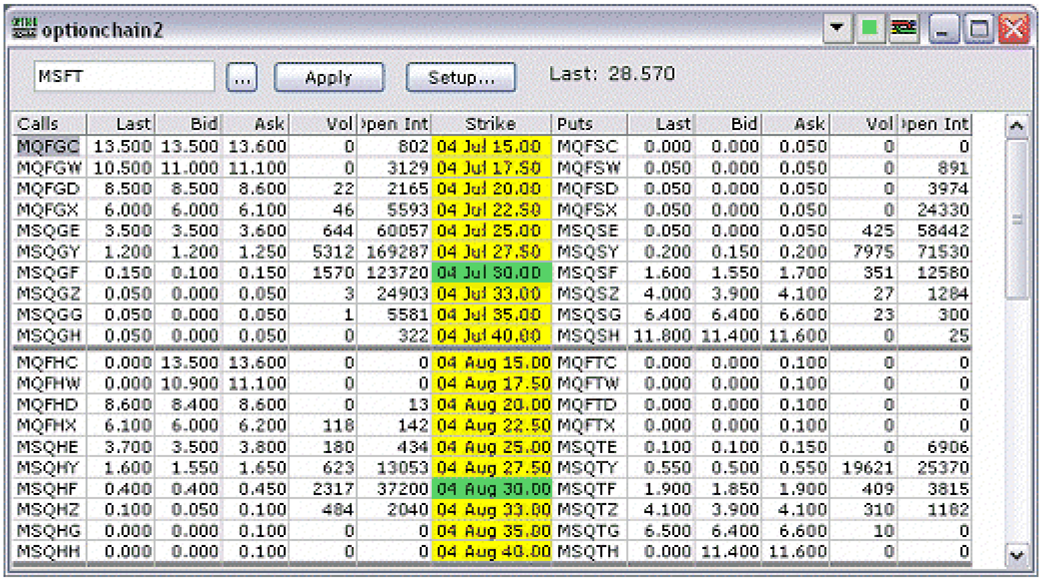
Creating Option Chain Window

To create an option chain window, either

- Press the option chain button OPTION, or
- choose **Window>New>Option Chain** from the main window.

Basic Operations

The following is an option chain window.



MSFT

Last: 28.570

Calls	Last	Bid	Ask	Vol	›pen Int	Strike	Puts	Last	Bid	Ask	Vol	›pen Int
MQFGC	13.500	13.500	13.600	0	802	04 Jul 15.00	MQFSC	0.000	0.000	0.050	0	0
MQFGW	10.500	11.000	11.100	0	3129	04 Jul 17.50	MQFSW	0.050	0.000	0.050	0	891
MQFGD	8.500	8.500	8.600	22	2165	04 Jul 20.00	MQFSD	0.050	0.000	0.050	0	3974
MQFGX	6.000	6.000	6.100	46	5593	04 Jul 22.50	MQFSX	0.050	0.000	0.050	0	24330
MSQGE	3.500	3.500	3.600	644	60057	04 Jul 25.00	MSQSE	0.050	0.000	0.050	425	58442
MSQGY	1.200	1.200	1.250	5312	169287	04 Jul 27.50	MSQSY	0.200	0.150	0.200	7975	71530
MSQGF	0.150	0.100	0.150	1570	123720	04 Jul 30.00	MSQSF	1.600	1.550	1.700	351	12580
MSQGZ	0.050	0.000	0.050	3	24903	04 Jul 33.00	MSQSZ	4.000	3.900	4.100	27	1284
MSQGG	0.050	0.000	0.050	1	5581	04 Jul 35.00	MSQSG	6.400	6.400	6.600	23	300
MSQGH	0.050	0.000	0.050	0	322	04 Jul 40.00	MSQSH	11.800	11.400	11.600	0	25
MQFHC	0.000	13.500	13.600	0	0	04 Aug 15.00	MQFTC	0.000	0.000	0.100	0	0
MQFHW	0.000	10.900	11.100	0	0	04 Aug 17.50	MQFTW	0.000	0.000	0.100	0	0
MQFHD	8.600	8.400	8.600	0	13	04 Aug 20.00	MQFTD	0.000	0.000	0.100	0	0
MQFHX	6.100	6.000	6.200	118	142	04 Aug 22.50	MQFTX	0.000	0.000	0.100	0	0
MSQHE	3.700	3.500	3.800	180	434	04 Aug 25.00	MSQTE	0.100	0.100	0.150	0	6906
MSQHY	1.600	1.550	1.650	623	13053	04 Aug 27.50	MSQTY	0.550	0.500	0.550	19621	25370
MSQHF	0.400	0.400	0.450	2317	37200	04 Aug 30.00	MSQTF	1.900	1.850	1.900	409	3815
MSQHZ	0.100	0.050	0.100	484	2040	04 Aug 33.00	MSQTZ	4.100	3.900	4.100	310	1182
MSQHG	0.000	0.000	0.100	0	0	04 Aug 35.00	MSQTG	6.500	6.400	6.600	10	0
MSQHH	0.000	0.000	0.100	0	0	04 Aug 40.00	MSQTH	0.000	11.400	11.600	0	0

Setting/Button	Function
Symbol Field	Lets you enter the stock symbol here
...	Lets you pick the stock symbol from the real-time lookup table
Apply	Retrieves the option quotes. If you have not previously quote options for this stock, NeoTicker® will retrieve the option symbols from your data vendor
Setup	Lets you set up how the option chain window displays the option
Last traded price	Quotes the last traded price of the stock

The option chain window uses the following colors to help you read the option quotes:

Color	Meaning
Yellow	Expiration date and strike price
Green	Expiration date and strike price when the strike price is close to the last traded price of the stock
Cyan	The option has been traded recently
Red	The option price is lower than the close price (in the net column only)
Green	The option price is higher than the close price (in the net column only)

Specifying Stock Symbol

Enter the stock symbol in the field, and press the **Apply** button.

- For Quote.com users, quoting an option chain for the first time is a time consuming operation. It will take several seconds to several minutes for the initialization to complete.
- For eSignal users, please choose **Download Option Symbols** in the pop up menu. This will download option symbols from TickQuest's Website as eSignal does not provide a complete option chain information to NeoTicker®.

Reading Option Chain Window

Option chain window uses the following coloring to help you read the option quotes:

Yellow	Expiration date and strike price
Green	Strike price that is close to the current trading price
Cyan	The option is recently traded

The recently traded options are updated in real-time during trading hours. The last traded price of the stock is also displayed.

Loading Option Information from TickQuest's Website

This section is obsoleted. Nowadays data vendors provided up to date option symbol resolving. It is no longer necessary to resolve option symbols from TickQuest website.

Customizing Option Chain Window

You can change the way options are displayed to suit your trading need. Press the **Setup** button. The option chain setup window will be opened.

For example, if you choose **Sort by strike price**, and press the **Apply** button, the option chain will sort the options by their strike price.

Customizing Columns

The columns in the option chain windows are customizable. You can hide them or rearrange them anyway you like.

In this example, we customize the option chain window so that it is more suitable for trading call options.

Press the **Setup** button in the option chain window. Click on the **Columns** tab in the setup window. Disable all the put option columns, and enable all the call option columns.

Drag and drop the **Strike** column to the first column to complete the example.

Setting up Option Display

You can open the option chain setup window by pressing the **Setup** button or by choosing **Setup** in the pop up menu. The option chain setup window controls how the option chain window interprets and displays option quotes.

General Tab

Setting	Meaning
Stock Symbol	You can change the stock symbol here .
Show a small range of strike prices	This option restricts the range of strike price displayed in the option chain window. In general, options with strike prices that are too far away from the last traded price of the stock are not displayed. By default, this option is true.
Do not show expired options	Your real-time database may still have quotes of expired options. If this option is true, NeoTicker® will not display these options. By default, this option is true .
Decimal Pricing	Displays the option quotes in decimal pricing. By default, this option is false.
Do Not Show LEAPS	Controls whether LEAPS are displayed. By default, this option is true.
Sort by expiration	Sorts the options by their expiration date.
Sort by strike price	Sorts the options by their strike price.
Sort by root	Sorts the options by their root symbol.

Column Tab

Check boxes under the columns tab control whether a specific column is displayed for the option quotes.

Option Database

Option quotes are notorious for their problems with expiration dates and strike prices. To help you solve these problems, option chain window maintains an option database. This database is used to look up expiration dates and strike prices when you quote an option.

The expiration dates and strike prices information inside the database come from the following sources:

- Option chain window knows how to obtain strike prices and expiration dates from your data vendor (if your data vendor supports this feature). This is the default way to resolve strike prices and expiration dates.
- Option chain window can make intelligent guess on strike price and expiration dates. This is the alternative way to resolve strike prices and expiration dates when option chain window fails to resolve these with the data vendor.
- Option chain window can download strike prices and expiration dates from TickQuest's website. This is a very reliable way to obtain strike prices and expiration dates for options if your computer is connected to the Internet

Under most circumstances, the option database is transparent to you as a user. However, should all of the above methods fail to correctly resolve the strike price and expiration date, you can fix the option symbol by using the option editor.

Option Editor

The option editor lets you edit the expiration date and strike price of an option symbol. The changes you made are stored in the option database.

To open the option editor, right click on the option you want to change, and choose **Edit** from the pop up menu.

The option editor lets you change the following information:

Setting	Meaning
Expiration Year	The year of expiration. If you change the year, all option symbols under the same root will change to the year you specified.
Expiration Month	The month of expiration.
Strike Price	The strike price of the option.
Cycle	The option cycle of the stock. Option chain window supports CBOE cycle 1, 2 and 3. Set the option to unknown if the option cycle does not follow CBOE cycle 1, 2 or 3.
Call Option/Put Option	Whether the option is a call or a put.

Options Chain Tab in User Preference

The **Options Chain** tab in user preference lets you set option chain password.

This feature is obsoleted.

Order Interface Manager Operation Guide

This section has been moved to ***Order Interface Manager*** (see "Order Interface" on page 735).

Order Related Topics

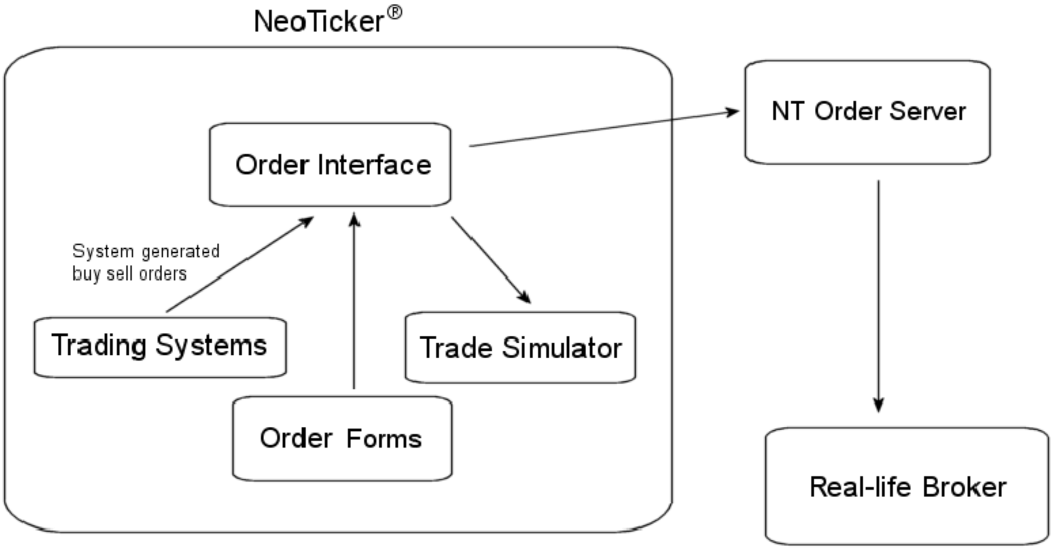
How All Order Related Items Work Together

Overview

NeoTicker® offers you a seamless transition from experimental trading (training, scenario replay, trading system testing) to real-life trading (order placement, trading system deployment).

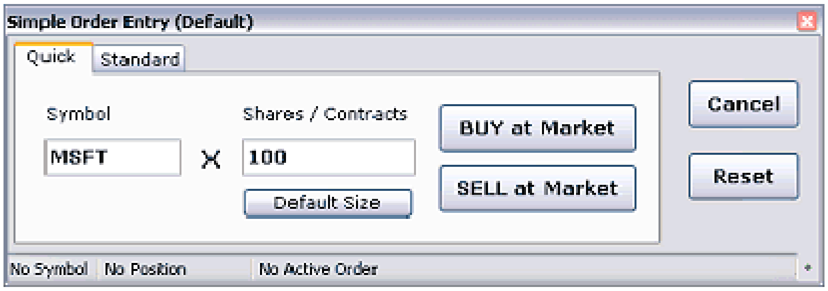
One of the key component is NeoTicker®'s comprehensive order placement and management system, which takes order generation, processing, routing and filling into account.

Below is a diagram show the flow of orders in NeoTicker®.



How Orders are Placed (Order Forms)

Order form lets you place orders manually. NeoTicker® has many types of built-in order forms so you can choose one that you are most comfortable with. Below is an example of an order form:


Simple Order Entry (Default)

Symbol: MSFT

Shares / Contracts: 100

No Symbol No Position No Active Order

As you see you can place buy/sell orders manually with order form. You can find the order forms under **Order** in main window. Related topics:

- ***Manual Order Placement*** (on page 705) for a complete discussion on order forms and manual order placement.

How Orders are Placed (Trading Systems)

Mechanical trading systems (computer programs that trade) can also place orders from within NeoTicker®. You can create trading systems with Backtest EZ indicator, formulas or scripting/programming languages.

Backtest EZ is suitable for users who want a quick and simple solution to system trading. Formulas offers more power and flexibility. Programming languages offer maximum power and flexibility. Related topics:

- ***Tutorial: Trading System Testing with Backtest EZ*** (on page 175) is a tutorial on using Backtest EZ.
- ***Formula Topics and Tutorials*** (see "Formula Topics and Tutorials" on page 255) for an overview of formulas.
- ***Creating Trading System with Formula*** (on page 311) for trading system related topics for formulas.
- ***Programming Guide*** (on page 1361) for an overview of scripting/programming.
- ***Trading System Programming Topics*** (on page 1491) for trading system related topics for scripting/programming
- ***Trade Object*** (on page 1577) is the object to use in scripts/programs. Specifically, the order placement methods are used.

If you prefer reading trading systems output and place orders manually, you can use the System Monitor to help you. In this case you will use manual order placement to place orders. Related topics:

- ***System Monitor Operation Guide*** (on page 939)

How Orders are Handled (Order Interface)

Order Interface is the central hub for sending orders between NeoTicker® components. Order Interface performs the following roles:

- Fills in any missing pieces in an order (symbol types, etc).
- Performs symbol translation (your data vendor and broker may use different symbols for the same instrument, so a translation is necessary).
- Decides who is responsible for filling the order and sends the order to the responsible party.

Related Topics:

- ***Order Interface*** (on page 735)

How Orders are Filled (NT Order Server)

If NeoTicker® is configured to send orders to your broker, Order Interface will send orders to an external plug-in called NT Order Server and NT Order Server will send the orders to your broker.

You may wonder why Order Interface does not place orders directly to your broker, but instead send the orders through NT Order Server. The reason is each broker work differently and few standards exist for order placement.

You can think of NeoTicker® and your broker speak different languages. They cannot directly communicate and NT Order Server acts an interpreter between the two. For this reason, you will need a different NT Order Server for each broker.

Related Topics:

- ***Connecting to Your Broker*** (on page 49)
- ***NT Order Server for Interactive Brokers Reference*** (on page 793)
- ***NT Order Server for MB Trading Reference*** (on page 796)

How Orders are Filled (Trade Simulator)

If NeoTicker® is not configured to send orders to your broker, Trade Simulator is responsible for simulating order fill.

Trade Simulator simulates order filling by taking tick data into account. So it offers an extremely realistic simulation of order filling in the following applications:

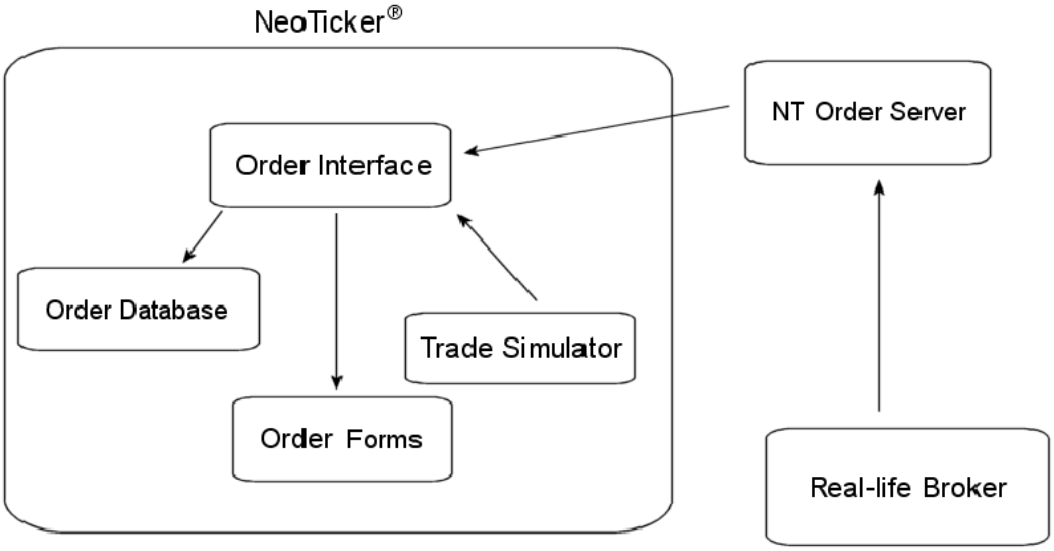
- Trading training
- Testing trading strategy in real-time
- Testing trading system in real-time

Related Topics:

- ***Trade Simulator*** (on page 755)

Order Status Feedback

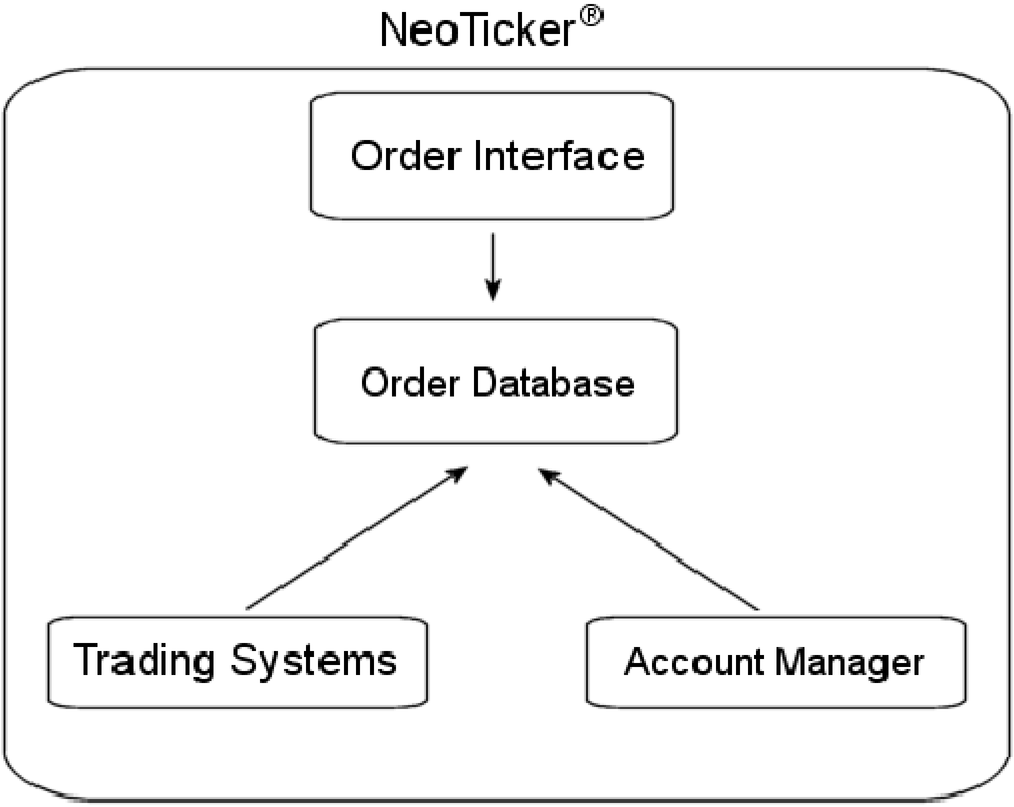
Below is a diagram showing the flow of order status once an order is filled, rejected or cancelled. It is similar to the first diagram, except the flow is reversed, i.e. from broker/Trade Simulator to Order Forms.


NeoTicker®
Order Interface
Order Database
Order Forms
Trade Simulator
NT Order Server
Real-life Broker

For example, when an order is filled by your broker, your broker will send the order status to NT Order Server, which will send the order status to Order Interface, which will in turn sends the order status to where the order originated.

Notice that we have introduced Order Database in the diagram above. Order Database stores all orders and order status for later reference and is discussed in the section below.

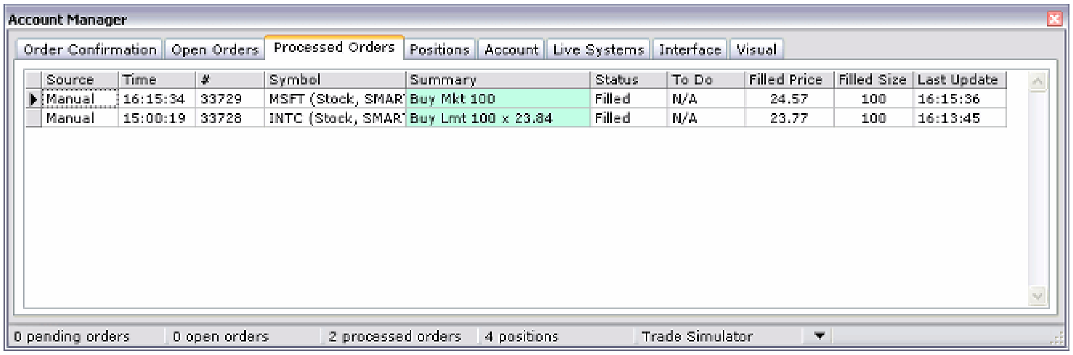
How to Audit Orders and Positions (Order Database)



When an order is placed through Order Interface, it will record the order in Order Database. When an order status is received by Order Interface, it will update Order Database.

Order Database provides a mean for you to audit all orders. This includes orders requiring confirmation, open orders and processed orders.

To access Order Database manually, you will need to use Account Manager. The figure below shows Account Manager listing processed orders.



Source	Time	#	Symbol	Summary	Status	To Do	Filled Price	Filled Size	Last Update
Manual	16:15:34	33729	MSFT (Stock, SMAR	Buy Mkt 100	Filled	N/A	24.57	100	16:15:36
Manual	15:00:19	33728	INTC (Stock, SMAR	Buy Lmt 100 x 23.84	Filled	N/A	23.77	100	16:13:45

0 pending orders | 0 open orders | 2 processed orders | 4 positions | Trade Simulator

Related topics:

- ***Account Manager*** (on page 767)

Some trading systems also need to query orders. Related topics:

- ***Trading System Programming Topics*** (on page 1491) for trading system related topics for scripting/programming
- ***Trade Object*** (on page 1577) is the object to use in scripts/programs. Specifically, the Open/Close Position objects and Transaction object are used.

Usage Scenarios

In this section, we listed many trading scenarios and the possible NeoTicker® setups to handle the scenarios.

Usage Scenario 1: Trading Training

You are new to day trading. You want to get a feel before you commit to day trading. You are busy during regular trading hour, so you want to train at your own schedule.

Setup:

- NeoTicker® connected to Simulation Server
- Order Interface sends orders to Trade Simulator

Usage:

Don't confuse Simulation Server with Trade Simulator. The former simulates a data vendor (which provides data to NeoTicker®), the latter simulates a broker (which handles orders from NeoTicker®).

With Simulation Server, you can repeatedly practice trading with the same set of data. You can run Simulation Server any time you like at your own pace, not restricted by regular trading hour.

Tick data from Simulation Server is used to simulate order filling in Trade Simulator so the experience is realistic.

Once you are ready for real-life trading, you can use the same order form you trained with to place orders to your broker.

Related topics:

- ***Simulation Server Operation Guide*** (on page 899)

Usage Scenario 2: Discretionary Trading Strategy

You are a discretionary trader. You developed a strategy that looks good on a chart. You want to know if it will work in real-time.

Setup:

- NeoTicker® connected to real-time data feed
- Order Interface sends orders to Trade Simulator

Usage:

You place order in real-time to Trade Simulator instead of your broker. Trade Simulator will simulate order filling with tick data from your real-time data feed. Because it is a simulation, no real money is involved.

At the end of the day, you can use System Performance Viewer to collect statistics about your performance to decide if the trading strategy is any good.

Related topics:

- Generating Performance Report Using System Performance Viewer
- ***System Performance Viewer*** (on page 951)

Usage Scenario 3: Discretionary Trading (Quick Reversal)

You are a discretionary trader. You only trade a single instrument. You want a way to place order quickly without too much typing.

Setup:

- NeoTicker® connected to real-time data feed
- Order Interface sends orders to your broker, with order confirmation disabled
- You use Quick Order Entry form to place orders

Usage:

You place order to Quick Order Entry form, which has buttons to help you place orders with minimal number of clicks.

Your orders are sent to broker without confirmation. So you can place order as quickly as possible.

Usage Scenario 4: Discretionary Trading (Fast Order Entry)

You are a discretionary trader. You require order entry for multiple symbols.

Setup:

- NeoTicker® connected to real-time data feed
- Order Interface sends orders to your broker, with order confirmation disabled
- You use Compact Order Entry form to place orders

Usage:

You place order to Compact Order Entry form, which is designed to minimize screen space usage. You can create multiple Compact Order Entry form, one form for each symbol to place orders, without cluttering your desktop.

Your orders are sent to broker without confirmation. So you can place order as quickly as possible.

Usage Scenario 5: Replaying a Market Scenario

You are a seasoned discretionary trading with a good trading strategy. But today is a bad day and your strategy does not quite work. You want to analyze what went wrong.

Setup:

- NeoTicker® connected to Simulation Server
- Order Interface sends orders to Trade Simulator

Usage:

Don't confuse Simulation Server with Trade Simulator. The former simulates a data vendor (which provides data to NeoTicker®), the latter simulates a broker (which handles orders from NeoTicker®).

At end-of-day, you download today's tick data and replay the tick data with Simulation Server. You can adjust simulation speed so you can go through the whole trading day in a short period of time.

Trade Simulator uses tick data from Simulation Server to simulator order filling. This provides a very good approximation to price you can get in real-life.

Related topics:

- ***Simulation Server Operation Guide*** (on page 899)

Usage Scenario 6: Real-Time Testing of Trading System

You have a trading system that performs well using traditional back testing. You want to see how well it performs in real-time, with tick-by-tick data.

Setup:

- NeoTicker® connected to real-time data feed
- Order Interface sends orders to Trade Simulator, with order confirmation disabled

Usage:

Your trading system will work as if it is connected to a real-life broker. Tick data from your real-time data feed is used to simulate order filling. At the end-of-day, you can use System Performance Viewer to collect statistics and review performance.

The workflow is exactly the same as if you are connecting to a real-life broker, except no real money is involved. Trading performance will be a much closer approximation to real-life performance than traditional back testing.

Usage Scenario 7: Life Deployment of Trading System (Automatic Order Entry)

You have a trading system you want to deploy in real-life. The system fires quite frequently so you want the order to send to your broker without user intervention.

Setup:

- NeoTicker® connected to real-time data feed
- Order Interface sends orders to your broker, with order confirmation disabled

Usage:

This is an automatic setup. When your trading system fires an order, it is sent directly to your broker without asking any question.

Usage Scenario 8: Life Deployment of Trading System (Manual Order Entry)

You have a trading system you want to deploy in real-life. You prefer to review each order before sending it to your broker.

Setup:

- NeoTicker® connected to real-time data feed
- Order Interface sends orders to your broker, with order confirmation enabled

Usage:

This is a semi-automatic setup. When your trading system fires an order, Order Interface will prepare the order for you to review before sending it to the broker. You can review the orders in Account Manager and decide whether to actually send the order.

Manual Order Placement

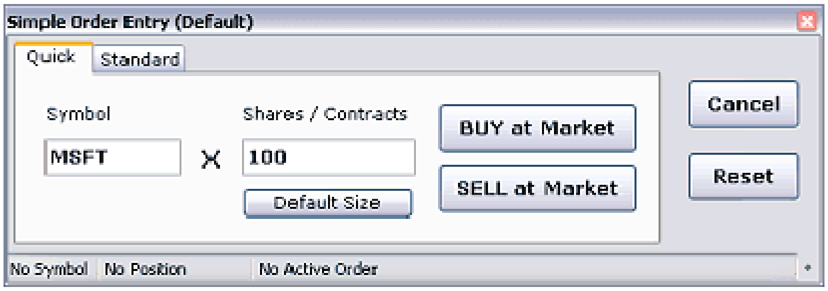
General Usage

You place orders using order forms in NeoTicker®.

You have a choice for different order forms to use. Each form is designed to suit a particular style of trading. You can open multiple forms.

You can access the order forms under the **Order** menu in main window.

Below is an example of the default ***Simple Order Entry Form*** (on page 709).

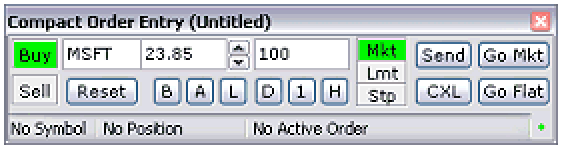


Simple Order Entry (Default)

Symbol MSFT × Shares / Contracts 100

No Symbol No Position No Active Order

Another example is the ***Compact Order Entry Form*** (on page 710).

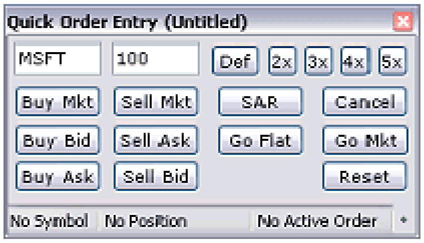


Compact Order Entry (Untitled)

MSFT 23.85 100

No Symbol No Position No Active Order

Another example is the ***Quick Order Entry Form*** (on page 712).



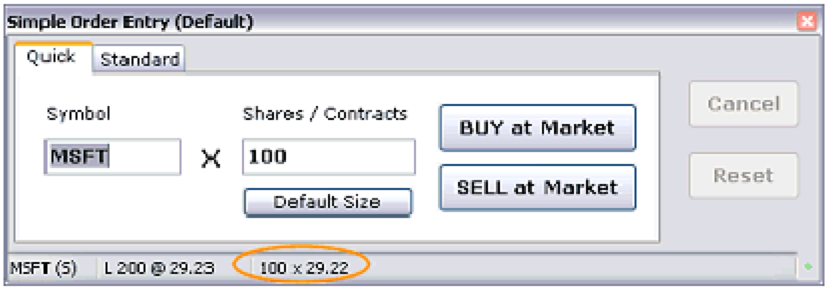
Quick Order Entry (Untitled)

MSFT 100

No Symbol No Position No Active Order

Form Order Relationship

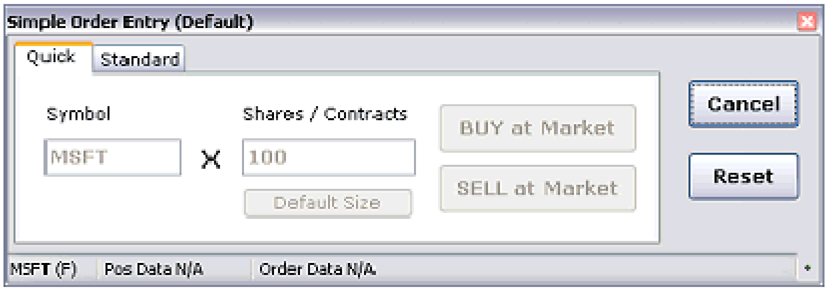
An order form is connected to the order it places. This lets the order form to report order status.



Symbol MSFT × Shares / Contracts 100

MSFT (S) L 200 @ 29.23 100 x 29.22

Because the order form is connected to the order, once you place an order, you cannot use the order form until the order status comes back (except to cancel the order). Below shows a form after an order is placed. You can see that most controls are disabled.



Symbol MSFT × Shares / Contracts 100

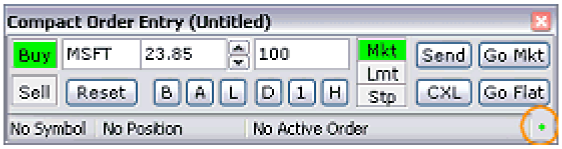
MSFT (F) Pos Data N/A Order Data N/A

You can break the connection between order form and order by pressing **Reset** button. Once you break the connection, you can place order again in the form.

Order Destination - Broker or Trade Simulator

You can quickly tell where the order will be sent by looking at the lower right corner of an order form.

There is a dot there. If the dot is green, the order will be sent to your broker. If the dot is red, the order will be sent to Trade Simulator.



MSFT 23.85 100

No Symbol No Position No Active Order

Default Size

In all order form, there is a **Default Size** button. Based on the symbol type (stocks, futures, etc), pressing the **Default Size** button can fill an order size and saves you some typing.

For example, usually stocks have a default size of 100, whereas futures have a default size of 1.

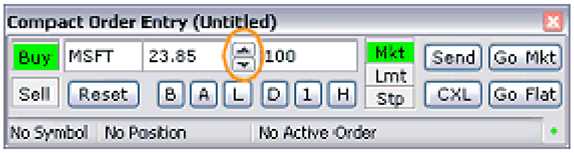
Default Size button requests a default order size from ***Symbol Info Manager*** (on page 921).

Bid Ask Price Buttons

Some order forms have bid ask buttons. Pressing the button will get the current bid ask price of the symbol and fill the bid ask price in the price field.

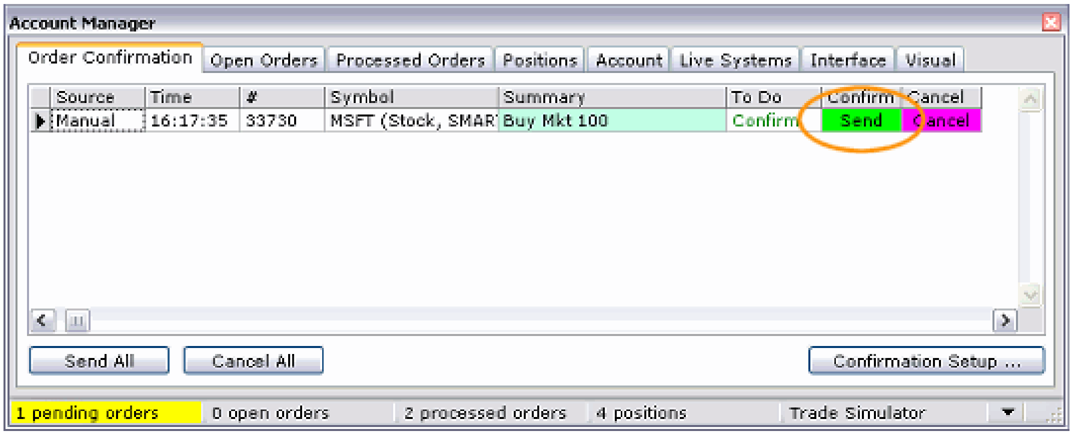
Price Adjustment

In price field, you can use the up down arrow keys to adjust the price without typing. You can also use the mouse wheel to adjust price.



Order Confirmation

By default, after you place an order, you need to confirm it in Account Manager before the order will be sent. Account Manager is opened automatically if confirmation is required. Press the **Send** button under **Confirm** column in Account Manager to confirm the order.



Source	Time	#	Symbol	Summary	To Do	Confirm	Cancel
Manual	16:17:35	33730	MSFT (Stock, SMAR	Buy Mkt 100	Confirm	Send	Cancel

You can disable order confirmation. When order confirmation is disabled, after you place an order, it is sent directly to your broker or Trade Simulator to process. See ***Enabling/Disabling Order Confirmation*** (on page 733).

Cancelling an Order

In all order forms, there is a **Cancel** button. The **Cancel** button is enabled after you place an order.

Pressing the **Cancel** button will send a cancel message to your broker/Trade Simulator for the order.

Cancelling is not always successful. Your broker/Trade Simulator can refuse a cancellation request for many reasons. Most common reason is the order is already filled.

Another scenario is if the order is partially filled. Usually a cancellation request will stop the broker/Trade Simulator from completing the order.

If you cancel an order, you should always use Account Manager to review the order.

Hot Track

Hot Track is a feature linking symbols to order forms. When you click a symbol in a function window (e.g. chart, quote window, etc), order form will automatically fill in the symbol and default order size for you.

If you want to disable Hot Track, right click on the order form to open pop up menu, and disable **Hot Track**.

If you want Hot Track, but do not want to fill in the default size, right click on the order form to open pop up menu, and disable **Auto Size**.

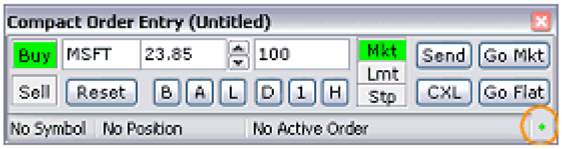
Overriding Order Routing to Trade Simulator

By default, an order form will route orders according to the settings in Order Interface Setup.

In Order Interface Setup, if you have order routing set to your broker, but you want a specific order form to place order to Trade Simulator, you can override the order routing. In the order form, right click on the order form to open pop up menu, and enable **Route Orders To Trade Simulator**.

Shortcut to Account Manager

If available, you can open ***Account Manager*** (on page 767) by clicking the dot at lower right corner of the order form.


Compact Order Entry (Untitled)

MSFT 23.85 100

No Symbol No Position No Active Order

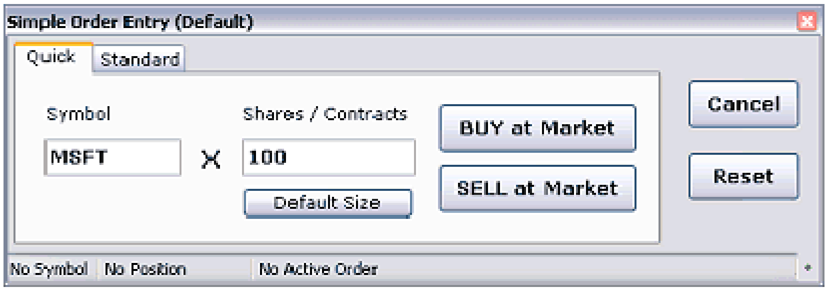
Simple Order Entry Form

Simple Order Entry Form is the default order form. It is designed to be easy to use. It provides the basic order commands.

You can open Simple Entry Order form by choosing **Order>Simple Order Entry** in the main window.

To open additional Simple Order Entry Form, choose **Order>New Order Form>Simple Order Entry**.

Quick Tab



Simple Order Entry (Default)

Symbol: MSFT

Shares / Contracts: 100

No Symbol No Position No Active Order

Under **Quick** tab are the controls to place market orders. To place an order:

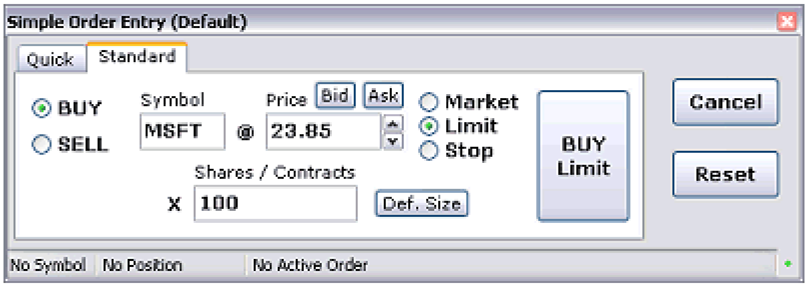
1 Enter symbol.

2 Enter number of shares/contract.

3 Press **BUY at Market** or **SELL at Market.**

4 You may need to confirm sending the order in Account Manager.

Standard Tab



Simple Order Entry (Default)

● BUY
○ SELL

Symbol: MSFT @ Price: 23.85

● Limit

Shares / Contracts: x 100

No Symbol | No Position | No Active Order

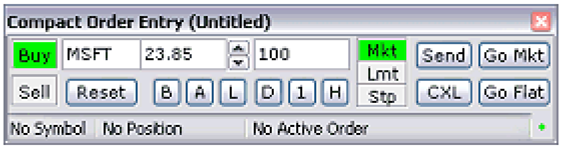
Under **Standard** tab are the controls to place market, limit and stop orders. To place an order:

1. Choose **BUY** or **SELL**.
2. Enter **Symbol**.
3. Enter **Price** if you want to place a limit or stop order.
4. Choose **Market**, **Limit** or **Stop**.
5. Enter number of shares/contracts under **Shares/Contracts**.
6. Press the **BUY** or **SELL** button.

Compact Order Entry Form

Compact Order Entry Form is designed to be small. So you can put many Compact Order Entry Forms on your desktop.

To open Compact Order Entry Form, in main window, choose **Order>New Order Form>Compact Order Entry**.



Compact Order Entry (Untitled)

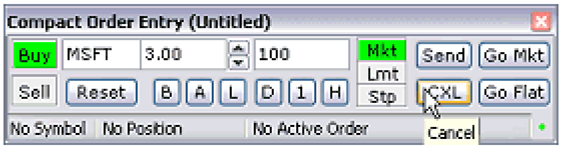
Buy | MSFT | 23.85 | 100 | Mkt

No Symbol | No Position | No Active Order

In order to save space, Compact Order Form uses short form for controls.

Tool Tips

If you let your mouse cursor rest on a control, a tool tip will come out telling you what the control is.



Compact Order Entry (Untitled)

Buy | MSFT | 3.00 | 100 | Mkt

No Symbol | No Position | No Active Order

Placing Order

1 Press **BUY** or **SELL**.

2 Enter symbol in the left field.

3 Enter **Price** if you want to place a limit or stop order in the middle field.

4 Enter number of shares/contracts in the right field.

5 Press **Mkt**, **Lmt** or **Stp** for market, limit or stop order respectively.

6 Press **Send** button to send order.

7 You may need to confirm sending the order in Account Manager.

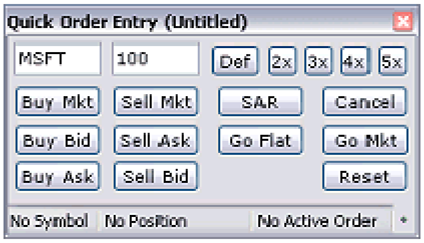
Reference to Short Forms

Short Form	Meaning
B	Fill bid price to price field
A	Fill ask price to price field
L	Fill last price to price field
D	Fill default size to shares/contract field
1	Fill the number 1 to shares/contract field
H	Fill the number 100 to shares/contract field
Mkt	Set order to market order
Lmt	Set order to limit order
Stp	Set order to stop order
CXL	Cancel button
Go Mkt	Turn a limit/stop order into market order
Go Flat	If currently holding a position, go flat

Quick Order Entry Form

Quick Order Form is designed for fast order entry. Shortcuts are available to help you place orders with minimal number of clicks.

To open Quick Order Entry Form, in main window, choose **Order>New Order Form>Quick Order Entry**.



In Quick Order Entry, you do not type in price. You either place market orders, or the price is auto filled in by bid/ask price.

To place an order, press one of the following buttons (note that there is no need to press a send button):

- Buy Mkt
- Sell Mkt
- Buy Bid
- Sell Ask
- Buy Ask
- Sell Bid

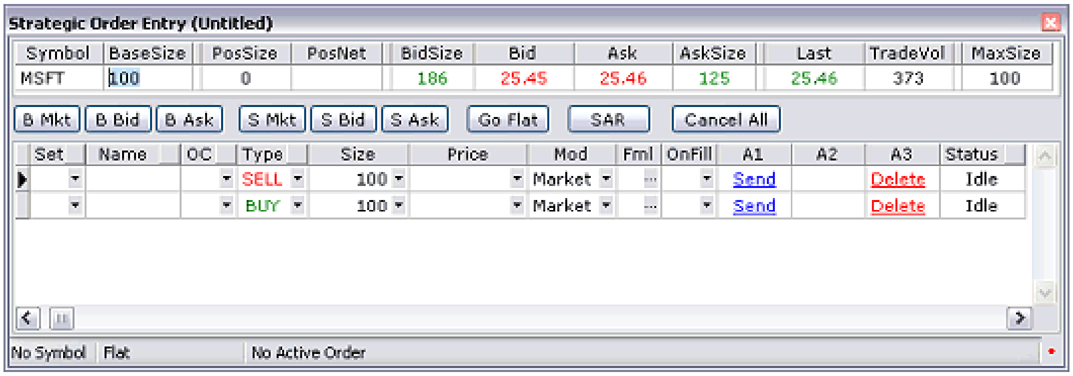
Table below shows what each button does:

Button	Action
Def	Fill default size to the right field
2x, 3x, 4x, 5x	Fill a multiple of default size to the right field
Buy Mkt	Send a buy at market order
Sell Mkt	Send a sell at market order
Buy Bid	Send a limit order to buy at bid
Sell Ask	Send a limit order to sell at ask
Buy Ask	Send a limit order to buy at ask
Sell Bid	Send a limit order to sell at bid
SAR	Stop And Reversal. Send market order to reverse position, e.g. if you are currently long 200 shares, this button will send orders to sell 400 shares. Result is short 200 shares.
Go Flat	Go flat on current position.
Go Mkt	Turn current limit order to market order

Strategic Order Entry Form

Basic Concept

Strategic Order Entry Form is designed to let you handle multiple orders with ease.



Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last	TradeVol	MaxSize
MSFT	100	0		186	25.45	25.46	125	25.46	373	100

Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A2	A3	Status
			SELL	100		Market	...		Send		Delete	Idle
			BUY	100		Market	...		Send		Delete	Idle

No Symbol Flat No Active Order

Above figure is a Strategic Order Entry Form. When using Strategic Order Entry Form, you will manage a list of orders that are centered around a single symbol. Features are provided so you can quickly enter and modify orders.

With Strategic Order Form, you can:

- Create a list of orders, waiting to be sent to your broker at the right moment.
- Have order triggering orders
- Have order cancelling orders
- Order decision can be based on formula calculations, including indicator calculations

In another words, Strategic Order Entry form is a tool to execute complex trading strategies for discretionary trading.

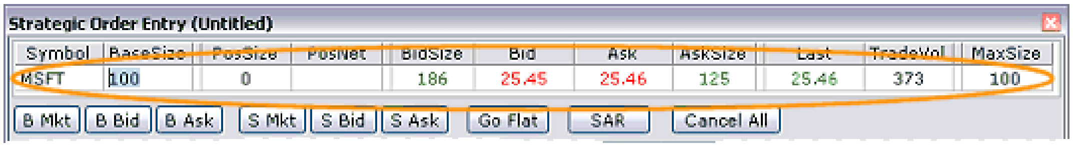
Before You Use Strategic Order Entry Form

- You should be familiar with order entry in NeoTicker® in general. Strategic Order Entry Form is an advanced order entry tool with numerous options. Your learning curve will be steep if you start with Strategic Order Entry Form.
- When you use Strategic Order Entry Form, you should consider having order confirmation turn off. Because you will be issuing multiple orders, having to confirm each order can become tiring. In this section, it is assumed confirmation has been turned off.
- It is strongly recommended you use Trade Simulator to practice when you are learning to use Strategic Order Entry Form.

Setup

To open Strategic Order Entry Form, in main window, choose **Order>New Order Form>Strategic Order Entry**.

On the top of the form is an area displaying various information. This area provides a summary of the symbol, position, current quote information, etc.



Strategic Order Entry (Untitled)

Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last	TradeVol	MaxSize
MSFT	100	0		186	25.45	25.46	125	25.46	373	100

You should set up the top area first before you use Strategic Order Entry Form. There are three things to set up:

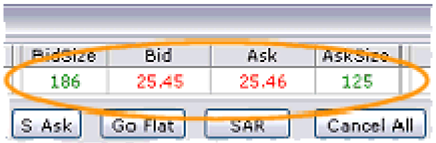
- Type in a symbol
- Check the value under the **BaseSize**. This value is used for the default order size when you place an order. Usually Strategic Order Entry Form will fill this value in correctly for you. If not, you should adjust this value. For example, you should set this value to 100 for stocks and 1 for futures.
- **Max Size** is a value that limits your position to a maximum size. Strategic Order Entry Form will not send any order to your broker if your current position has reached Max Size. Max Size protects you against accidentally getting into too large of a position. You should set this value to something your trading account can handle.

Tip: The Max Size value will turn yellow if you attempt to place an order that violates Max Size limitation

Placing Orders

There are two methods to place orders:

Method 1: Press the value under **Bidsize**, **Bid**, **Ask** or **AskSize** columns.

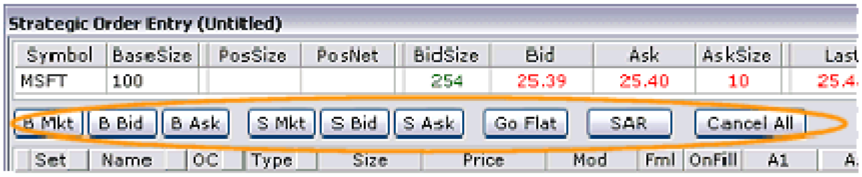


BidSize	Bid	Ask	AskSize
186	25.45	25.46	125

Pressing one of these values will create a sell or buy order in the order list (**BidSize** and **Bid** for sell orders; **Ask** and **AskSize** for buy orders). You can then modify various attributes of the order. The order will not be sent to your broker until you explicitly send the orders. This is the method to use for creating multiple orders to form a trading strategy.

Method 2: Press the quick buttons

Pressing the quick buttons will generate an order that will be sent to your broker right away. The purpose of the quick buttons is to quickly adjust your position.



Strategic Order Entry (Untitled)

Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last
MSFT	100			254	25.39	25.40	10	25.4

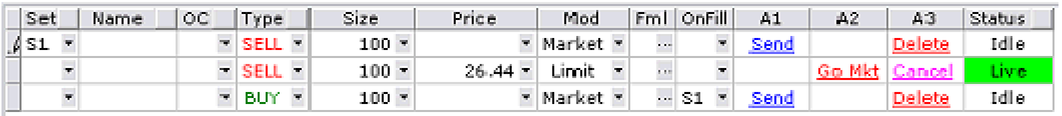
Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A.

The table below summarizes quick button actions.

Quick Button	Action
B Mkt	Send a buy at market order to your broker.
B Bid	Send a limit order to buy at bid price to your broker.
B Ask	Send a limit order to buy at ask price to your broker.
S Mkt	Send a sell at market order to your broker.
S Bid	Send a limit order to sell at bid price to your broker.
S Ask	Send a limit order to sell at ask price to your broker.
Go Flat	Send a market order to buy/sell in order to go flat on your position.
SAR	Send a market order to buy/sell to reverse your position (e.g. from long 100 to short 100).
Cancel All	Cancel all open orders.

Orders

At the bottom of Strategic Order Form is a list of orders that you can manage.



Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A2	A3	Status
S1			SELL	100		Market	...		Send		Delete	Idle
			SELL	100	26.44	Limit	...			Go Mkt	Cancel	Live
			BUY	100		Market	...	S1	Send		Delete	Idle

The value under **Status** column shows the current status of orders. In general, you can think of orders having two status: Idle and Live.

Idle orders are created when you press the value under **Bidsize**, **Bid**, **Ask** or **AskSize** columns. Idle orders have not been sent to your broker yet, so you can make all kinds of changes to the orders. Once you think the order is ready, you need press **Send** to send the order to your broker.

Once an order has been sent to your broker, its status becomes Live. A live order is subjected to fill by your broker. You can only make limited changes to a live order. And when a change is made, Strategic Order Entry Form will notify your broker to carry out the change (e.g. cancelling an order, changing price in limited order).

So when you are using Strategic Order Entry Form, you are creating a list of idle orders, filling in the order information and sending them to your broker for execution. Once an order becomes live, Strategic Order Entry Form will provide tools to help you manage the order.

There is a also an Armed status, which is a specialized status for orders waiting to be triggered by a condition (e.g. RSI < 30). For more information, see ***Strategic Order Entry Form, Usage Example: Using Formula to Trigger Order*** (see "Usage Example: Using Formula to Trigger Order" on page 725).

Once your broker fills the order, Strategic Order Entry Form will do the following:

- Removing the order from the order list if the order is plain, i.e. it does not have any settings in **Set**, **Name**, **Fml** and **OnFill**.
- Keeping the order in the order list if the order has settings mentioned above. The order will become idle again.

The table below summarizes the order fields. If you are new to Strategic Order Entry Form, you should read one of the usage sections to gain understanding on how these fields are used.

Field	Meaning
Set	The set in which the order belongs to. This field is used for on fill triggering. See ***Strategic Order Entry Form, Usage Example: Order Triggering a Set of Orders*** (see "Usage Example: Order Triggering a Set of Orders" on page 721).
Name	Order name. This is user set field to help you organize Strategic Order Entry Form. Note that once this field is set, the order is no longer considered a plain order and will not be removed automatically after fill.
OC	The field is used for order cancelling orders. See ***Strategic Order Entry Form, Usage Example: Order Cancelling Orders*** (see "Usage Example: Order Cancelling Orders" on page 723).
Type	Order type - buy or sell.

Size Order size.

Price Order price. This field is applicable for limit and stop orders only.

Mod Modifier to the order - Market, Limit and Stop. For Market, Limit and Stop orders triggered by formula, see ***Strategic Order Entry Form, Usage Example: Using Formula to Trigger Order*** (see "Usage Example: Using Formula to Trigger Order" on page 725).

Fml Formula for triggering orders. See ***Strategic Order Entry Form, Usage Example: Using Formula to Trigger Order*** (see "Usage Example: Using Formula to Trigger Order" on page 725).

OnFill Order set to trigger when the order is filled. See ***Strategic Order Entry Form, Usage Example: Order Triggering a Set of Orders*** (see "Usage Example: Order Triggering a Set of Orders" on page 721).

A1 Action. Clicking on a cell under this column will carry out an action. Usually **A1** is for sending order to broker and arming a formula order.

A2 Action. Clicking on a cell under this column will carry out an action. Usually **A2** is for converting limit and stop orders to market.

A3 Action. Clicking on a cell under this column will carry out an action. Usually **A3** is for deleting/cancelling/unarming orders.

Status Order status - Idle, Live or Armed.

Trouble Shooting

Below is a list of things to check for common problems encountered when using Strategic Order Entry Form:

Symptom	Check
Orders are not sent	Check if Max Size is reached.
Orders are not sent	Check if confirmation setting in Order Interface Setup is set to on.
Orders are not removed	Check if the order is not plain, i.e. it has **Set**, **Name**, **Fml** or **OnFill** settings. Also in pop up menu, if auto remove plain order is set to off.
Unable to cancel order	Order could have become stale. This can be caused by a lost connection to a live broker. Use Account Manager to mark order as cancelled.
Formula orders are not sent	Check if formula is correct. You can use a quote window to help you evaluate formula.

Orders are sent multiple times	Check if order **Mod** is set to MktCyc, LmtCyc, StpCyc. These orders are supposed to be sent multiple times.

Usage Example: Having Multiple Orders Waiting for Opportunity

In this example, we will set up multiple buy sell orders at different limit prices. You can then use discretion to decide when to enter which order. In addition, we will set up the form so that these orders are not removed after fill. So that you can reuse them without re-entering the order.

1 In a new Strategic Order Entry Form, enter MSFT for **Symbol**, 100 for **BaseSize**, 500 for **MaxSize**.

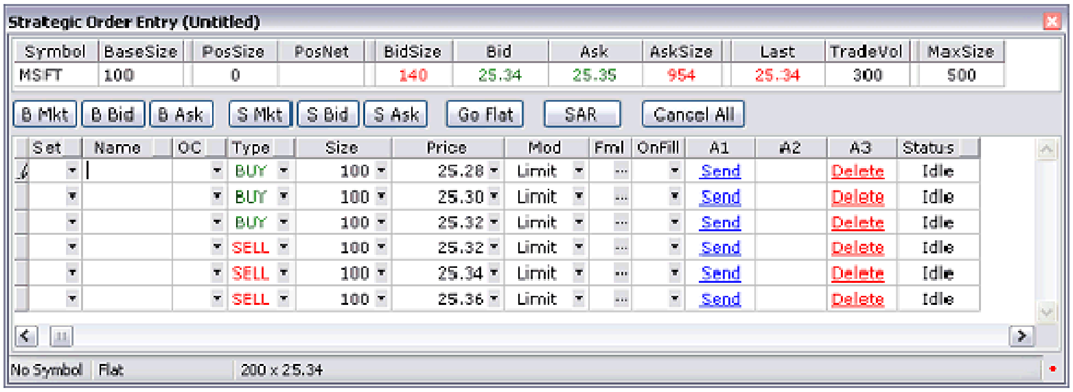
2 Press the value under **Ask** 3 times. This will create 3 buy orders.

3 Press the value under **Bid** 3 times. This will create 3 sell orders.

4 Modify the orders to limit at price close to MSFT's trading price. You can use the price drop down menu to help you fill in a price quickly.

5 Right click on Strategic Order Entry Form and turn off **Auto Remove Plain Orders**.

Strategic Order Entry Form will look something like:

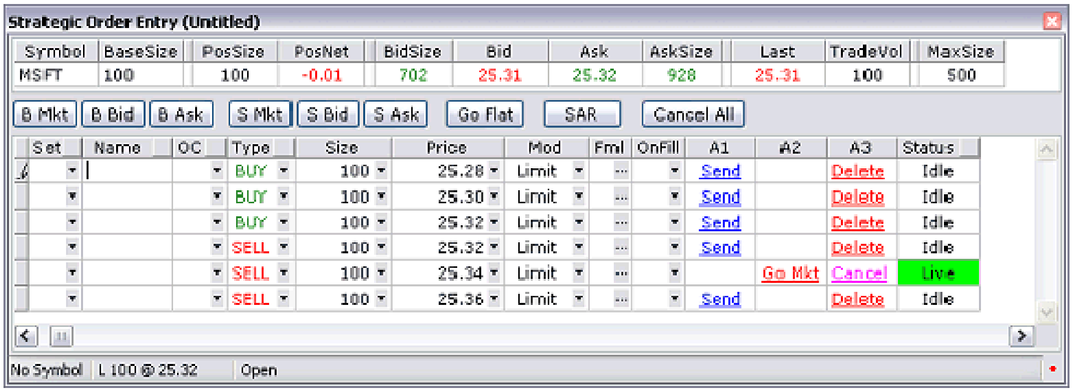


Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last	TradeVol	MaxSize
MSFT	100	0		140	25.34	25.35	954	25.34	300	500

Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A2	A3	Status
			BUY	100	25.28	Limit	...		Send		Delete	Idle
			BUY	100	25.30	Limit	...		Send		Delete	Idle
			BUY	100	25.32	Limit	...		Send		Delete	Idle
			SELL	100	25.32	Limit	...		Send		Delete	Idle
			SELL	100	25.34	Limit	...		Send		Delete	Idle
			SELL	100	25.36	Limit	...		Send		Delete	Idle

No Symbol | Flat | 200 x 25.34

After you press **Send**, the corresponding order will be sent to your broker and becomes live.

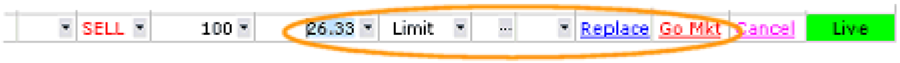


Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last	TradeVol	MaxSize
MSFT	100	100	-0.01	702	25.31	25.32	928	25.31	100	500

Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A2	A3	Status
			BUY	100	25.28	Limit	...		Send		Delete	Idle
			BUY	100	25.30	Limit	...		Send		Delete	Idle
			BUY	100	25.32	Limit	...		Send		Delete	Idle
			SELL	100	25.32	Limit	...		Send		Delete	Idle
			SELL	100	25.34	Limit	...			Go Mkt	Cancel	Live
			SELL	100	25.36	Limit	...		Send		Delete	Idle

No Symbol | L 100 @ 25.32 | Open

Notice that once an order becomes live, you can press **Go Mkt** to turn it into a market order. You can also change the price of a limit order. Then press **Replace** to update the order price.



Because you have turned off **Auto Remove Plain Orders**, the orders are not removed after your broker fills the orders or you cancel them.

For learning purpose, we recommend you to use Trade Simulator to test this set up before using a real account.

Usage Example: Order Triggering a Set of Orders

In this example, we will set up a buy order that will triggers a set of sell at limit orders. In addition, we will set up the form so that these orders are not removed after fill. So that you can reuse them.

1 In a new Strategic Order Entry Form, enter MSFT for **Symbol**, 100 for **BaseSize**, 500 for **MaxSize**.

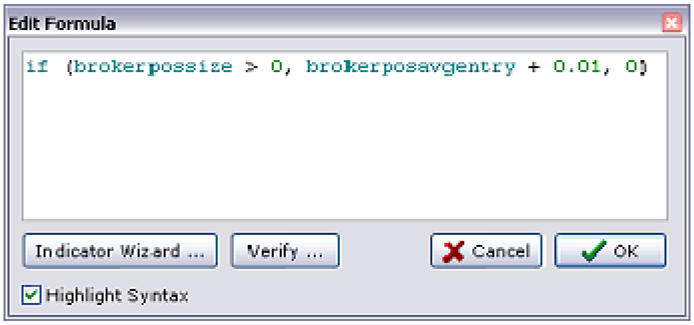
2 Press the value under **Ask.** This will create a buy order.

3 Modify the buy order. Enter 200 for the order size. Keep the buy order as a market order. Under the **OnFill** column, set the value to S1.

4 Press the value under **Bid**. This will create a sell order.

5 Modify the sell order. Under the **Set** column, change the value to S1. Keep the order size at 100. Under the **Mod** column, change value to LmtFml. Under the **Fml** column, press the ... button to open a formula editor, and enter the following formula:

```
if (brokerpossize > 0, brokerposavgentry + 0.01, 0)
```



```
if (brokerpossize > 0, brokerposavgentry + 0.01, 0)
```

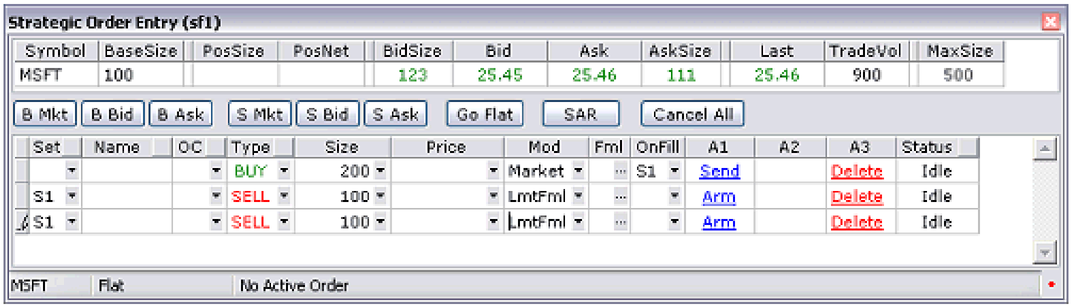
6 Press the value under **Bid** again. This will create another sell order.

7 Modify the second sell order. Under the **Set** column, change the value to S1. Keep the order size at 100. Under the **Mod** column, change value to LmtFml. Under the **Fml** column, press the ... button to open a formula editor, and enter the following formula:

```
if (brokerpossize > 0, brokerposavgentry + 0.02, 0)
```

8 Right click on Strategic Order Entry Form and turn off **Auto Remove Plain Orders**.

Strategic Order Entry Form will look something like:



Strategic Order Entry (sf1)

Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last	TradeVol	MaxSize
MSFT	100			123	25.45	25.46	111	25.46	900	500

Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A2	A3	Status
			BUY	200		Market	...	S1	Send		Delete	Idle
S1			SELL	100		LmtFml	...		Arm		Delete	Idle
S1			SELL	100		LmtFml	...		Arm		Delete	Idle

MSFT Flat No Active Order

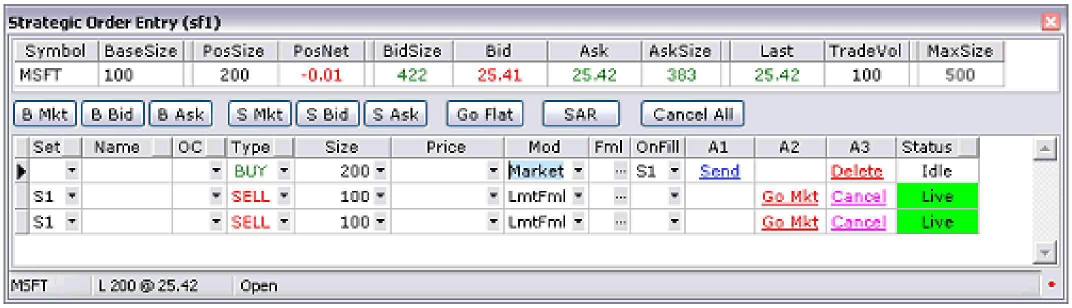
Let's explain the three orders you've entered.

First, the buy order is pretty simple. It is a buy at market order for 200 shares of MSFT. We are using this order as a trigger. The **OnFill** column specifies the set of orders to trigger once the buy order is filled. In this example, it is the order set S1.

As for the two sell orders, let's not worry about the meaning of the formula now. All you need to know is they will fill in a limit price that is $0.01 and $0.02 above the price the buy order is filled. The important thing is both sell orders belong to the set S1 (under the column **Set**). This means when the buy order is filled, both sell orders will be triggered.

In another words, the three orders combined has the following logic: buy 200 shares of MSFT at market, once the order is filled, send one limit order to sell at $0.01 above the price MSFT was bought and one limit order to sell at $0.02 above.

When you are using this type of setup, all you have to do is press **Send** at the right moment to capture a trading opportunity. Once the buy order is filled, Strategic Order Entry Form will send out both sell orders.



Strategic Order Entry (sf1)

Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last	TradeVol	MaxSize
MSFT	100	200	-0.01	422	25.41	25.42	383	25.42	100	500

Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A2	A3	Status
			BUY	200		Market	...	S1	Send		Delete	Idle
S1			SELL	100		LmtFml	...			Go Mkt	Cancel	Live
S1			SELL	100		LmtFml	...			Go Mkt	Cancel	Live

MSFT L 200 @ 25.42 Open

Because you have turned off **Auto Remove Plain Orders**, the orders are not removed after your broker fills the orders or you cancel them.

For learning purpose, we recommend you to use Trade Simulator to test this set up before using a real account.

Usage Example: Order Cancelling Orders

In this example, we will set up a buy order that will triggers a sell at limit order and a sell stop order. In this set up, the limit order is for capturing profit, and the stop order is for protecting capital. Because only one of these orders should be filled, we will set up Strategic Order Entry Form such that when one order is filled, the other will be cancelled.

In addition, we will set up the form so that these orders are not removed after fill. So that you can reuse them.

1 In a new Strategic Order Entry Form, enter MSFT for **Symbol**, 100 for **BaseSize**, 500 for **MaxSize**.

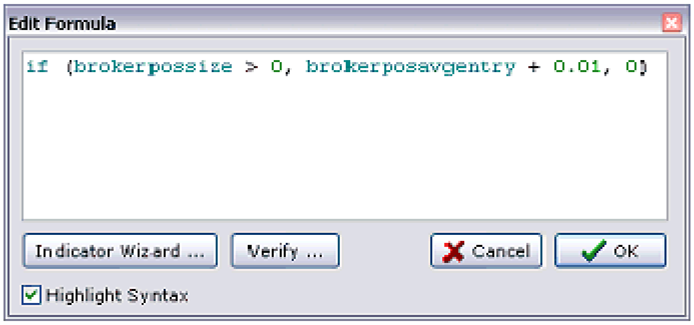
2 Press the value under **Ask.** This will create a buy order.

3 Modify the buy order. Enter 200 for the order size. Keep the buy order as a market order. Under the **OnFill** column, set the value to S1.

4 Press the value under **Bid**. This will create a sell order.

5 Modify the sell order. Under the **Set** column, change the value to S1. Under the **OC** column, change the value to A. Change the order size to 200. Under the **Mod** column, change value to LmtFml. Under the **Fml** column, press the ... button to open a formula editor, and enter the following formula:

```
if (brokerpossize > 0, brokerposavgentry + 0.01, 0)
```



Edit Formula

```
if (brokerpossize > 0, brokerposavgentry + 0.01, 0)
```

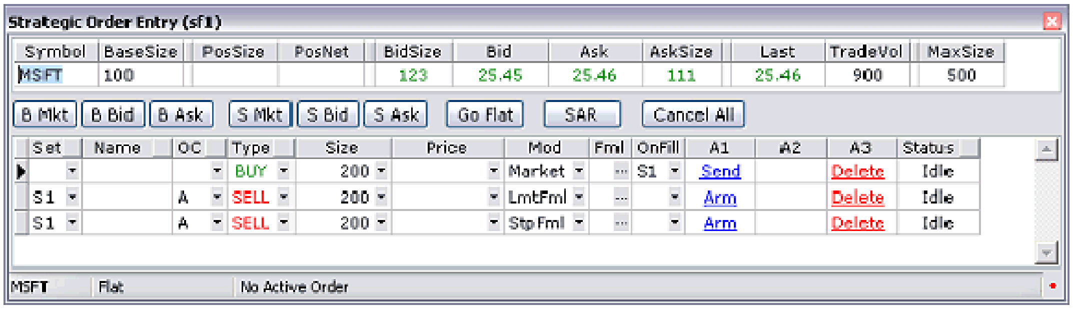
6 Press the value under **Bid** again. This will create another sell order.

7 Modify the second sell order. Under the **Set** column, change the value to S1. Under the **OC** column, change the value to A. Change the order size to 200. Under the **Mod** column, change value to StpFml. Under the **Fml** column, press the ... button to open a formula editor, and enter the following formula:

```
if (brokerpossize > 0, brokerposavgentry - 0.03, 0)
```

8 Right click on Strategic Order Entry Form and turn off **Auto Remove Plain Orders**.

Strategic Order Entry Form will look something like:



Strategic Order Entry (sf1)

Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last	TradeVol	MaxSize
MSFT	100			123	25.45	25.46	111	25.46	900	500

Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A2	A3	Status
			BUY	200		Market	...	S1	Send		Delete	Idle
S1		A	SELL	200		LmtFml	...		Arm		Delete	Idle
S1		A	SELL	200		StpFml	...		Arm		Delete	Idle

MSFT | Flat | No Active Order

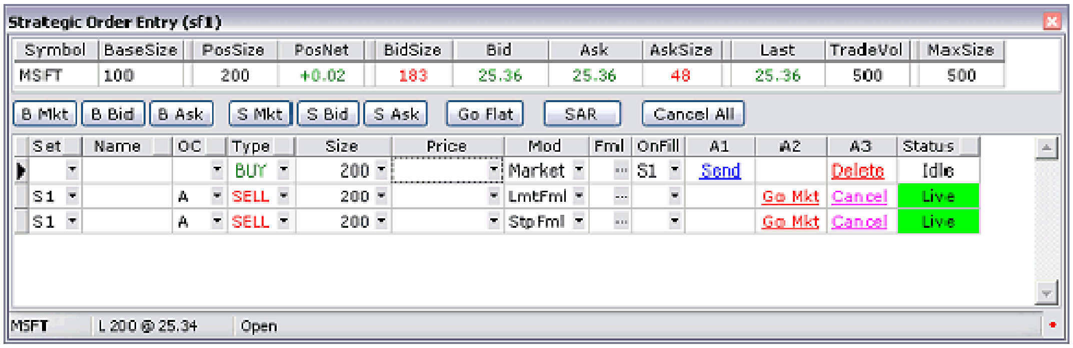
Let's explain the three orders you've entered.

First, the buy order is pretty simple. It is a buy at market order for 200 shares of MSFT. Once filled, this buy order will trigger the sell limit and sell stop orders (for more information on order triggering, see ***Strategic Order Entry Form, Usage Example: Order Triggering a Set of Orders*** (see "Usage Example: Order Triggering a Set of Orders" on page 721)).

As for the two sell orders, let's not worry about the meaning of the formula now. All you need to know is one has a limit price at $0.01 above the buy price and one has a sell stop price at $0.03 below the buy price. The important thing is both sell orders have **OC** set to A. This means when one order is filled, all orders have **OC** set to A will be cancelled.

In another words, the three orders combined has the following logic: buy 200 shares of MSFT at market, once the order is filled, send one limit order to sell at $0.01 above the price MSFT was bought and one stop order to sell at $0.03 below. If one of the sell order is filled, the rest of the orders will be cancelled.

When you are using this type of setup, all you have to do is press **Send** at the right moment to capture a trading opportunity. Once the buy order is filled, Strategic Order Entry Form will send out both sell orders to your broker.



Strategic Order Entry (sf1)

Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last	TradeVol	MaxSize
MSFT	100	200	+0.02	183	25.36	25.36	48	25.36	500	500

Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A2	A3	Status
			BUY	200		Market	...	S1	Send		Delete	Idle
S1		A	SELL	200		LmtFml	...			Go Mkt	Cancel	Live
S1		A	SELL	200		StpFml	...			Go Mkt	Cancel	Live

MSFT | L 200 @ 25.34 | Open

Because you have turned off **Auto Remove Plain Orders**, the orders are not removed after your broker fills the orders or you cancel them.

For learning purpose, we recommend you to use Trade Simulator to test this set up before using a real account.

Usage Example: Using Formula to Trigger Order

In previous examples, we've touched lightly the use of formula in Strategic Order Entry Form.

Here we will provide two examples of using formula. The first one is a sent-once order. When a formula condition becomes true, the order is sent to your broker. The second example is an order automatically managed by formula. Not only the order is sent when formula condition becomes true, it will be re-sent when the condition becomes true again.

We suggest you to quickly browse through ***Strategic Order Entry Form, Formulas in Strategic Order Entry Form*** (see "Formulas in Strategic Order Entry Form" on page 728) to get a basic understanding of formula in Strategic Order Entry Form.

Example: Long when RSI below 30

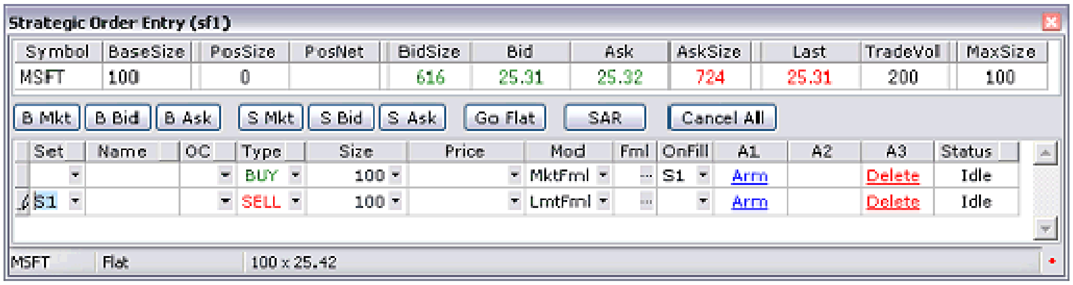
1. In a new Strategic Order Entry Form, enter MSFT for **Symbol**, 100 for **BaseSize**, 500 for **MaxSize**.
2. Press the value under **Ask.** This will create a buy order.
3. Modify the buy order. Keep order size at 100. Under the **Mod** column, change value to MktFml. Under the **Fml** column, press the ... button to open a formula editor, and enter the following formula:

   ```
   if (brokerpossize = 0 and RSIndexMod(0,M1,14) < 30, 1, 0)
   ```

   Under the **OnFill** column, set the value to S1.
4. Press the value under **Bid.** This will create a sell order.
5. Modify the sell order. Under the **Set** column, change the value to S1. Keep order size at 100. Under the **Mod** column, change value to LmtFml. Under the **Fml** column, press the ... button to open a formula editor, and enter the following formula:

   ```
   brokerposavgentry + 0.05
   ```
6. Right click on Strategic Order Entry Form and turn off **Auto Remove Plain Orders**.

Strategic Order Entry Form will look something like:



Strategic Order Entry (sf1)

Symbol	BaseSize	PosSize	PosNet	BidSize	Bid	Ask	AskSize	Last	TradeVol	MaxSize
MSFT	100	0		616	25.31	25.32	724	25.31	200	100

Set	Name	OC	Type	Size	Price	Mod	Fml	OnFill	A1	A2	A3	Status
			BUY	100		MktFml	...	S1	Arm		Delete	Idle
S1			SELL	100		LmtFml	...		Arm		Delete	Idle

MSFT | Flat | 100 x 25.42

Let's explain the orders you've entered.

First, the buy order is a market order triggered by formula. It is not yet armed, until you press **Arm** under **A1**. Once armed, the formula:

```
if (brokerpossize = 0 and RSIndexMod(0,M1,14) < 30, 1, 0)
```

is evaluated constantly. This formula will return 1 (true) when you do not have any position in MSFT and when a 14-period, 1-minute RSI indicator goes below 30.

Once the buy order is triggered, it will be sent to your broker. Since this is a market order, it will likely be filled right away and the buy order will trigger the sell order. Notice that you do not need to manually arm the sell order.

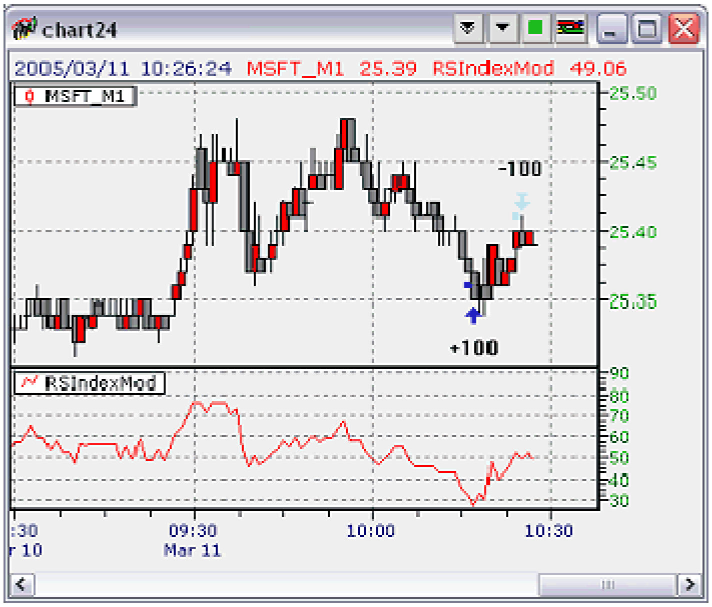
The sell order formula:

```
brokerposavgentry + 0.05
```

will return average entry price + $0.05.

Because the sell order is triggered by a buy order, average entry price + $0.05 will be larger than 0, it will be considered as a true condition and Strategic Order Entry Form will send the sell at limit order to your broker, using this value as the limit price.

If you run a 1-minute chart with the same RSI indicator along side with Strategic Order Form, you will see something like:



You can re-use this setup by re-arming the buy order.

For learning purpose, we recommend you to use Trade Simulator to test this set up before using a real account.

Example: Automatically Manage Orders with Cycle Orders

In the previous example, if you want to reuse the setup, you will need to rearm the buy order manually. In this example, we will show you how to set up the same RSI trading strategic throughout the day with all orders managed automatically.

The setup is exactly the same as above, except in the buy order, you set the **Mod** value to MktCyc. To use this setup, simply arm the buy order.

Let's go through how this setup works:

1. You arm the buy order.
2. The buy order will be sent when the formula condition is met (not currently in position and RSI < 30).
3. After a buy order is sent and filled, the formula still affects the order, but no more buy order will be sent. The buy order formula specifically forbids sending order when it is currently in position.
4. Once the buy order is filled, it will trigger the sell order.
5. After the sell order is filled, position size will become 0. So the buy order formula will send orders again when the RSI < 30 condition is met again.

Formulas in Strategic Order Entry Form

Overview

In Strategic Order Entry Form, you can use formulas to trigger orders. Basically, an order is sent to your broker only when a formula condition becomes true. Furthermore, the formula is used to determine order price in limit and stop orders.

Armed Status

When an order is triggered by formula, it can have an armed status. An armed status is an intermediate status between idle and live.

Similar to idle order, an armed order has not been sent to your broker yet. The formula condition for the order is being constantly evaluated. Once the formula condition becomes true, the order will be sent to your broker and the order will become live.

Types of Formula Condition

Setting the **Mod** column to one of the following values makes an order triggered by formula - MktFml, LmtFml, StpFml, MktCyc, LmtCyc and StpCyc.

MktFml, LmtFml, StpFml are sent-once orders. The following list illustrates how you can use these formula order types.

1. You create a new order (idle), fill in order information, formula condition, etc.

2 You change the **Mod** value to MktFml, LmtFml or StpFml.

3 You press **Arm** under **A1** column. The order becomes armed.

4 The formula is being evaluated. When the formula becomes true, Strategic Order Entry Form sends the order to your broker and the order becomes live.

5 At this stage, the order is similar to a regular order you send manually, you can adjust it just like a regular order (changing price, cancelling, going market, etc).

6 Once the order is filled/cancelled, the order becomes idle again.

7 You can arm the order again, i.e. Step 3 above.

MktCyc, LmtCyc, StpCyc are cyclical orders. They are sent multiple times and managed automatically. The following list illustrates how you can use these formula order types:

1 You create a new order (idle), fill in order information, formula condition, etc.

2 You change the **Mod** value to MktCyc, LmtCyc or StpCyc.

3 You press **Arm** under A1 column. The order becomes armed.

4 The formula is being evaluated. When the formula becomes true, Strategic Order Entry Form sends the order to your broker and the order becomes live. Unlike sent-once orders, you cannot adjust the order manually once it becomes live.

5 If the order is filled, another order will be placed provided the formula condition is still true.

6 If the formula becomes false, the order will be cancelled and the order status becomes armed again.

7 If the formula determines the price has changed (limit and stop orders), the price for the order will change.

8 Press **Unarm** under **A3** to stop a cyclical order.

Types of Formula

Strategic Order Entry Form uses quote window formula - the type of formula used in quote window. This means:

- You have access to all quote fields
- You have access to all formula functions that is specific to quote window (Level 2 functions, etc).
- You have access to all indicators.

Strategic Order Entry Form will start evaluating the formula once the order becomes armed. The return value of the formula determines the triggering condition. A return value of 0 means false, i.e. order will not be sent to broker. A non-0 return value means true, i.e. order will be sent to broker.

Furthermore, the return value also determines the price for limit and stop orders, i.e. if a formula returns 25.5 for StpFml, then Strategic Order Entry Form will send a stop order to your broker at the price of 25.5.

It is beyond the scope of this section to completely describe NeoTicker®'s formula language. For that, you should refer to ***Formula Topics and Tutorials*** (on page 255). Here we will provide some usage examples.

Formula	Type and Mod Column Value	Meaning
`if (brokerpossize > 0, brokerposavgentry + 0.02, 0)`	Sell, LmtFml	if currently in a long position, send a limit order to sell at $0.02 above the average entry price to your broker.
`if (brokerpossize > 0, brokerposavgentry - 0.03, 0)`	Sell, StpFml	if currently in a long position, send a sell stop order at $0.03 below the average entry price to your broker.
`if (brokerpossize = 0 and RSIndexMod(0,M1,14) < 30, 1, 0)`	Buy, MktFml	if currently not in position and RSI goes below 30, send a buy at market order to your broker.
`if (brokerpossize = 0 and (last - 0.10) < prevDLow(0,D1), 1, 0)`	Buy, MktFml	if currently not in position and last traded price is reaching $0.10 above previous day low, send a buy at market order to your broker.

Essential Formula Functions

Strategic Order Entry Form uses quote window formulas. Since it deals with orders, the following functions are particularly important, `BrokerPosSize` and `BrokerPosAvgEntry`. These functions returns the position size and the average entry price of the position at your broker's side.

For more information, see ***Position Information as Quote Fields and Formula Functions*** (on page 782).

Usage Example

See ***Strategic Order Entry Form, Usage Example: Using Formula to Trigger Order*** (see "Usage Example: Using Formula to Trigger Order" on page 725) for an example of using formulas in Strategic Order Entry Form.

Additional Settings

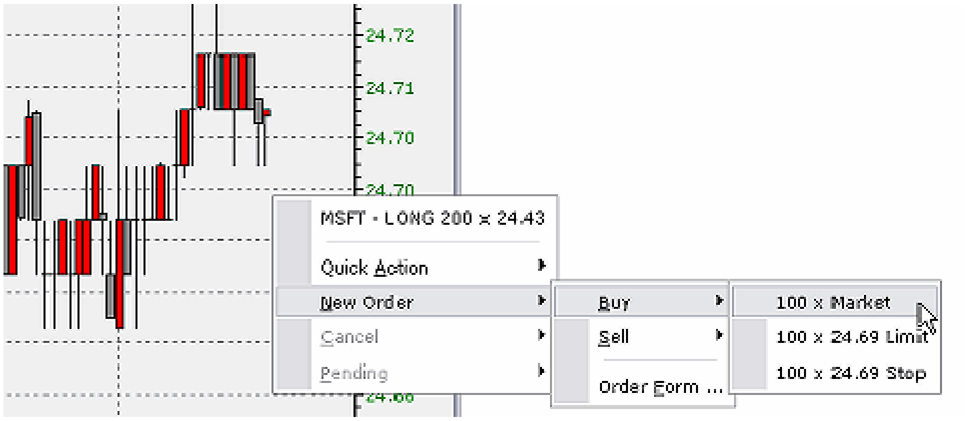
If you right click on Strategic Order Entry Form, a pop up menu is opened for additional settings.

Menu Item	Action
Hot Track	Hot tracking symbol from charts. See ***Manual Order Placement, General Usage Section*** (see "General Usage" on page 705).
Auto Size	Auto sizing position for symbol. See ***Manual Order Placement, General Usage Section*** (see "General Usage" on page 705).
Auto Remove Plain Orders	After a plain order is filled, it is automatically removed.
Quick Buttons	Show/Hide quick buttons.
Color Bid/Ask Changes	Use Color/Black&White to draw bid/ask changes
Update PosNet Using Bid/Ask	When calculation net position, use bid/ask price. If disabled, last traded price is used.
Time In Force	Day Only or Good Til Cancel. This is the setting to use when sending orders to your broker.
Time Frame	Time frame (trading time, holidays, etc) used for indicator calculation.
Max Size Per Direction	Same as Max Size column.
Route Orders to Trade Simulator	Route orders to Trade Simulator. Useful if you set Order Interface to connect to a live broker, but you want to test Strategic Order Entry Form using Trade Simulator.
Save, Save As	Saving Strategic Order Entry Form for reuse.

Order Entry Shortcut Keys

Instead of order forms, you can use shortcut keys to enter orders. To enter a buy order, press F2. To enter a sell order, press F3. You can press the order entry keys in any function window, e.g. charts, quote windows, pattern scanners, etc.

By default, pressing F2/F3 will open a menu to let you enter the order. The following figure shows the order menu when you press F2 in a chart.



It is possible to set F2/F3 keys to behave differently. The settings are in Order Interface Setup, under the **Shortcut** tab and **Order Defaults** tab. For more information, see ***Order Entry Shortcut Keys Setup*** (on page 739).

If you install NeoTicker® before version 4.1, F2/F3 are default to directly placing order without the menu. You may want to double check the shortcut keys settings to see if F2/F3 behavior is currently set to your preference.

Reusing Order Forms

You can save order forms with your favorite settings. Here is an example with Simple Order Entry Form:

1. In main window, choose **Order>New Order Form>Simple Order Entry**.
2. Enter a symbol and size under the **Quick** tab.
3. In main window, choose **Order>Save Order Form**.
4. A save dialog will open, choose a name.
5. To reuse the order form, choose from main window, **Order>Open Order Form**. Then select the file you saved.

Saving Multiple Order Forms

Save All

If you use multiple order forms frequently, you can save them all together.

1 In main window, choose **Order>Save All Order Forms**.

2 To open all order forms, in main window, choose **Order>Open Order Forms from Previous Session**, or press the main window tool bar icon .

Save As Set

Saving/loading order forms as a set gives you more control on which order forms to open.

1 Make sure you save each form form individually.

2 In main window, choose **Order>Save Set Of Order Forms As**.

3 A file dialog will open, choose a name to save.

4 Next time you want to use the order forms, in main window, choose **Order>Open Set Of Order Forms**.

Enabling/Disabling Order Confirmation

By default, after you place an order, you need to confirm it in Account Manager before the order will be sent.

To disable/enable order confirmation for orders entered by order forms:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup dialog.

2 In Order Interface Setup, press the **Confirmation** tab.

3 Under **Manual Order Confirmation**, under **Order Form Entry**, choose **Skip Confirmation** to disable order confirmation; choose **Centralize Confirmation** to enable order confirmation.

To disable/enable order confirmation for orders entered by shortcut keys (F2/F3):

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup dialog.

2 In Order Interface Setup, press the **Confirmation** tab.

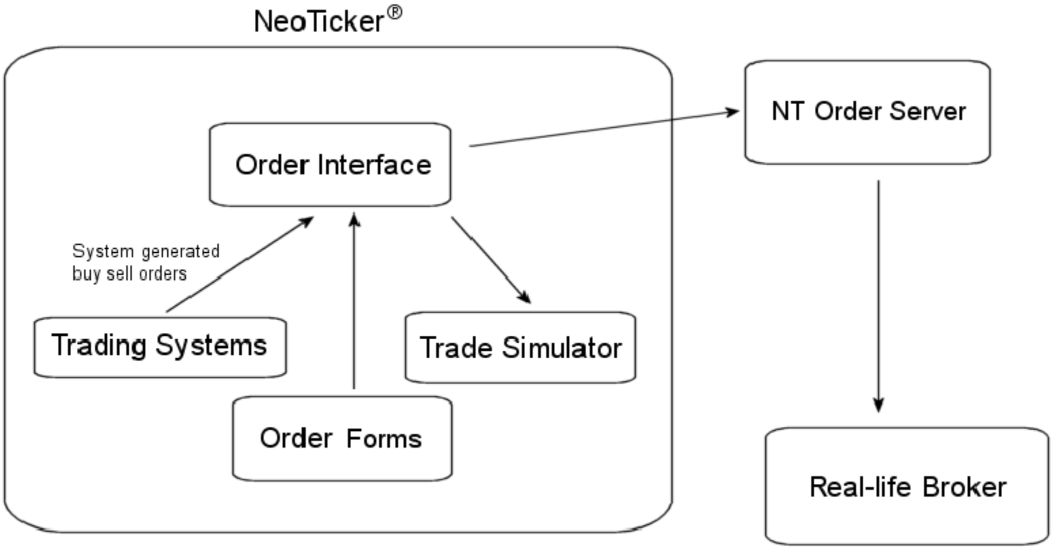
3 Under **Manual Order Confirmation**, under **Shortcut Entry**, choose **Skip Confirmation** to disable order confirmation; choose **Centralize Confirmation** to enable order confirmation.

If you disable order confirmation, orders are sent directly to your broker or Trade Simulator for processing without requiring you to confirm them.

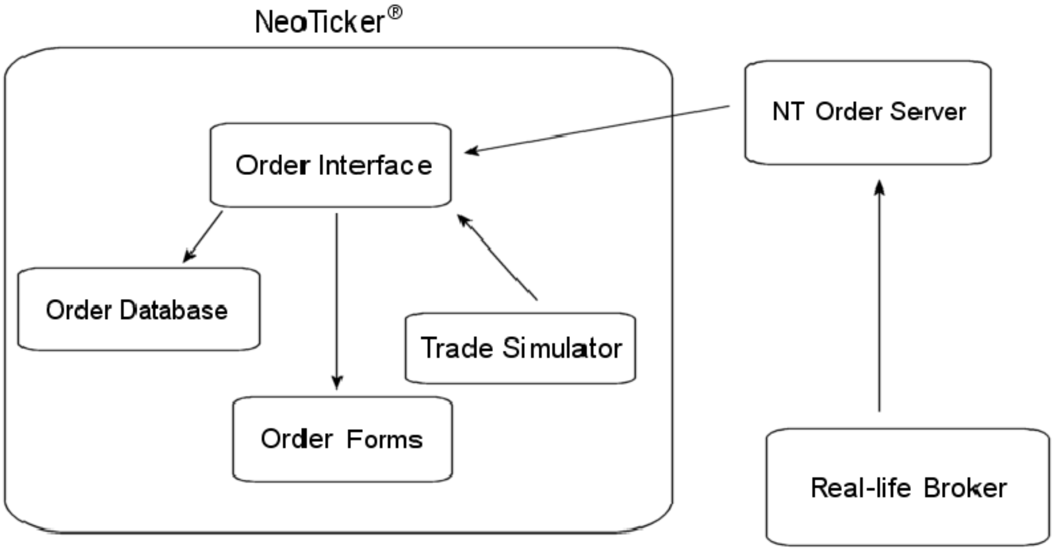
Order Interface

Introduction to Order Interface

Order Interface is the central hub for handling orders in NeoTicker®. When an order is placed, Order Interface decides where the order will go: a real-life broker or Trade Simulator. The scheme is outlined in the diagram below:



After an order is processed, order status is sent back through Order Interface to all interested parties. The scheme is outlined in the diagram below:



For an overview of the components, see ***How All Order Related Items Work Together*** (on page 695).

Roles of Order Interface Manager

All orders in NeoTicker® are first sent to Order Interface Manager. Order Interface Manager does the following:

- Not all orders are complete. Order Interface will fill in any missing parameters with default values. See ***Order Defaults and Completion Related Topics*** (on page 737).
- Route orders to either Trade Simulator or a broker connection/interface. See ***Order Routing Related Topics*** (on page 736).
- Route order status to interested parties

Accessing the Order Interface Setup

To open the Order Interface Setup, choose from the main window, **Order>Order Interface Setup.**

Order Routing Related Topics

Connecting to a Real-life Broker

See ***Connecting to Your Broker*** (on page 49).

Using Trade Simulator with Order Interface

Instead of a real broker, Trade Simulation can simulate order filling in a realistic manner.

To setup Order Interface to use Trade Simulator:

1. Choose from the main window, **Order>Order Interface Setup.**
2. Press **Connection** tab.
3. Choose **Trade Simulator**.
4. Press **OK** button.

The following options/features in Order Interface are not applicable if you use Trade Simulator:

- All settings that are related to exchange
- Position size match resolution

Trade Simulator relies on your real-time data feed to fill orders. Refer to ***Trade Simulator*** (on page 755) for more details.

Realistic Mode vs. Classic Mode

When interacting with Order Interface, Trade Simulator can behave in realistic mode (default) or classic mode.

In realistic mode, Trade Simulator behaves exactly like a real broker. You will use order entry forms to place orders, and use Account Manager to confirm and review orders.

Classic mode is for backward compatibility with Version 3 of NeoTicker® only.

To set classic/realistic mode:

1. Choose from the main window, **Order>Order Interface Setup.**
2. Press **Shortcut** tab.
3. Under **Order Routing When Connected To Trade Simulator**, choose **Realistic Mode** or **Classic Mode**.
4. Press **OK** button.

Order Defaults and Completion Related Topics

Day Orders and Good Til Cancel Orders

Day and good til cancel orders settings are required by real-life brokers. For Trade Simulator, this setting is optional.

To setup:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Order Defaults** tab.
3. Choose **Day** or **Good Til Cancel** under **Time Frame**.
4. Press **OK** button.

Decimal Places in Order

Decimal places setting is for formatting price information in order status returned by brokers.

To set decimal places:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Order Defaults** tab.
3. Enter settings under the **Dec Places** column for the instrument type.
4. Press **OK** button.

Order Confirmation

After you place an order, by default it requires confirmation in Account Manager. Confirmation gives you a last chance to review the order before it is sent.

In some circumstances, you may want to disable/enable order confirmation.

Manual Orders

For orders sent by order forms and F2/F3, see ***Enabling/Disabling Order Confirmation*** (on page 733).

Trading System Orders

For orders sent by trading systems:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Confirmation** tab.
3. Under **System Order Confirmation**, choose **Skip Confirmation** to disable order confirmation; choose **Centralize Confirmation** to enable order confirmation.
4. Press **OK** button.

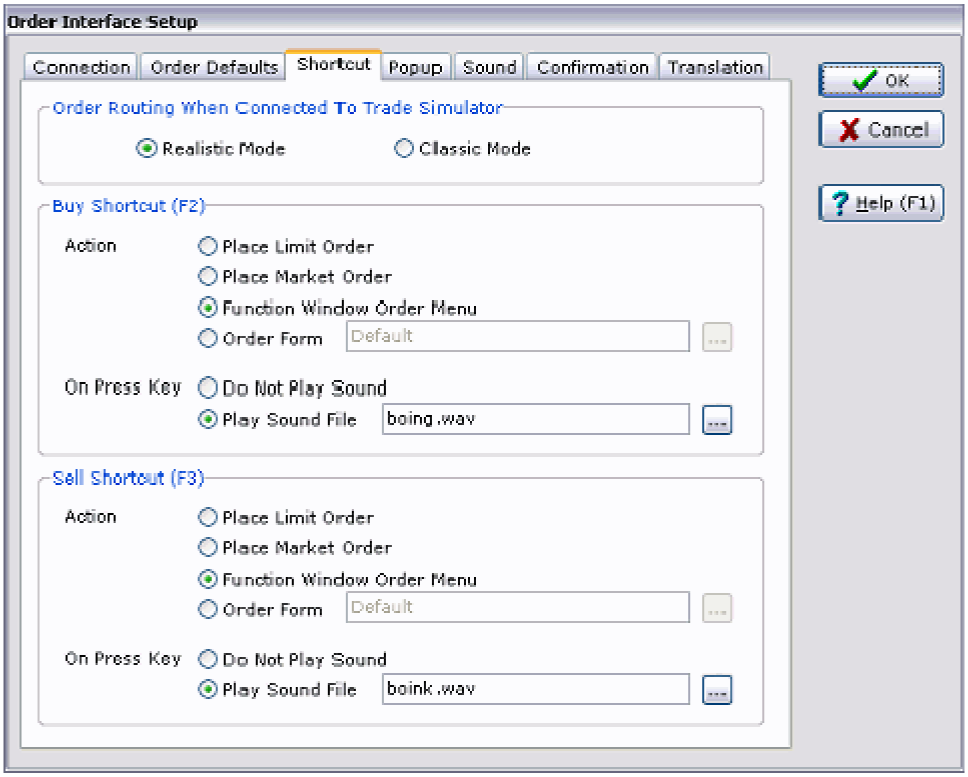
Order Entry Shortcut Keys Setup

F2 and F3 keys are order entry shortcut keys. When you press F2/F3 in a function window, one of the following actions is taken:

- (Default) An order menu is opened to let you choose an action;
- Limit order is placed;
- Market order is placed; or
- An order form is open to let you fill in the order details yourself

The exact action will depend on the shortcut keys settings. To change the settings:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Shortcut** tab.
3. Choose a setting for **Action**.
4. Press **OK** button.



Order Routing When Connected To Trade Simulator

● Realistic Mode ○ Classic Mode

Buy Shortcut (F2)

Action
○ Place Limit Order
○ Place Market Order
● Function Window Order Menu
○ Order Form Default

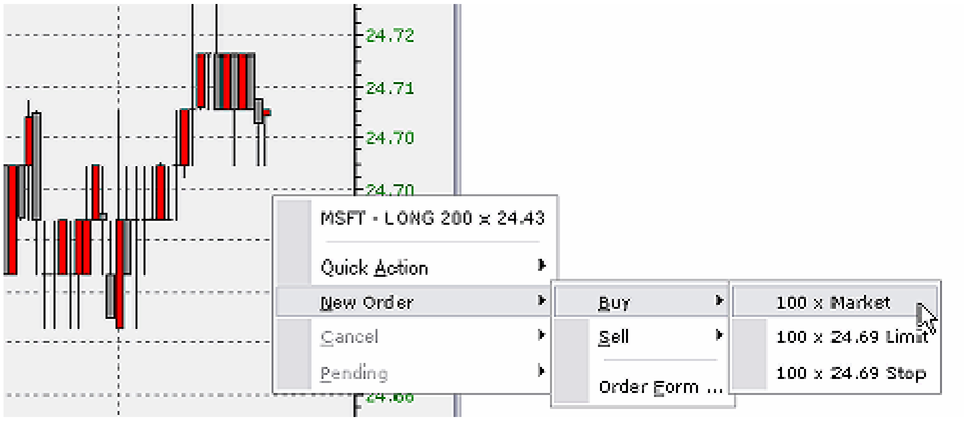
On Press Key
○ Do Not Play Sound
● Play Sound File boing.wav

Sell Shortcut (F3)

Action
○ Place Limit Order
○ Place Market Order
● Function Window Order Menu
○ Order Form Default

On Press Key
○ Do Not Play Sound
● Play Sound File boink.wav

Order Menu


24.72
24.71
24.70
24.70
24.68
MSFT - LONG 200 x 24.43
Quick Action
New Order
Cancel
Pending
Buy
Sell
Order Form ...
100 x Market
100 x 24.69 Limit
100 x 24.69 Stop

Order menu provides various options for placing orders:

Entry	Action
First entry	For displaying current position of the symbol.
Quick Action	If you already has a position, entries under this submenu will let you flat or reverse the position
New Order	For placing new market, limit and stop orders. It can also open order forms.
Cancel	For cancelling manual orders (open/pending orders that are placed manually, i.e. not placed by an automatic trading system).
Pending	For sending/cancelling all pending orders
Last entries	List all pending orders. You can send/cancel individual pending orders.

Limit/Market Order

If you set order entry shortcut keys to place a limit or market order, an order is sent right away when you press F2/F3 key.

To complete an order, F2/F3 needs to determine the following information:

- Symbol
- Price
- Order Size
- Order Type

Intelligence is used to determine these information. Because it is not possible to be 100% accurate when using intelligence, we strongly recommend you to have order confirmation enabled if you set F2/F3 keys to place orders right away.

Order Form

You can choose to use F2/F3 to open one of the order form for placing orders.

Order Confirmation

By default, when you place an order with shortcut keys, you need to confirm the order before it is sent to your broker. You can confirm the order in order form or in Account Manager. You can turn on/off order confirmation by:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Confirmation** tab.
3. Choose the setting under **Manual Order Confirmation**, **Shortcut Entry**.
4. Press **OK** button.

Play Sound

You can set to play a sound when F2/F3 is pressed. To setup:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Shortcut** tab.
3. Choose a sound file under **Buy Shortcut** and **Sell Shortcut**.
4. Press **OK** button.

Order Resolving Intelligence

The rest of the section describes how F2/F3 resolved various symbol, price, size information automatically. This information is useful only when you set F2/F3 to send orders directly without going through a menu/order form.

Symbol Resolution

The active function window determines the symbol for the order. The resolutions are listed in the following table:

Active Function Window	Symbol
Quote window	Symbol of the current row
Time and sales	Primary symbol
News	No symbol resolution
Volume distribution chart	Primary symbol
Time chart	Symbol of first data series
Ticker	First symbol of the ticker
Alert	No symbol resolution
Option chain	Primary symbol
Level 2	Primary symbol
System monitor	No symbol resolution
Hot list	Selected symbol
Cluster	Symbol of current row
Dynamic Table	No symbol resolution
Quote memo	Primary symbol

Price Resolution

The price for the order is determined for you. By default, the price is determined by the following table:

Active Function Window	Price
Time Chart	The pane price where the Y position of the mouse is located.

Quote window	If the mouse is on a price related column, such as last, bid, ask, the price on the column is used. Otherwise, last price is used.
All other function window	Last price of the symbol is used.

You can configure the order to use last price or price in function window. To configure:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

2 Press **Order Defaults** tab.

3 Under **Price**, choose **Last Price** or **Price Obtained From Function Window**.

If you choose **Last Price**, the last price of the symbol is used to fill the order's price.

If you choose **Price Obtained From Function Window**, the order's price is filled according to the table above.

Order Size Resolution

See ***Trade Size Setup*** (see "Trade Size and Scale Size Setup" on page 752).

Order Type Resolution

F2 and F3 key can be set to issue market or limit order. To setup:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

2 Press **Shortcut** tab.

3 Choose either **Limit** or **Mkt** under **Buy Shortcut** and **Sell Shortcut**.

4 Press **OK** button.

Position Size Matching

Position Size Matching is relevant only if you use a real-life broker, and the orders come from a trading system.

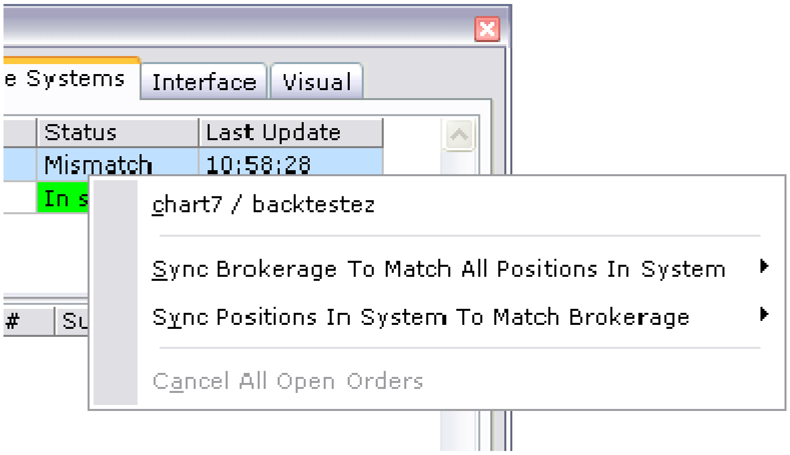
When a trading system holds a position, it assumes certain position size. This position size may or may not be the same as the position size at your brokerage account. For example, a trading system may think it longs 300 shares of IBM, while your brokerage account holds 200.

When trading system position size matches broker position size, we call it **in sync**. Otherwise,it is a **mismatch**.

A mismatch can happen when you place a manual order order outside of the trading system. For example, if your trading system is originally in sync with a position of long 300 shares IBM, it can become mismatch if you manually place an order to sell 100 shares of IBM.

Most trading systems are pretty simplistic and are not programmed to handle mismatch positions. For example, a system that does position sizing may not realize the broker position is different from what it thinks and size your position incorrectly. This can have undesirable consequences, like margin calls. So ideally, you will want trading system positions to be in sync with broker positions.

Position Synchronization With Account Manager



Tools are available in Account Manager to help bringing system and broker positions in sync. For more information, see ***Account Manager, Live Systems*** (see "Live Systems" on page 776).

Position Synchronization With Trading Systems Settings

When a trading system is transiting from historical calculation to real-time calculation, position mismatch can happen. Trading systems have options to deal with this type of mismatch automatically. See ***Time Chart, Sending Orders to Your Broker*** (see "Sending Orders to Your Broker" on page 1203).

Position Mismatch Warnings

When a trading system sends an order to Broker Interface, Broker Interface can detect position mismatch and warns you about it. You can decide whether to place order at the warning dialog.

The warning dialog can be turn on/off warning:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

2 Press **Confirmation** tab.

3 In **System Position Size Match Resolution**, choose **Centralize Confirmation, Send Order As Is,** or **Drop Order Without Warning**.

4 Press **OK** button.

For the warning dialog to work, Broker Interface must be able to query position information from your broker.

Security Type, Exchange and Currency Information

This section is relevant only for Interactive Brokers users

Interactive Broker requires exchange information when placing orders.

When you place order in NeoTicker®, Order Interface will automatically fill in the exchange information for you. So how does it work?

First, if exchange information is defined for the symbol in ***Symbol Info Manager*** (on page 921), Order Interface will use the defined exchange.

But if the symbol is not defined, Order Interface uses your data vendor to resolving for security type, then it maps the security type to an exchange.

We will illustrate with two examples, MSFT and ES Z4.

When you place an order for MSFT, Order Interface will query your data vendor to determine the security type of MSFT. In this case, it is a stock. Knowing MSFT is a stock, Order Interface will place the order to the SMART exchange of IB. IB will in turn send the order to NASDAQ.

When you place an order for ES Z4, Order Interface will query your data vendor to determine the security type of ES Z4. In this case, it is a future contract. Knowing ES Z4 is a future contract, Order Interface will place the order to the GLOBEX exchange of IB.

So everything is done automatically, what's to worry?

First, if you are not connected to a real-time data vendor, or if your data vendor does not provide security type information, Order Interface cannot resolve exchange information properly.

Second, if you are trading non-North America instruments, you will need to configure Order Interface before you can send orders to IB.

Default Security Type Setting

Default security type setting is used when Order Interface cannot resolve security type information from data vendor.

To change default security type setting:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Order Defaults** tab.
3. Under **Def Type** column, check a security type. This is the security type to use when Order Interface cannot resolve this information from data vendor.
4. Press **OK** button.

For example, if you trade stocks, set **Security Type** to **Stock**.

Exchange

To resolve exchange information, Order Interface looks at the security type of the instrument, and map it to an exchange.

To set exchange information:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Order Defaults** tab.
3. Fill in the exchange name under Exchange column. For example, you can fill in GLOBEX for S&P e-mini contracts.
4. Press **OK** button.

Currency

Some orders require a settlement currency setting, Order Interface looks at the security type of the instrument, and map it to a settlement currency. Usually you can leave currency setting blank unless you are trading Forex.

To set currency information:

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. Press **Order Defaults** tab.
3. Fill in the settlement currency under **Currency** column.
4. Press **OK** button.

Questions

We maintain a forum area to answer questions about order related information.

Order Interface Forum
(http://www.tickquest.com/forums/forumdisplay.php?forumid=30)

If you are not sure how to fill in order related information, please visit the forum and post a question there. It will be promptly answered.

If All Else Failed

You can always set the exchange and currency information using ***Symbol Info Manager*** (on page 921).

Single Entry Per Direction

If you enable Single Entry Per Direction (in Indicator Setup, **System** tab or in programming logic), NeoTicker® will block order placement in the same direction.

To establish a direction, one of the following conditions must be true:

- System already in position, or
- There is an open order

Once a direction is established, orders on the same instrument are blocked according to the following rules:

Direction Established By	Orders Blocked
Position	All entry orders in the same direction
Open market order	All entry orders in the same direction
Open order (non-market)	All entry orders that are of the same order type, with same non-blank comment.

In addition, once a direction is established, additional open entry orders are cancelled retroactively, according to the table above.

For live deployment issue related to Single Entry Per Direction, see ***Issue: Single Entry Per Direction Inconsistency*** (on page 792).

Sound Setup

You can setup Order Interface to play sound after certain events happen.

To setup sound for order forms and trading systems.

1. In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.
2. In Order Interface Setup, press **Sound** tab.
3. For order forms events, the settings are grouped under **Order Form Activities**.
4. For trading systems events, the settings are grouped under **System Activities**.
5. After you adjust the settings, press **OK** button.

The following table shows what events will trigger sound playing.

Event	Setting
User places an order with order form	**Order Form Activities**, **On Action** setting
User cancels an order with order form	**Order Form Activities**, **On Action** setting
Broker cancels/rejects an order that originates from an order form	**Order Form Activities, On Change** setting
Broker fills (including partial fills) an order originates from an order form	**Order Form Activities, On Change** setting
Trading system places an order	**System Activities**, **On Action** setting
Trading system cancels an order	**System Activities**, **On Action** setting
Broker cancels/rejects an order that originates from a trading system	**System Activities, On Change** setting
Broker fills (including partial fills) an order originates from a trading system	**System Activities, On Change** setting

To set up sound for shortcut keys, see ***Shortcut Key Setup*** (see "Order Entry Shortcut Keys Setup" on page 739).

Stealth Stop and Limit Orders

By default, stop and limit orders are sent to your broker for resolving, i.e. your broker or the exchange handles the logic of the order. In general, your limit and stop orders are visible to other traders.

If you have concerns about your limit/stop order being taken advantage of by other traders, NeoTicker® provides a stealth mode for limit and stop orders. With a stealth order, NeoTicker® monitors the price of the instrument and when the limit/stop price is hit, it places a market order to the broker. So with a stealth order, it is not visible to other traders until the corresponding market order is placed.

Pros:

- Your orders are not visible to other traders.

Cons:

- Possible slippage with stealth limit orders
- Possible slower execution speed

To enable/disable stealth orders:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup window.

2 In Order Interface Setup window, press **Translation** tab.

3 Under **Stop Orders** and **Limit Orders**, choose **Direct Resolving By Brokerage** or **Stealth Resolving By NeoTicker**.

4 Press **OK** button.

Symbol Info Manager Provides Defaults

When Order Interface receives an order, it prepares the order to send to your broker. For example, some broker requires orders to have exchange information. So Order Interface will fill in exchange information before sending out the order.

In general, when filling in order information, Order Interface goes through a two step process:

- First, ***Symbol Info Manager*** (on page 921) is consulted. If Symbol Info Manager has information about the symbol, Order Interface will use the information provided by Symbol Info Manager.
- If Symbol Info Manager does not have information about the symbol, Order Interface will use various intelligence to fill in order information. The settings are under Order Interface Setup's **Order Defaults** tab.

Symbol Translation Between Data Vendor and Broker

What is Symbol Translation

Data vendor and broker use different symbology to represent the same symbol. This is generally an issue for futures traders, but in some occasion can cause problems for stock traders.

For example, eSignal (a data vendor) calls the December 2004 S&P 500 e-mini contract ES Z4. On the other hand, MB Trading calls the same contract /ESZ4.

When you use NeoTicker® to trade, your charts, quote windows, etc use the data vendor symbology. When you place an order, the symbol must be properly translated to a symbol your broker understands.

How Order Interface Translates Symbols

When Order Interface receives an order, the default is to assume the symbol is in data vendor symbology. Order Interface will then use ***Symbol Info Manager*** (on page 921) to perform the symbol translation.

Trade Size and Scale Size Setup

Trade size is used by:

- The Default Size button in order form to fill in the **Number of Shares/Contracts** field.
- When you press F2/F3 shortcut key, it uses the trade size to create an order.

Scale size is used when you use up/down triangle buttons to quickly adjust trade size in order forms.

To set:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

2 Press **Order Defaults** tab.

3 Set the parameters under **Trade Size** and/or **Scale Size**.

Typical trade size for stocks is multiple of 100 (shares). Typical trade size for futures is multiple of single contract. Typical trade size for Forex is a large dollar value (e.g. 100000).

Typical scale size for stocks is 100 (shares). Typical scale size for futures is 1 (contract). Typical scale size for Forex is a large dollar value (e.g. 25000).

Trade Size can be overridden by Symbol Info Manager

Connection Related Options

Auto Connect on Startup

If Order Interface is configured to connect to an ActiveX NT Order Server such as Interactive Brokers or MB Trading, by default, NeoTicker® will connect to this order server on startup.

If you prefer to connect to order server manually:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

2 Press **Connection** tab.

3 Toggle **Auto Connect ActiveX Broker Interface on Startup** under **Options**.

4 Press **OK** button.

If this option is turned off, next time you start NeoTicker®, order placement will default to Trade Simulator. When you want to connect to a live broker, you will need to choose from main window, **Order>Connect>ActiveX Broker Interface** or press the main window tool bar button .

Turning off Connection to Live Broker

If you are connecting to a live broker, you can stop the connection by switching Order Interface to connect to Trade Simulator instead.

1 In main window, choose **Order>Connect>Trade Simulator** or press the main window tool button .

2 To reconnect to live broker, choose **Order>Connect>ActiveX Broker Interface** or press the main window tool bar button .

Auto Reconnect

It is possible to lose connection to a live broker. In this case, NeoTicker® is default to automatically reconnect to the live broker. If you prefer to turn off auto reconnect:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

2 Press **Connection** tab.

3 Toggle **Auto Reconnect ActiveX Broker Interface** under **Options**.

4 Press **OK** button.

Accept Order Placement Usage Agreement by Default

By default, when connecting to a real-life broker, NeoTicker® wants you to manually confirm this action. This can present a problem for turn key systems where NeoTicker® is running 24/7 with automatic system reboot. You can configure the Order Interface to auto confirm by:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

2 Press **Connection** tab.

3 Check **Accept Order Placement Usage Agreement by Default** under **Options**.

4 Press **OK** button.

Auto Track Symbol Positions

See ***Account Manager, Position Tracking*** (see "Position Tracking" on page 773).

Broker Specific Order Placement Script (Legacy)

Broker Specific Order Placement Scripts are no longer supported. Old scripts from version 3 are not compatible with version 4.

If you you are interested in developing an ActiveX Broker Interface for NeoTicker®, please contact our sales team.

Trade Simulator

Trade Simulator is a training tool for the practice of trading in a simulated environment. It can be used to help both discretionary trading and trading systems.

Trade Simulator

Trade Simulator is a broker simulation that comes with NeoTicker®. Trade Simulator execute trades using real-time data from your data feed. Because the trades are simulated, money is not involved. If orders are routed to Trade Simulator, you are conducting a real-life testing of the orders.

If the orders come from manual order entry, you are testing your trading skill.

If the orders come from a trading system, you are conducting a real-life testing of the trading system. Real-life testing of trading system is critical because historical testing often does not account for tick data.

Discretionary Traders

When you place orders within Trade Simulator, they are filled using real-time or historical data. You can place order for multiple symbols, and hold multiple positions at the same time just like a real brokerage account. All the orders you have placed and all the transactions that have occurred in the simulated account are saved.

You can analyze your performance using System Performance Viewer. System Performance Viewer provides detail statistical analysis of your performance.

Trading Systems

When an order from a trading system is routed to Trade Simulator, it is filled using real-time data. This gives you a feel on how a trading system works in real-time before money is involved.

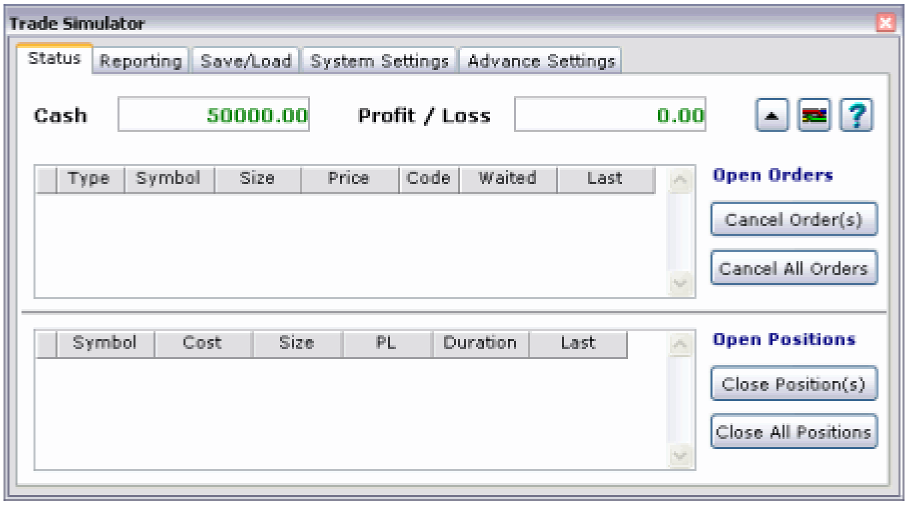
Trade Simulator offers the following advantages for testing trading system:

- Real-time testing is more realistic because real-time ticks are used to fill orders, versus the fill rules in back testing with historical data.
- Timing between orders can be observed in real-life.
- It helps debugging tick-by-tick updating trading systems.
- Trade Simulator collects statistics and report them in System Performance Viewer. Result from real-time testing can be compared with System Performance from historical data. This is not possible with your Broker's test account.

Opening the Trade Simulator

To open the Trade Simulator, choose from the main menu **Order>Trade Simulator.**

Trade Simulator will show up.



Cash 50000.00

Profit / Loss 0.00

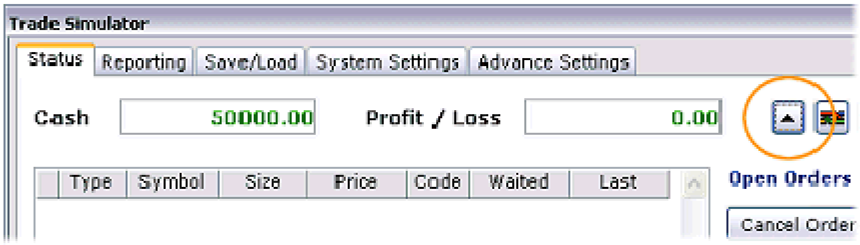
Think of Trade Simulator as your broker. You view your account status under the **Status** tab and use the other tabs for setting up various options.

Order Placement in Trade Simulator

You can place orders in Trade Simulator.

Placing order in Trade Simulator is mainly for legacy support and entering historical orders into Trade Simulator. To place real-time orders, you should use order forms (see ***Manual Order Placement*** (on page 705)).

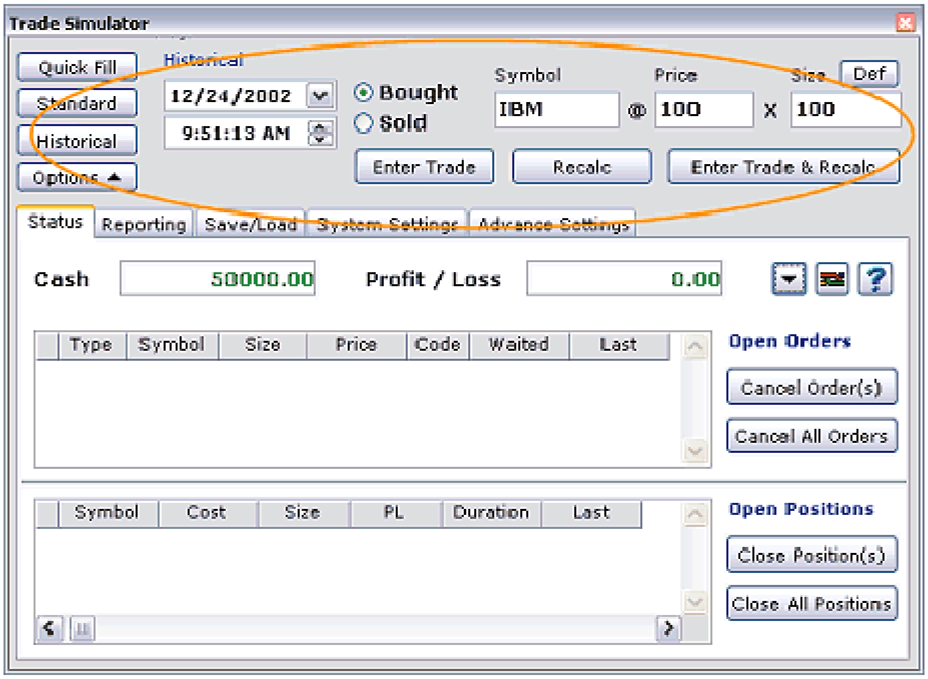
To place orders, press the triangle button to expand Trade Simulator.



Cash 50000.00

Profit / Loss 0.00

Once expanded, Trade Simulator has controls to place various orders.


Trade Simulator

Historical 12/24/2002 9:51:13 AM

● Bought ○ Sold

Symbol IBM @ Price 100 X Size 100

Cash 50000.00 Profit / Loss 0.00

The top part of the window is the order entry area. There are three modes you can place orders. They modes are controlled by the **Quick Fill**, **Standard** and **Historical** buttons at the upper left corner of Trade Simulator. The modes are:

- Quick Fill - Simply entry mode for market orders.
- Standard - Complete entry mode for market, limit and stop orders.
- Historical - Mode for entering orders in history.

Placing a Simple Buy or Sell at Market Order

To place a simple buy or sell at market order, all you need to do is to enter the order through the **Quick Fill** screen. If you are not currently within the **Quick Fill** screen, you can press the **Quick Fill** button at the upper left hand corner of the window to switch to **Quick Fill**.

In Quick Fill, you need to enter only the symbol and the corresponding number of shares or contracts you want to trade. Then press the **Buy at Market** button to enter a buy order, or press the **Sell at Market** button to enter a sell order.

You can change the size by entering it manually, or, you can click the **Default Size** button to get the symbol specific default size from the Symbol Info Manager. See ***Symbol Info Manager*** (on page 921).

Placing a Limit or Stop Order

To place a complex order that involves limit, stop, etc. you can switch to the **Standard** screen for order entry. If you are not currently within the **Standard** screen, you can press the **Standard** button at the upper left hand corner of the window to switch to the **Standard** order entry screen.

When you fill in the details of an order, the order placement button will change its caption to match your order. For example if you are placing a buy stop order, the caption of the order placement button at right hand side will have **BUY Stop** as its caption. It will reduce your chance of placing an order that is not what you intended.

You can manually change all the fields easily. The **Def Size** button will search the Symbol Info Manager for you to get the correct default size for that particular symbol. For more information, refer to ***Symbol Info Manager*** (on page 921).

About Limit Orders

All limit orders are placed as good 'til cancelled orders (code GTC). You must cancel the order manually.

If you place a limited order that can be filled with a better price with market order, the price for the market order will be used.

Placing a Historical Order

Trade Simulator can place historical orders for the purpose of studying special setups based on review of charts, or for the purpose of analyzing real trades from your brokerage records.

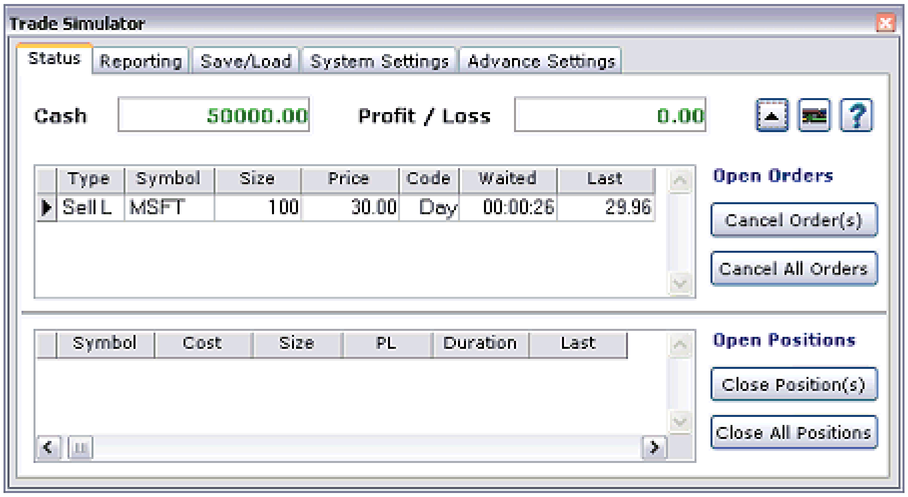
To place a historical order, switch to the **Historical Order** screen. You can switch to this screen by pressing the **Historical** button at the upper left corner of the window.

Within the **Historical Order** screen, you can enter the specific date time when the transaction took place. You must also enter directly the symbol, price, and size in order for Trade Simulator to record the transaction. Once you are done with the data entry, press the **Enter Trade & Recalc** button to save the transaction and update the Trade Simulator to reflect this change.

If you have a huge batch of orders to enter, you can choose to use the **Enter Trade** button to enter all the transactions first, then press the **Recalc** button after you are done with your data entry.

Cancelling Orders

Orders that are not filled immediately by Trade Simulator will show up under the **Status** tab in the Open Orders table.



Cash 50000.00 Profit / Loss 0.00

Type	Symbol	Size	Price	Code	Waited	Last
Sell L	MSFT	100	30.00	Day	00:00:26	29.96

Symbol	Cost	Size	PL	Duration	Last

The above figure shows that the there is a sell order for MSFT waiting to be filled.

You can select a pending order by clicking on the order within the table. Then press the **Cancel Order(s)** button to cancel the selected order.

To cancel multiple orders, you can select them from within the table using combination of regular left-click and CTRL left-click. Once you are done with your selection, press **Cancel Order(s)** button to cancel all the selected orders.

Cancel All Orders

To cancel all pending orders listed in the Open Orders table, press the **Cancel All Orders** button. Since the orders are filled in real-time by actual real-time data. If a tick arrives slightly earlier than your cancellation action, you would still end up with a transaction taken while most of the orders are cancelled successfully.

Closing Positions

Position Table

Once an order is filled, it will either establish a new position for you or change the standing of an existing position. In either case, you will see updated information in the Open Positions table.

Closing a Position

If you decide to close a position, you can either enter an order to offset the holding you have; or, you can select the open position in the table and then press the **Close Position(s)** button to automatically generate the correct order to close the open position for you.

Closing Multiple Positions

To close multiple positions, you can select all the positions that you want to close in the table with left-click and CTRL left-click combination. The press the **Close Position(s)** button, it will generate the necessary orders to close each position you have selected properly.

Closing All Positions

Use the **Close All Positions** button to close all the open positions listed in the table.

Performance Reporting in Trade Simulator

Generating System Performance Using Report Window, Excel, or Text File

To generate reports to output to report window, Microsoft Excel, or Text file, you can switch to **Reporting** tab. Check the report targets and enter the names of the target window name, workbook name or file name. Then press the corresponding report button to creation the report.

Automatic Saving of Performance

Your Trade Simulator performance is saved automatically in a file when you exit NeoTicker®. This allows you to continue to monitor your performance over multiple sessions.

Save the Performance Record Now

If you want Trade Simulator to save your performance immediately, switch to the **Save/Load** tab; press the **Save Now** button to force Trade Simulator to save all the records at once.

Modify the Default Trade Simulation Saving File

You can change the file name of the default Trade Simulator performance file to a name you like. Switch to the **Save/Load** tab you will see a **Filename** entry area, type the new name there will make all subsequent automatic save and load actions to use this new file name.

Disable Auto Save Option

By setting the **Auto Save/Load** option to **Disable**, under the **Save/Load** tab, Trade Simulator will no longer automatically load a default system performance file when NeoTicker® starts nor will it save the changes like new orders you have placed during the session when NeoTicker® exit. You will have to do that manually thru the **Save/Load** tab commands.

This option is useful if you intend to keep records of multiple trading models and do not want them to reflect orders and transactions from one another, then managing them manually is the best way.

Restoring Performance from File

If you are entering some historical trades and you find out you have made some mistakes and would like to start over, you can use the **Restore Now** button in the **Save/Load** tab to restore the Trade Simulator to the point where you last saved the data.

Trade Simulator usually save its data when NeoTicker® is exiting, or, when you press the **Save Now** button explicitly forcing it to save the data at once.

Resetting Your Trade Simulator Account

To erase all trades and reset trading system settings (e.g. commission, initial capital, multiples, etc) to default:

1 Press **System Settings** tab in Trade Simulator.

2 Press the **Reset Tracking** button.

To erase all trades but keep trading system settings intact:

1 Press **System Settings** tab in Trade Simulator.

2 Press the **Clear All Trades** button.

If you are interested to save a copy of the existing data into a backup file, use the commands under the **Save/Load** tab to do so first before completely resetting the simulator.

Saving/Loading Performance

Saving

You can save your current performance record into a separate file for backup purpose.

Switch to the **Save/Load** tab; press the **Save To File** button. You will be prompt to provide the file name to save to. Once you enter the file name, the current performance data will be saved to this new file.

Loading

You can load system data from a performance file to replace the data currently in your Trade Simulator.

Switch to the **Save/Load** tab and press the **Load From File** button. Select the performance file and the data in the Trade simulator will be replaced accordingly.

Equity Curve

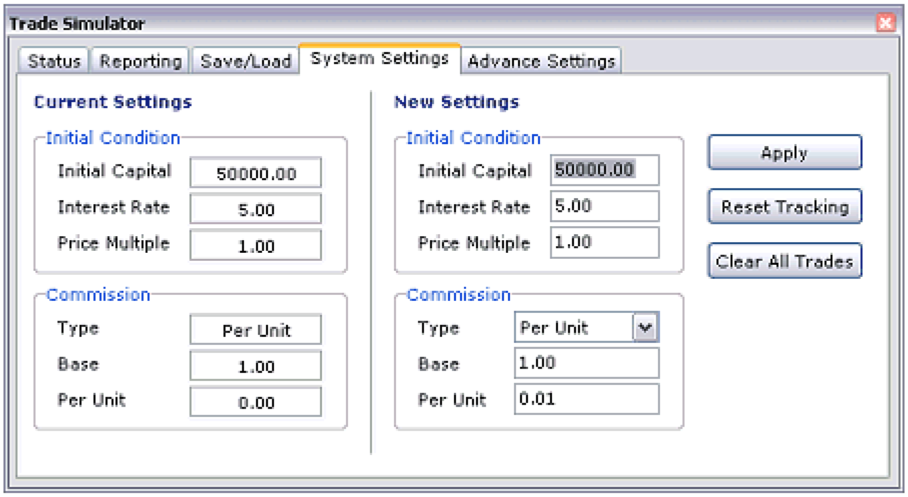
To generate more extensive trading reports such Maximum Averse Excursion and Maximum Favorable Excursion in Trade Simulator, you will need an equity curve. You can use the Trade Simulator or Trade Simulator Portfolio indicators to reconstruct the equity curve for a specific time frame from the trades you entered in Trade Simulator. Then you can generate system report from the equity curve.

For more information see ***Trade Simulator Indicator*** (on page 1341) and ***Trade Simulator Portfolio Indicator*** (on page 1343).

Simulation Settings

System Settings

The settings under **System Settings** tab controls settings such as initial capital, margin, etc.



Current Settings

Initial Condition

Initial Capital	50000.00
Interest Rate	5.00
Price Multiple	1.00

Commission

Type	Per Unit
Base	1.00
Per Unit	0.00

New Settings

Initial Condition

Initial Capital	50000.00
Interest Rate	5.00
Price Multiple	1.00

Commission

Type	Per Unit
Base	1.00
Per Unit	0.01

Modify the Initial System Settings

To adjust the initial capital or changing the commission scheme, you can switch to the **System Settings** tab. You can type your desired settings here and press the **Apply** button to modify the existing system to take your new settings. If the system already has recorded transactions, it will confirm with you first that you want to clear the existing records.

Price Multiple

The price-multiple information is set to 1. This default is useful for stocks, which have the move of 1 point in the price represents $1 movement per share.

For say the SP e-mini contract, its current price multiple is 50. That means every move of 1 point in the index future; it represents a movement of $50 per contract.

Price multiple is override by symbol specific settings in Symbol Info Manager. For example, if you trade a mixture of stocks and e-mini S&P contracts, you can set the e-mini S&P contract to have override multiple of 50 and have the trade simulator to have price multiple of 1.

Advance Settings - Decimal Places

If you are trading instruments like currencies and need to display 4 decimal places, you can modify **Decimal Places** settings in **Advance Settings** to allow Trade Simulator to properly display all the prices in 4 decimal places.

Fill on Bid/Ask

Because Trade Simulator uses real-time data to fill orders, you can improve the realism by using bid/ask data to determine the fill price and the fill's success.

To enable this feature:

1. Open Trade Simulator.
2. Press **Advance Settings** tab.
3. Enable the options **Market Order Fill On Bid/Ask** and **Limit Order Fill On Bid/Ask**.

If market orders are filled on bid/ask, for buy orders, the orders will be filled at ask price. For sell orders, the orders will be filled at bid price.

If limit orders are filled on bid/ask, for buy orders, the orders will be filled at ask price, and filled only if the limit price is equal or above the ask price. For sell orders, the orders will be filled at bid price, and filled only if the limit price is equal or below the bid price.

Partial Fill Simulation

Trade Simulator can simulate partial fills. This option is useful if you plan to trade on large volume or trade thinly traded instruments.

To enable partial fill simulation:

1. Choose **Order>Trade Simulator** to open Trade Simulator.
2. Under **Advance Settings** tab, enable **Generate Random Partial Fill**.

How Partial Fill Works (Fill on Bid/Ask Disabled)

When partial fill simulation is enabled, the order will be randomly filled by partial orders, e.g. if you order 5 contracts, the fill may consists of two partial fills, 2 and 3 contracts each.

Each partial fill will take some time to complete. So if the order is cancelled before completion, the final fill of the order will be partial.

If the size of the order is less than 100, partial fill simulation will fill on unit of 1 or more. This is to simulate contracts.

If the size of the order is more than 100, partial fill simulation will fill on unit of 100 multiples. This is to simulate stocks.

How Partial Fill Works (Fill on Bid/Ask Enabled)

When fill on bid/ask is enabled, an order will be completely filled if the bid/ask size overwhelms the order size. For example, if the order is to buy 100 shares, and if the ask size is 200, the order will be filled in one single fill.

Importing Trades

You can import trades in a text file into Trade Simulator. NeoTicker® must be off line (i.e. not connected to a real-time feed) for importing to work.

1 Open Trade Simulator.

2 Press **Import** tab.

3 Specify the fields.

4 Specify the separators.

5 Specify the options.

6 Select how the fields are separated (e.g. CSV, tab, space, semi-colon).

7 Press the **Import** button.

Field Order

Each line in the text file represents a trade. In another words, if you organize the trades into one line per trade in the text file, Trade SImulator should be able to import it. Your job is to tell Trade Simulator the format of the text file.

A trade in a single line consists of the following fields:

- Symbol
- Date time
- Trade size
- Fill price
- Commission cost

All fields are mandatory except commission. So in **Field Order**, you need to tell which field representing what value. If the text file does not include commission cost, check off **Has Commission Field** in **Options**.

Separators

The settings under **Separators** tells Trade Simulator how price and datetime are formatted. The default setting is for US English.

Options

Has Commission Field - check off if your text files does not have a field for commission cost.

Skip First Lines - if your text file contains header lines that do not represent trades, you can use this option to skip these lines.

Recalc system statistics after import - Once trades are imported, trading system statistics are affected. If this option is on, Trade Simulator recalculates statistics after text file is imported. In general you should leave this checked on. If off, you will need to press the **Recalc** button to recalculate statistics.

Use Special Date Format, Use Special Time Format - When interpreting date time, Trade Simulator uses your Windows' default date time format. Use these options to specify a rigid format (along with the settings in **Separators**) if you encounter problem during importing.

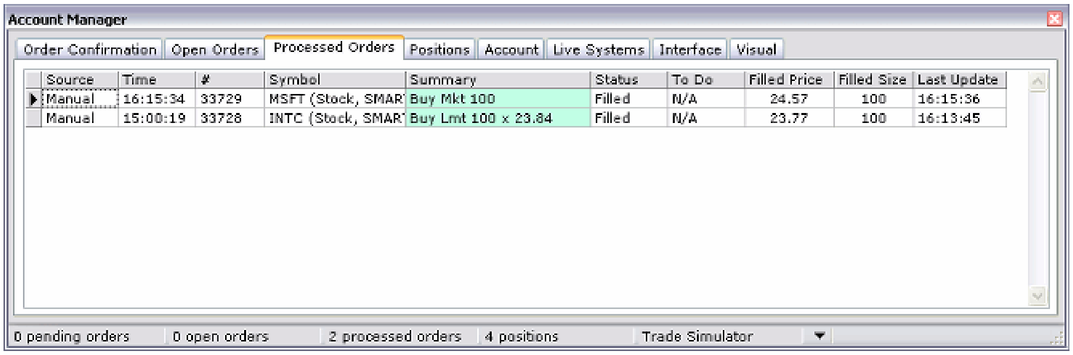
Field Separator

Text import supports fields that are separated by comma (CSV), tab, space and semicolon.

Account Manager

Introduction to Account Manager

Account Manager is the tool to manage orders and positions.

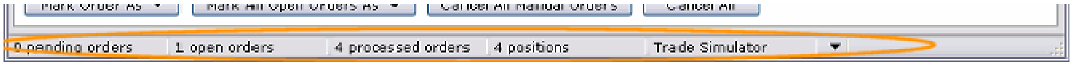


Source	Time	#	Symbol	Summary	Status	To Do	Filled Price	Filled Size	Last Update
Manual	16:15:34	33729	MSFT (Stock, SMAR	Buy Mkt 100	Filled	N/A	24.57	100	16:15:36
Manual	15:00:19	33728	INTC (Stock, SMAR	Buy Lmt 100 x 23.84	Filled	N/A	23.77	100	16:13:45

Account Manager will open on demand when you place an order that requires confirmation. To open Account Manager yourself, from main window, choose **Order>Account Manager**.

You use Account Manager to perform the following tasks:

- Confirming orders
- Querying orders, position and account information
- Viewing status of Broker Interface
- Quickly place orders to adjust positions



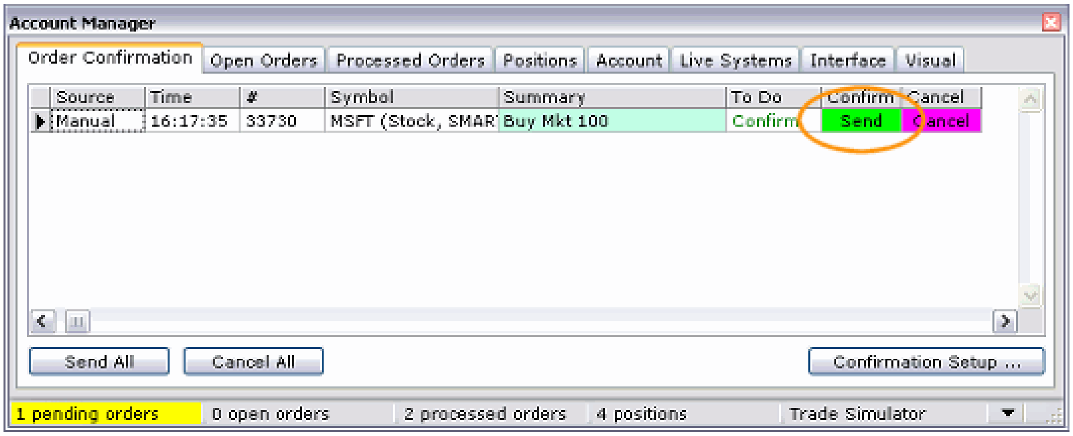
Bottom of Account Manager shows various status:

- Number of pending orders. Pending orders are orders that are in NeoTicker® order database. They are waiting for your confirmation before being sent to your brokerage account for execution.
- Number of open orders. Open orders are orders that is waiting to be filled in your brokerage account.
- Number of processed orders. Orders that have been processed by your broker.
- Number of positions in your brokerage account.
- Brokerage account information. This is where orders are sent and position information is queried. Brokerage account is connected through Broker Interface. It can be a real brokerage account, or Trade Simulator.
- Up/down triangle button. This button is for expanding/collapsing the integrated order form in Account Manager.
- When you click on an area in the bottom (e.g. pending orders), Account Manager will switch to the corresponding tab (e.g. **Order Confirmation** tab).

Confirming Orders

Pending orders are order residing in NeoTicker®'s database. They have not been sent to your brokerage account (via Broker Interface) for execution. By default, NeoTicker® requires you to confirm a pending order before it will be sent to your brokerage account.

When there are pending orders require you to confirm, Account Manager is open automatically. To confirm orders, go to **Order Confirmation** tab, and press **Send** under the **Confirm** column for the orders you want to send to your brokerage account.



Source	Time	#	Symbol	Summary	To Do	Confirm	Cancel
Manual	16:17:35	33730	MSFT (Stock, SMAR	Buy Mkt 100	Confirm	Send	Cancel

The table below shows the function of each column.

Column	Function
Source	Source of the order (manual, shortcut(F2/F3)) or from a trading system.
Time	Time the order is entered.
#	Internal order number.
Symbol	Symbol of the order. Resolved security type and exchange is shown in brackets.
Summary	Order summary.
To Do	Action requires you to do.
Confirm	Press **Send** under this column to confirm and send a pending order to broker.
Cancel	Press **Cancel** under this column to cancel a pending order.
Last Update	Time the order is last updated.

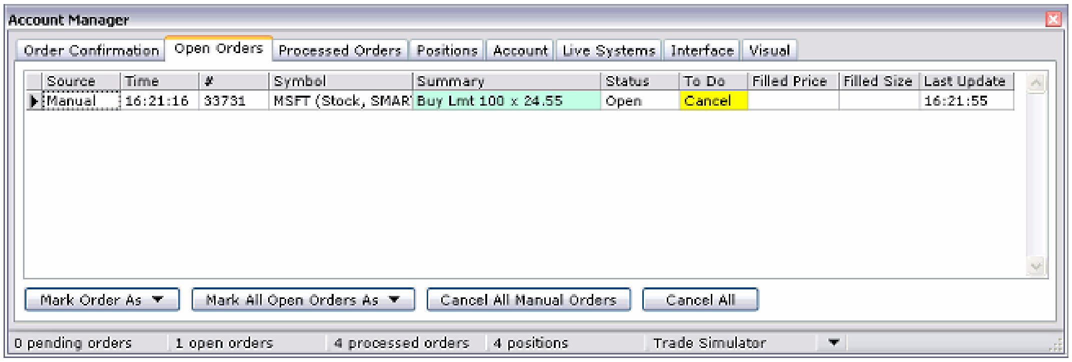
Pressing the **Send All** button will send all pending orders to your brokerage account.

Pressing the **Cancel All** button will cancel the pending order.

Press the **Confirmation Setup** button will open a dialog to let you adjust various confirmation settings.

Open Orders

All open orders (orders that have not been completely filled) are shown under the **Open Orders** tab.



Source	Time	#	Symbol	Summary	Status	To Do	Filled Price	Filled Size	Last Update
Manual	16:21:16	33731	MSFT (Stock, SMAR	Buy Lmt 100 x 24.55	Open	Cancel			16:21:55

0 pending orders | 1 open orders | 4 processed orders | 4 positions | Trade Simulator

You can right click on an order to open pop up menu to make adjustment to the open order.

The table below shows the function of each column.

Column	Function
Source	Source of the order (manual, shortcut(F2/F3)) or from a trading system.
Time	Time the order is entered.
#	Internal order number.
Symbol	Symbol of the order. Resolved security type and exchange is shown in brackets.
Summary	Order summary.
Status	See ***Order Status*** (on page 780).
To Do	Action you apply on the order.
Filled Price	Current filled price. This is the average of partial fill price.
Filled Size	Current filled size. This is the sum of partial fill size.
Last Update	Time the order is last updated.

The buttons **Mark Order As** and **Mark All Open Orders As** are for marking selected/all order(s) to a status. This is for handling stale orders. See ***Stale Order Handling*** (on page 780).

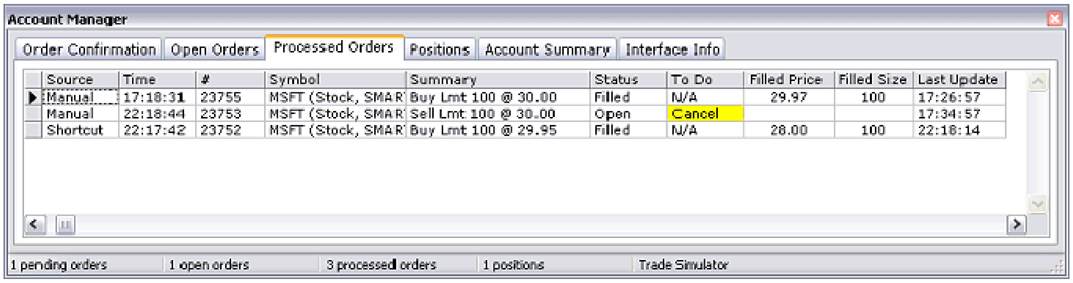
Pressing the **Cancel All Manual Orders** button will cancel all orders that are manually placed, i.e. orders that are not placed by an automatic trading system.

Pressing the **Cancel All** button will cancel all orders.

Note that open orders are orders that are already residing on your brokerage account waiting to be filled. So order cancellation may not succeed.

Processed Orders

All processed orders (orders that have been confirmed and sent to Broker Interface) are shown under the **Processed Orders** tab.



Source	Time	#	Symbol	Summary	Status	To Do	Filled Price	Filled Size	Last Update
Manual	17:18:31	23755	MSFT (Stock, SMAR	Buy Lmt 100 @ 30.00	Filled	N/A	29.97	100	17:26:57
Manual	22:18:44	23753	MSFT (Stock, SMAR	Sell Lmt 100 @ 30.00	Open	Cancel			17:34:57
Shortcut	22:17:42	23752	MSFT (Stock, SMAR	Buy Lmt 100 @ 29.95	Filled	N/A	28.00	100	22:18:14

1 pending orders | 1 open orders | 3 processed orders | 1 positions | Trade Simulator

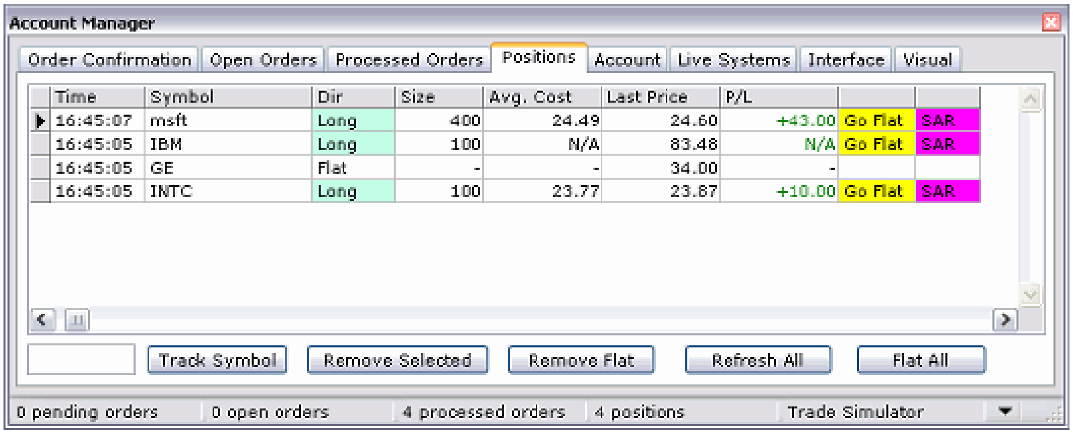
By checking off **Show Cancelled Orders**, you can hide all cancelled orders.

The table below shows the function of each column.

Column	Function
Source	Source of the order (manual, shortcut(F2/F3)) or from a trading system.
Time	Time the order is entered.
#	Internal order number.
Symbol	Symbol of the order. Resolved security type and exchange is shown in brackets.
Summary	Order summary.
Status	See ***Order Status*** (on page 780).
To Do	Action you apply on the order.
Filled Price	Current filled price. This is the average of partial fill price.
Filled Size	Current filled size. This is the sum of partial fill size.
Last Update	Time the order is last updated.

Positions

Your current positions are listed under the **Positions** tab.



Time	Symbol	Dir	Size	Avg. Cost	Last Price	P/L		
16:45:07	msft	Long	400	24.49	24.60	+43.00	Go Flat	SAR
16:45:05	IBM	Long	100	N/A	83.48	N/A	Go Flat	SAR
16:45:05	GE	Flat	-	-	34.00	-		
16:45:05	INTC	Long	100	23.77	23.87	+10.00	Go Flat	SAR

0 pending orders | 0 open orders | 4 processed orders | 4 positions | Trade Simulator

Below is the meaning of each column:

Column	Meaning
Time	Time the position is last updated. If the time stops progressing, it means no NeoTicker® component is tracking this position.
Symbol	Symbol of the position.
Direction	Long, short or flat.
Size	Position size.
Avg. Cost	Average cost of the position.
Last Price	Last price of the symbol.
P/L	Profit/loss of the position.
Go Flat	Pressing cell under this column will make the position go flat.
SAR	Pressing cell under this column will stop and reverse the position.

The buttons **Track This Symbol**, **Remove Selected** and **Refresh All** are for position tracking. See ***Position Tracking*** (on page 773).

Pressing the **Flat All** button will flat all positions.

Placing orders in Account Manager is closely related to position information. See ***Account Manager Order Placement*** (on page 773).

Position Tracking

When position of a symbol is tracked, position information is retrieved from broker automatically when NeoTicker® starts. Position of a symbol is tracked automatically if orders for the symbol has been filled in current and previous sessions of NeoTicker®. This is useful to monitor positions that are opened by previous sessions of NeoTicker®.

Tracking positions automatically (default behavior) is usually desirable. To turn off automatic position tracking:

1 In main window, choose **Order>Order Interface Setup** to open Order Interface Setup.

2 Press **Connection** tab.

3 Under **Options**, toggle **Auto Track Symbol Positions**.

4 Press **OK** button.

If you turn off auto symbol tracking, position information, you can still manually track the position of a symbol (without placing an order):

1 Open Account Manager.

2 Under **Positions** tab, enter the symbol in the box in the lower left corner of Account Manager, then press **Track This Symbol** button.

Remove Tracking

If you want to manually disable tracking for a symbol:

1 Open Account Manager.

2 Under **Positions** tab, select the symbol you want to disable tracking, then press **Remove Selected** button.

When you disable tracking for a symbol, you are only removing the symbol from being tracked. This will not affect your actual position.

Refresh Tracking

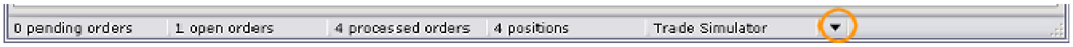
You can force the tracked position to refresh. Latest information will be gathered from your broker.

1 Open Account Manager.

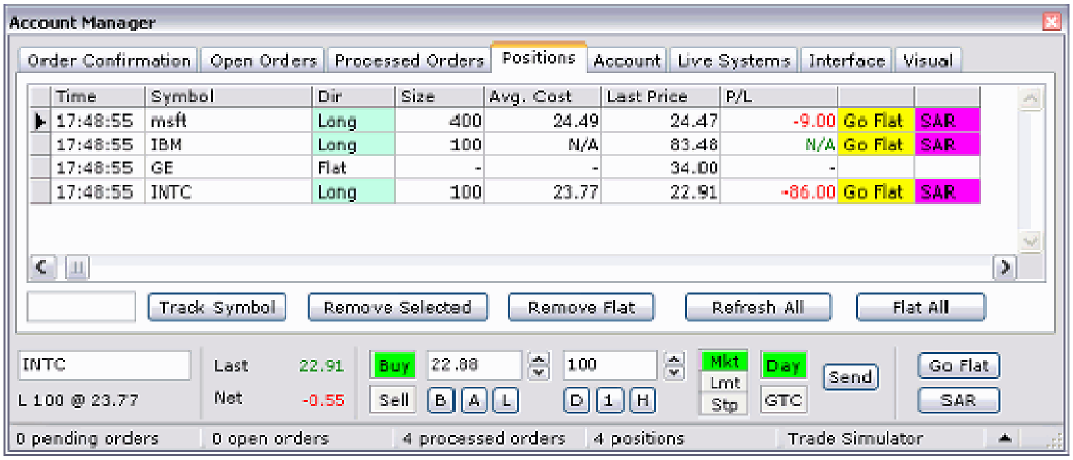
2 Under **Positions** tab, press **Refresh All** button.

Account Manager Order Placement

You can place orders directly in Account Manager. The primary reason of doing so is for position control. This means you will want to place order while **Position** tab is active, although you can place orders anywhere in Account Manager.



To open the integrated order form in Account Manager, click on the down triangle in the lower right corner of Account Manager.

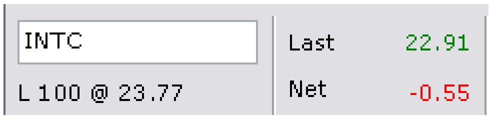


Time	Symbol	Dir	Size	Avg. Cost	Last Price	P/L		
17:48:55	msft	Long	400	24.49	24.47	-9.00	Go Flat	SAR
17:48:55	IBM	Long	100	N/A	83.48	N/A	Go Flat	SAR
17:48:55	GE	Flat	-	-	34.00	-		
17:48:55	INTC	Long	100	23.77	22.91	-86.00	Go Flat	SAR

INTC
L 100 @ 23.77
Last 22.91
Net -0.55
22.88
100

0 pending orders | 0 open orders | 4 processed orders | 4 positions | Trade Simulator

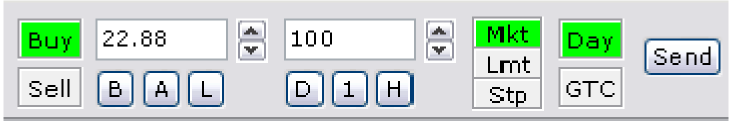
Symbol and Position Information



INTC
L 100 @ 23.77
Last 22.91
Net -0.55

This part shows the symbol and position information. You can change the symbol quickly by clicking on a position at the top of Account Manager.

Order Placement



22.88
100

This part is for placing order. The basic workflow is:

1 Choose **Buy** or **Sell**.

2 Enter price.

3 Enter size.

4 Choose an order type.

5 Choose **Day** or **GTC** order.

6 Press **Send** button.

Table below describes the various items:

Item	Description
Buy / Sell	Whether the order is a buy or a sell order.
B, A, L	For quickly fill in the price. B for bid price. A for ask price. L for last price.
D, 1, H	For quickly fill in the order size. D for default size (as determined by Symbol Info Manager), 1 for size 1, H for 100.
Mkt, Lmt, Stp	Order type - Market, Limit or Stop.
Day, GTC	Day or GTC (Good Til Cancel) order.
Send	Button for sending the order.

Go Flat and SAR

Pressing the **Go Flat** button will flat the position.

Pressing the **SAR** button will stop and reverse the position.

Order Confirmation

The integrated order form is a type of order confirm. The confirmation behavior is controlled by the same settings used by other order forms. See ***Confirming Orders*** (on page 768).

Hot Track

Hot Track is a feature linking symbols to Account Manager's order placement. When you click a symbol in a function window (e.g. chart, quote window, etc), Account Manager will automatically fill in the symbol and default order size for you.

If you want to disable Hot Track, right click on the order form to open pop up menu, and disable **Hot Track**.

Account Summary

Account tab shows the account summary reported by your broker. Fields under this tab is broker dependent. Different brokers return different information about your account.

Live Systems

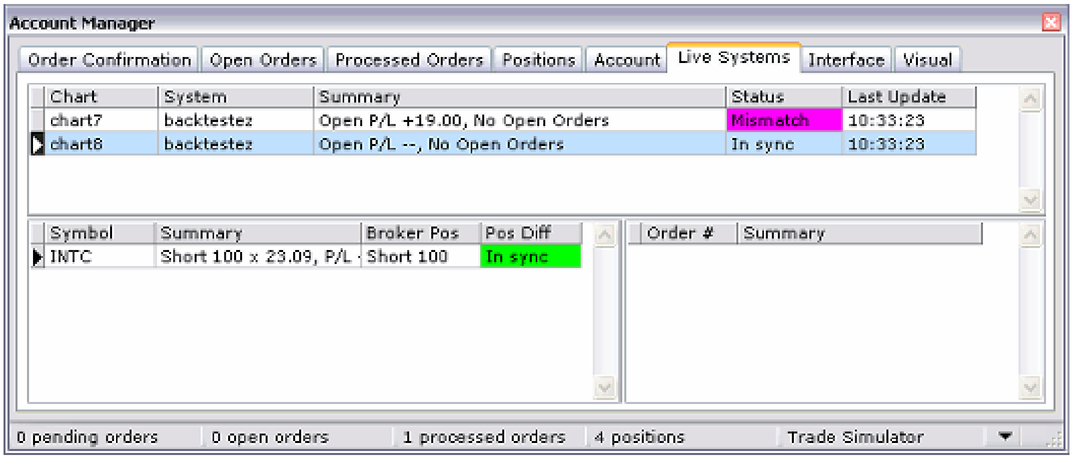


Chart	System	Summary	Status	Last Update
chart7	backtestez	Open P/L +19.00, No Open Orders	Mismatch	10:33:23
chart8	backtestez	Open P/L --, No Open Orders	In sync	10:33:23

Symbol	Summary	Broker Pos	Pos Diff
INTC	Short 100 x 23.09, P/L ·	Short 100	In sync

Order #	Summary

0 pending orders | 0 open orders | 1 processed orders | 4 positions | Trade Simulator

Live System tab is used to track order placed by automatic trading system. Orders that go through Broker Interface are tracked under this tab.

Top Table

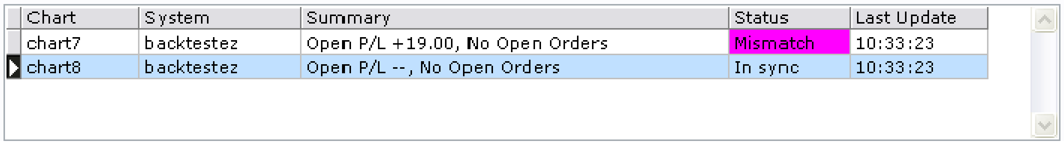


Chart	System	Summary	Status	Last Update
chart7	backtestez	Open P/L +19.00, No Open Orders	Mismatch	10:33:23
chart8	backtestez	Open P/L --, No Open Orders	In sync	10:33:23

The top table lists all the charts that contain trading system(s). You can click on a chart to select it to view more details about the chart and its system. If a chart contains multiple trading systems, each trading system will be listed as a single entry in this table.

Chart column displays the chart names.

System column displays the name of the systems.

Summary column display system summaries.

Status column displays the system status. Status can be **In Sync** or **Mismatch**. See System Broker Position Synchronization section below for more information.

Last Update column is the last time Live Systems tab obtained information about the system.

Lower Left Table

Lower left table lists the position information of the selected system.

Symbol is the symbol for the position.

Summary is the position summary. The position listed here is the position expected by the trading system.

Broker Pos is the broker position size, i.e. position size at your brokerage account.

Pos Diff is the difference between system position size and broker position size.

Note that system position size and broker position size can mismatch. See System Broker Position Synchronization section below for more information.

Lower Right Table

Lower right table lists the open orders of the selected system.

System Broker Position Synchronization

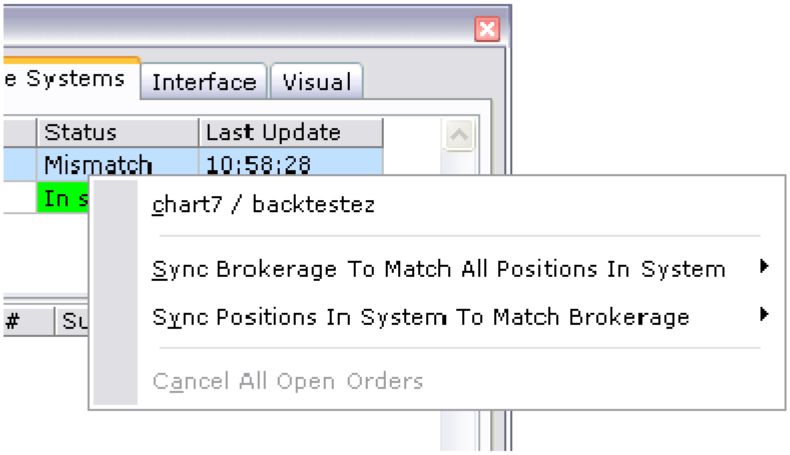
When a trading system holds a position, it assumes certain position size. This position size may or may not be the same as the position size at your brokerage account. For example, a trading system may think it longs 300 shares of IBM, while your brokerage account holds 200.

When trading system position size matches broker position size, we call it **in sync**. Otherwise,it is a **mismatch**.

A mismatch can happen when you place a manual order order outside of the trading system. For example, if your trading system is originally in sync with a position of long 300 shares IBM, it can become mismatch if you manually place an order to sell 100 shares of IBM.

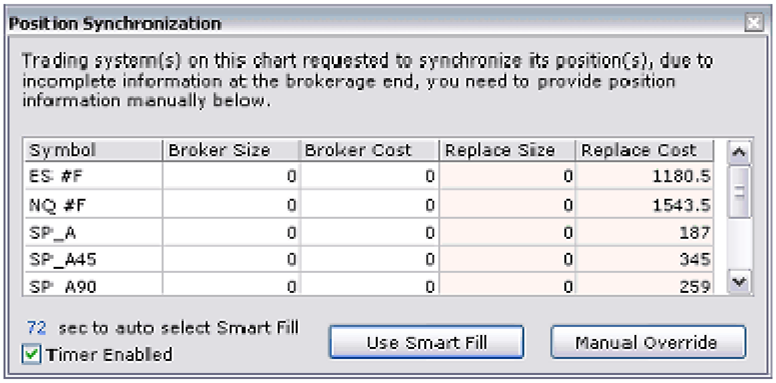
Most trading systems are pretty simplistic and are not programmed to handle mismatch positions. For example, a system that does position sizing may not realize the broker position is different from what it thinks and size your position incorrectly. This can have undesirable consequences such as margin calls. So ideally, you will want trading system positions to be in sync with broker positions.

Tools are available to help bringing system and broker positions in sync. When you have a mismatch position, you can right click in the top table to open a menu. Various synchronization options are available:



From the menu, you can:

- Synchronize your broker position to match trading system. This is done by sending real orders to your broker. You should use it with extreme caution.
- Synchronize your trading system to match broker position. This option can work automatically provided Broker Interface can query position entry price and size from your brokerage account. This type of query is not supported by all brokers, if querying is not possible, a dialog is opened to let you fill in position entry price and size.


Position Synchronization

Trading system(s) on this chart requested to synchronize its position(s), due to incomplete information at the brokerage end, you need to provide position information manually below.

Symbol	Broker Size	Broker Cost	Replace Size	Replace Cost
ES #F	0	0	0	1180.5
NQ #F	0	0	0	1543.5
SP_A	0	0	0	187
SP_A45	0	0	0	345
SP_A90	0	0	0	259

72 sec to auto select Smart Fill

☑ Timer Enabled

- Cancel all open orders.

Synchronization orders are the same type as order form orders. So order confirmation is controlled by the option that controls order form orders confirmation.

A Note on Trade Simulator

If you want to use **Live Systems** tab to track orders sent to Trade Simulator, you will need to:

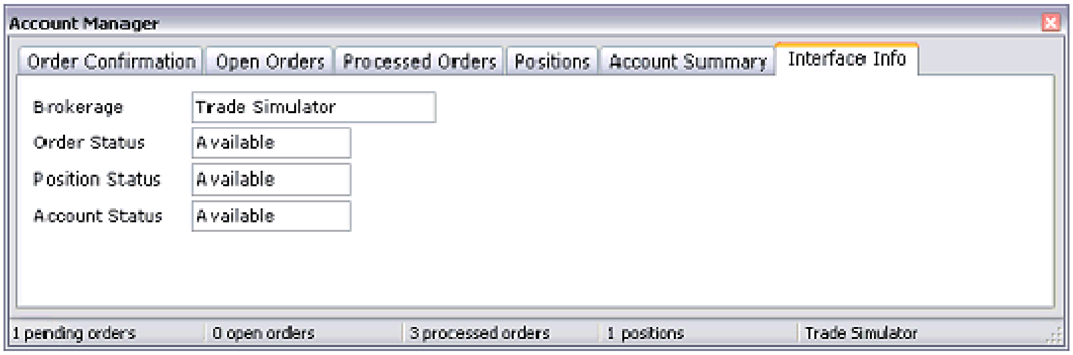
1 Configure the trading system to send orders to Broker Interface.

2 Configure Broker Interface to send orders to Trade Simulator.

Live Systems tab does not track orders that are sent directly from trading systems to Trade Simulator without going through Order Interface.

Broker Interface Information

Under **Interface** tab is the information of the current Broker Interface you are currently connected to.



Brokerage: Trade Simulator
Order Status: Available
Position Status: Available
Account Status: Available

1 pending orders | 0 open orders | 3 processed orders | 1 positions | Trade Simulator

Field	Meaning
Brokerage	The name of Broker Interface or Trade Simulator.
Order Status	Whether the broker supports returning order status to NeoTicker®.
Position Status	Whether the broker supports returning position status to NeoTicker®.
Account Status	Whether the broker supports returning account status to NeoTicker®.

Order Status

Once an order is sent to Broker Interface, Account Manager will display its order status under the **Open Orders** and **Processed Orders** tab.

Table below shows the meaning of each order status:

Order Status	Meaning
Open	Order is open.
Partial Filled	Order has been partially filled.
Rejected	Either Broker Interface, your broker or Trade Simulator rejects the order.
Cancelled	Either you or your broker cancelled the order.
Filled	Order has been completely filled.

Stale Order Handling

It is possible an order becomes stale, i.e. the order no longer receives order status updates from your broker. In this case, the order will still be listed in Account Manager. It is recommended you manually change the status of the staled order (e.g. mark an open order to cancel).

Marking a staled order only changes the status of the order. The marking does not issue commands to your broker.

1 Open Account Manager.
2 Press **Open Order** tab.
3 Click on the order to be marked.
4 Press the **Mark Order As** button to open a menu, and choose status you want to mark the order as.

Visual Options

In Account Manager, orders are highlighted by different colors. You can specify your own color scheme under the **Visual** tab.

Connecting to a Live Broker

See ***Connecting to Your Broker*** (on page 49).

Order Events

When Order Interface is connected to NT Order Server to place order, it produces logs in Order Events. The logs can be useful to help figuring out connection problem to brokers.

To view Order Events:

1 In main window, choose **Program>System Log**.

2 Press the **Order Events** tab.

Position Information as Quote Fields and Formula Functions

Quote Fields

You can access position information as quote fields, i.e. you can use these quote fields directly in quote window, dynamic window, quote memo, etc.

Quote Field	Value
`BPosAvgEntry`	Average entry price of the position for the symbol.
`BPosSize`	Position size for the symbol.

Formula Functions

You can access average entry price and position size also as quote formula functions. These functions are available in quote window formula. In function form, you can specify account type and symbol explicitly.

Function	Value
`BrokerPosSize`	Position size for the default symbol.
`BrokerPosSize(AcctType)`	Position size for the default symbol at the specified account.
`BrokerPosSize(AcctType, Symbol)`	Position size for the specified symbol at the specified account.
`BrokerPosAvgEntry`	Average entry price of the position for the default symbol.
`BrokerPosAvgEntry(AcctType)`	Average entry price of the position for the default symbol at the specified account.
`BrokerPosAvgEntry(AcctType, Symbol)`	Average entry price of the position for the specified symbol at the specified account.

`AcctType` is an integer. If `AcctType` is 0, the function returns the value of the default broker, e.g. if your Order Interface is specified to connected to MB Trading, `BrokerPosSize(0, "MSFT")` returns the position size for MSFT in your MB Trading account. If `AcctType` is 1, the function returns the value of Trade Simulator.

For example, the following quote window formula calculates the position size difference between the symbols in a quote window and MSFT:

```
BrokerPosSize(0) - BrokerPosSize(0, "MSFT")
```

Note that the functions `BrokerPosSize` and `BrokerPosAvgEntry` without any parameter returns exactly the same value as the quote fields `BPosSize` and `BPosAvgEntry`. You can use either one according to your preference.

Trading System Live Deployment Guide

Introduction

This section is about live deploying a trading system, i.e. following the trading system to trade. It is assumed that the trading system has been back tested with historical data rigorously and you are satisfied with the result.

Past performance is not a guarantee of future results. Only risk capital should be used to trade futures, stocks, options on futures or stocks, mutual funds or any other type of financial instruments. Whether this product is used in conjunction with futures, stocks, stock indices or mutual funds, all involve a high degree of inherent financial risk and the possibility of loss is great. TickQuest Inc. does not assume any responsibilities, make any guarantees whatsoever, or make any trading recommendations in NeoTicker®. All such investment vehicles carry risks and all trading decisions are ultimately made by you. You are solely and individually responsible for those decisions and the results of those decisions.

Manual vs. Automatic Order Entry

First thing you need to decide is how you plan to deploy the trading system. You can either enter the order manually, or use Order Interface (see ***Order Interface*** (on page 735)) to automatically send the order to your broker.

Manual Entry

With manual entry, you enter the order manually, following the signals from the trading system. You can observe the signal directly in a chart (see ***Trading System Markings*** (on page 1185)), or you can use System Monitor to guide you (see ***System Monitor Operation Guide*** (on page 939)).

To enter orders, you can either use the program your broker provides, or you can use the order forms or shortcut keys to manually enter the order (see ***Manual Order Placement*** (on page 705))

Pros:

- You can intervene the order when factors outside of pure price movements are affecting the market.
- You can intelligently resolve position discrepancy (see ***Issue: Position Mismatch*** (on page 791)).
- With System Monitor, it works with all brokers, not just the one with an interface.

Cons:

- You have to type in the orders. So it is less suitable for systems that trade frequently.

Automatic Entry

With automatic entry, trading system orders are sent directly to Order Interface (see ***Order Interface*** (on page 735)). Order Interface Manager will route the order to the Broker Interface (NT Order Server), which will in turn send the order to your broker.

In the time chart that hosts the trading system, you need to enable broker order placement (see ***Brokerage Order Placement*** (see "Sending Orders to Your Broker" on page 1203)).

In Order Interface, decide the order confirmation type you want for trading systems ***(see Order Confirm***ation (on page 738)). If you enable confirmation, you need to confirm orders with Account Manager. Otherwise, orders are sent directly to your broker.

Pros:

- Less typing. So fewer mistakes especially with limit orders.
- Fully automatic system requires little supervision.

Cons:

- Extra attention is required to resolve position discrepancy (see ***Issue: Position Mismatch*** (on page 791)).

- Difficult to intervene the order when factors outside of pure price movements are affecting the market.
- A broker interface is required. Not all brokers support it.
- For higher time frame trading systems, automation does not save you much.

Real-Time Testing

Before you deploy a trading system, you should test run it in real-time. Real-time testing allows you to try the trading system in a realistic setting. You can discover issues such as position discrepancy (see ***Position Discrepancy*** (see "Issue: Position Mismatch" on page 791)) that is not revealed in historical testing.

While real-time testing is big step forward in terms of realism, real-time testing is still a simulation. It cannot reveal all possible problems in a trading system.

Testing Length

In real-time testing, you do not have the luxury of compressed time. It is unlikely you can cover as much time as in historical testing. As a system writer, you need to ensure there are enough data points (positions) to collect meaningful statistics.

How long you should conduct real-time testing depends on the trading system. It is the number of positions that matters. Each position has a chance of revealing problems in the trading system. By having large number of positions, you can reduce the probability of trading system design error.

Like any scientific test, even if there is no problem found during real-time testing, it is not a positive prove that there is no problem in the trading system.

Using Broker Demo Account to Conduct Real-Time Testing

Some broker offers demo accounts. You can use a demo account to conduct real-time testing. The only problem is many of these demo accounts work with delay data, and will not be in sync with the trading system, which is using real-time data from your real-time data vendor.

Using Trade Simulator to Conduct Real-Time Testing

In many cases, it is better to use NeoTicker®'s Trade Simulator (see ***Trade Simulator*** (on page 755)) to conduct real-time testing. The advantages over broker demo accounts are:

- Your data feed provides the data. If you are using real-time data, there is no delay.
- Statistics are collected can be viewed in System Performance Viewer.

Test: Broker Demo Account and Manual Order Entry (Broker Provided Program)

Conduct this test if you plan to run the trading system and enter the orders manually in the broker provided program.

Procedure:

1. Set up NeoTicker® as you would use it trading, including the trading system you want to test.
2. Read the signals from the trading system, directly from a time chart or from system monitor.
3. Enter the order directly in the program provided by the broker.

4 Observe things such as positions, fill price, fill status. Write down anything that you think that is not expected.

5 After you collect enough data. Analyze the data and check if there is any problem in the trading system. If so, fix the problem and re-conduct the test.

Test: Broker Demo Account and Manual Order Entry (Broker Interface)

Conduct this test if you plan run the trading system and enter the orders manually in NeoTicker® with order forms or F2/F3 shortcuts.

Before you conduct this test, you need to configure Order Interface to route orders to the demo account (see ***Connecting to a Real-life Broker*** (on page 736)).

Procedure:

1 Set up NeoTicker® as you would use it trading, including the trading system you want to test.

2 Read the signals from the trading system, directly from a time chart or from system monitor.

3 Use Order Forms or F2/F3 shortcuts to enter orders.

4 Observe things such as positions, fill price, fill status. Write down anything that you think that is not expected.

5 After you collect enough data. Analyze the data and check if there is any problem in the trading system. If so, fix the problem and re-conduct the test.

Test: Broker Demo Account and Automatic Order Entry

Conduct this test if you plan run the trading system and use automatic order entry.

Before you conduct this test, you need to configure Order Interface to route orders to the demo account (see ***Connecting to a Real-life Broker*** (on page 736)).

Procedure:

1 Set up NeoTicker® as you would use it trading, including the trading system you want to test.

2 Configure the trading system to send orders automatically (see ***Brokerage Order Placement*** (see "Sending Orders to Your Broker" on page 1203)).

3 Let the chart that hosts the trading system run.

4 When an order fired by the trading system, you may need to confirm it in Account Manager.

5 Observe things such as positions, fill price, fill status. Write down anything that you think that is not expected.

After you collect enough data. Analyze the data and check if there is any problem in the trading system. If so, fix the problem and re-conduct the test.

Test: Trade Simulator and Manual Order Entry

Conduct this test if you plan run the trading system and enter the orders manually.

Before you conduct this test, you need to configure Order Interface to route orders to Trade Simulator (see ***Using Trade Simulator with Order Interface*** (on page 736)).

Procedure:

1. Set up NeoTicker® as you would use it trading, including the trading system you want to test.
2. Read the signals from the trading system, directly from a time chart or from system monitor.
3. Use order form or F2/F3 shortcuts to enter orders.
4. Observe things such as positions, fill price, fill status. Write down anything that you think that is not expected.
5. After you collect enough data. Analyze the data and check if there is any problem in the trading system. If so, fix the problem and re-conduct the test.

Test: Trade Simulator and Automatic Order Entry

Conduct this test if you plan run the trading system and use automatic order entry.

Before you conduct this test, you need to configure Order Interface to route orders to Trade Simulator (see ***Using Trade Simulator with Order Interface*** (on page 736)).

Procedure:

1. Set up NeoTicker® as you would use it trading, including the trading system you want to test.
2. Configure the trading system to send orders automatically (see ***Brokerage Order Placement*** (see "Sending Orders to Your Broker" on page 1203)).
3. Let the chart that hosts the trading system run.
4. When an order is fired by the trading system, you may need to confirm it with Account Manager.
5. Observe things such as positions, fill price, fill status. Write down anything that you think that is not expected.
6. After you collect enough data. Analyze the data and check if there is any problem in the trading system. If so, fix the problem and re-conduct the test.

Issue: Limit Order Expectation

Problem

In historical testing, limit orders are filled when the price has been hit. The result from historical testing can be unrealistic. When you real-time test a system issuing limit orders, you should observe the filling price and check if filling price has a pattern that is different from historical testing.

Resolution

Rewrite the trading system to send limit order at less aggressive price.

Issue: Position Mismatch

Problem

This problem happens when the trading system uses a long/short order to change position, and the order is filled, another long/short order is issued. Because the position is ambiguous before the first order is filled, it may cause the position in the trading system differs from the position in the brokerage.

This problem typically happens when a trading system covers a position with limit order, then reverse in direction using long/short orders. For example, if the current position is long 2 contracts, when the trading system issues a long exit limit order, this order will be translated to an order buy 2 contracts to the Order Interface. Before the order is filled, a short order is issued. To the short order, the position is ambiguous because it is not known that the first order will be successfully filled. The short order thus will issue a long covering order, which may result in an oversold scenario.

When you real-time test, watch out for position mismatch warning issued by Order Interface.

Resolution

Several technique exists. You can use a combination of these to resolve this problem:

- For automatic order entry, Account Manager and trading system settings have options to help you resolve position mismatch. See ***Position Size Matching*** (on page 745).
- For manual order entry, you can inspect the orders to see if the they will result in position discrepancy.
- Modify the limit order to market order to reduce the chance of overlapping orders
- Replace long/short orders with buy/sell orders to eliminate ambiguity. You will need extra code to monitor positions.
- Rewrite trading system to spread out the orders to reduce the chance of overlapping orders

Issue: Single Entry Per Direction Inconsistency

Problem

This is a specialized case of position mismatch when Single Entry Per Direction is activated in a system. The problem happens when brokerage feedback is delayed and trading system position becomes inconsistent with the Single Entry Per Direction setting.

This problem can happen in live deployment even NeoTicker® takes all the precautions to block new orders and cancel duplicate orders.

Prevention

You can detect the problem in design time by testing the system with Trade Simulator. While this does not guarantee 100% detection, you can detect many instances and fix the system logic accordingly.

Resolution

1. Use Account Manager to manually sync the position, or
2. In the system, use Long/Short orders instead of Buy/Sell.

Going Live

Unless your trading system is specifically written to properly handle tick data, do not turn on update-by-tick for a trading system.

If you decided the trading system is ready to go live, you can easily modify your real-time testing for live trading. You should always closely monitor your live trading system to observe any abnormal behavior.

Manual Order Entry through Broker Program

The setup is almost identical to real-time testing, but instead of a demo account or Trade Simulator, you will trade with the broker with a real account.

Manual Order Entry through Order Interface

The setup is almost identical to real-time testing. You manually place order by order forms or F2/F3 shortcuts.

You will need to configure Order Interface to route the orders to your broker's interface. See ***Connecting to a Real-life Broker*** (on page 736).

Automatic Order Placement

The setup is almost identical to real-time testing.

You will need to configure Order Interface to route the orders to your broker's interface. See ***Connecting to a Real-life Broker*** (on page 736).

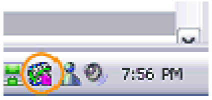
NT Order Server for Interactive Brokers Reference

NT Order Server for Interface Broker is the Broker Interface that places order to IB. If you configured NeoTicker® to place order to IB, this program is automatically launched.

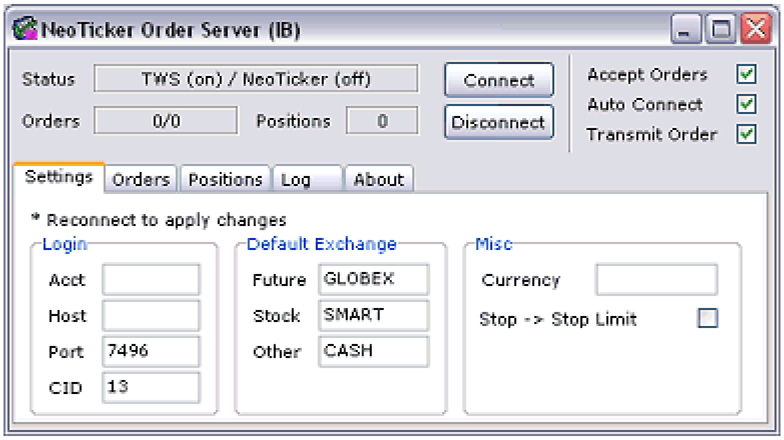
For information on how to set up an IB connection, see ***Configuring Order Placement to Interactive Brokers*** (on page 51).

Most of the time you do not have to touch NT Order Server for IB at all. Things will just work. If you have trouble connecting to IB, chances are you should reconfigure Order Interface Setup or TWS.

After NeoTicker® launched NT Order Server for IB, you can find the program icon in the lower right corner of your desktop, near Windows' clock.



Double click the icon to open NT Order Server for IB.



Status: TWS (on) / NeoTicker (off)

Orders: 0/0 Positions: 0

Accept Orders ☑

Auto Connect ☑

Transmit Order ☑

* Reconnect to apply changes

Login

Acct:

Host:

Port: 7496

CID: 13

Default Exchange

Future: GLOBEX

Stock: SMART

Other: CASH

Misc

Currency:

Stop -> Stop Limit □

Top Area Reference

Control	Function
Status	Connection status.
Orders	Number of orders handled by NT Order Server.
Positions	Number of positions tracked by NT Order Server.
Connect	Connect to broker.
Disconnect	Disconnect from broker.
Accept Orders	Accept order placement from Order Interface.
Auto Connect	Auto connect to broker when start.
Transmit Orders	TWS auto transmit order.

Settings Tab

Control	Function
Acct	If you have multiple IB accounts, specify the account NT Order Interface will place order to.
Host	If TWS resides on another computer, enter computer's IP address here.
Port	Port number for communication between NT Order Server and TWS
CID	TWS requires a client ID. Do not change this value.
Default Exchange	Default exchange to place order when Order Interface fails to include exchange information.

Currency If the instrument you trade is in a different currency from your base currency, specify your base currency here.

Stop -> Stop Limit Coverts stop orders to stop limit orders.

Orders Tab

This tab lists all orders sent to NT Order Server.

Positions Tab

This tab lists all positions tracked by NT Order Server.

Log Tab

Message log between NT Order Server and broker.

About Tab

Product information for NT Order Server.

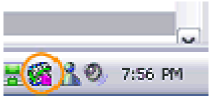
NT Order Server for MB Trading Reference

NT Order Server for MB Trading is the Broker Interface that places order to MB Trading. If you configured NeoTicker® to place order to MB Trading, this program is automatically launched.

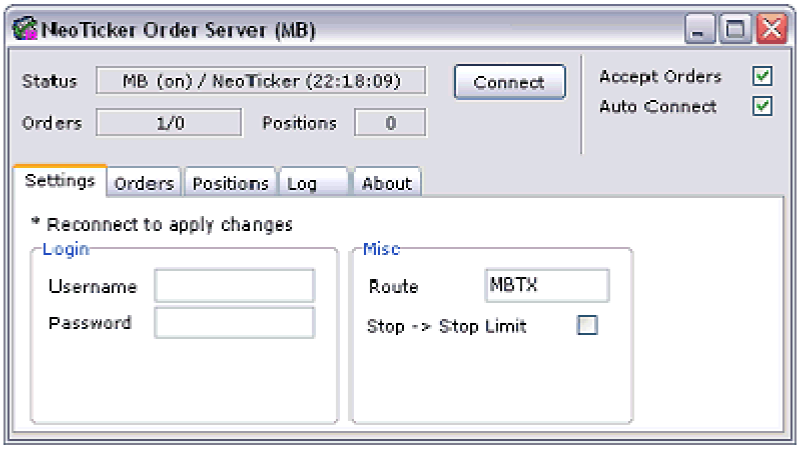
For information on how to set up an MB Trading connection, see ***Configuring Order Placement to MB Trading*** (on page 53).

Most of the time you do not have to touch NT Order Server for MB Trading at all. Things will just work. If you have trouble connecting to MB Trading, chances are you should reconfigure Order Interface Setup.

After NeoTicker® launched NT Order Server for MB Trading, you can find the program icon in the lower right corner of your desktop, near Windows' clock.



Double click the icon to open NT Order Server for MB Trading:



Status: MB (on) / NeoTicker (22:18:09)

Orders: 1/0

Positions: 0

Accept Orders ☑

Auto Connect ☑

* Reconnect to apply changes

Login

Username

Password

Misc

Route: MBTX

Stop -> Stop Limit □

Top Area Reference

Control	Function
Status	Connection status.
Orders	Number of orders handled by NT Order Server.
Positions	Number of positions tracked by NT Order Server.
Connect	Connect to broker.
Accept Orders	Accept order placement from Order Interface.
Auto Connect	Auto connect to broker when start.

Settings Tab

Control	Function
Login	If you fill in **Username** and **Password**, NT ORder Server will auto login to MBT Navigator.
Route	Which ECN should process the order. Use MBTX unless you have a specific ECN you want to specify.
Stop -> Stop Limit	Coverts stop orders to stop limit orders.

Orders Tab

This tab lists all orders sent to NT Order Server.

Positions Tab

This tab lists all positions tracked by NT Order Server.

Log Tab

Message log between NT Order Server and broker.

About Tab

Product information for NT Order Server.

Pattern Scanner Operation Guide

Introduction

NeoTicker®'s pattern scanner is a breakthrough in scanning technology. It combines all possible types of filtering capabilities into one unified model.

Pattern scanner is a very general tool. For real-time scanning for a small set of symbols (within your symbol limit, typically several hundred), consider using quote window as an alternative. Read ***Tutorial: Creating a Great Time Chart*** (on page 105) for an example on how to use quote window to scan.

Server Side Scanning vs. Pattern Scanner

Some data feed provides symbol lists that satisfy certain criteria and called this a scanning feature. The terminology is confusing because the scanning really happens on the data feed's servers and you have very little control over the scanning criteria. Pattern scanner is a different type of technology that perform scanning on your computer, allowing much greater flexibility in scanning criteria.

To access data feed provided scanning, use the top list window. Refer to ***Top List Operation Guide*** (on page 1227) for more information.

Working with One Pattern Scanner

When a pattern scanner is used on a standalone basis, it is a complete scanning tool with many features.

You can use data from your real-time data feed and with historical data of various data formats.

You can filter by current price data, pattern match, or even multiple indicators based on any time frame. There are pre-built filters for fast access.

You can also rank the scanning results with various ranking method.

You can scan on a timer basis, say, every 5-minute, using your criteria.

The resulted symbol list can be saved as a symbol list for use by other function windows in NeoTicker®, or even used by another program. The resulted list can also be directly sent to a quote window of your choice for immediate monitoring in real-time.

Working with Multiple Pattern Scanners

When a pattern scanner is used with another pattern scanner, you can create very complex scans based on multiple time frame.

You can instruct a pattern scanner to chain its scanning result to another scanner for further filtering, or just ranking which a different set of criteria.

You can also instruct a pattern scanner to get scanned results from another pattern scanner as input to launch its own scanning, which is very useful if the other scanner is based on a higher time frame and requires less frequent scanning during the day.

You can even chain multiple pattern scanners together for even more complex scanning criteria.

Think of each pattern scanner window as a building block. You can build your real-time data mining model easily with each block doing part of the job for you.

Working With Other Function Windows

The pattern scanner window is designed to work as a part of NeoTicker®, not as an isolated tool. There are many features in pattern scanner making it fully integrated with the rest of NeoTicker®.

First, you can simply send the symbols from the result list of a pattern scanner to another function window by right click on it, and choose which function window to switch to use that particular symbol.

Second, you can double click a symbol in the result list to send it to all the function windows within the current group that is part of the chain.

Third, you can send symbol list to a window that can accept symbol list (such as quote window) for further analysis.

Methodology

Follow the Tutorial First

You can follow the example given in ***Tutorial: Constructing a Scan using Pattern Scanner*** (on page 199). This will give you a taste of how pattern scanner works before you read the rest of this section.

Introduction

In a pattern scanner, you specify symbols, data source, scanning criteria as input and the pattern scanner produces the symbols that match the scanning criteria as output. The output can be viewed as it is, or you can further process the output. For example, you can tell pattern scanner to send the output to another pattern scanner as input.

Symbols

Pattern scanner is designed to identify what symbols are fulfilling your criteria. Thus the input is simply which symbols to scan through. Pattern scanner offers many choices to choose from for your convenience.

- You can choose from a symbol list file.
- You can choose to scan all the symbols that are being tracked in real time or, if your data source is set to something other than your real time data, then all the data within that directory.
- You can choose to scan all the symbols from within a quote window.
- You can choose to scan from the output obtained from another pattern scanner.

Data Source

Two things are specified here: the data server and how you intend to format the data for scanning.

For data server, you can use:

- Your real-time data feed
- Disk cache
- EOD sources such as Internet, an EOD data feed such as Quotes Plus, TC2005, data files, etc.

Data format is important if you intend to use the data for indicator calculation and visual pattern matching. You will have to provide information about the bars (1-min, 5-min, etc), trading time and holiday list.

For example, if you define your data source to be daily data, then if you specify the indicator simple moving average of 50 period. You are effectively looking at the daily simple moving average. When you modify the data source to 5-minute bar data, then you are working with 5-minute bar simple moving average.

Basic Filters

Basic filters are criteria that you specify as the first level of filtering. Basic filters are fast and can reduce the number of symbols drastically before complex criteria is applied.

For example, you can filter out many symbols if you specify the current price must be larger than 50. Than symbols with a price less than 50 will be filtered out and not included when more complex criteria is applied.

Within pattern scanner the Symbol filters, Historical filters, Current filters and Lookup filters are basic filters.

Complex Criteria

Complex criteria is the combination of the pattern matching tool and a set of indicators with filter rules.

Pattern matching tool works on its own with one output value - % confidence. It takes a pattern file, which is defined from a section of your data series in a time chart, and match that against the price series of each symbol to be scanned and provide an objective score.

Indicator values can be used as a filter (e.g. RSI must be larger than 50). The output of the indicator can be used as a ranking criteria. Pattern scanner can evaluate multiple indicators per symbol in a scan. Pattern scanner supports the indicator-on-indicator architecture and complex indicator setup can be imported from a time chart.

One of the hardest thing to master in the pattern scanner is how to combine the indicators into one or more pattern scanners such that the scanning time is minimized. We will discuss in details the techniques of optimization in a separate section.

Output

Once a scan is done, symbols that pass all filtering criteria are displayed in the pattern scanner. You can rank the symbols by confidence level, indicator value, etc.

This is the primary result of a single pattern scanner.

Post Processing

Pattern scanner offers a few ways to post process the scanned results.

- You can save the resulting symbols into a symbol list file.
- You can also send the result to a quote window automatically.

- You can send the results to another pattern scanner to automatically start that pattern scanner.

Timer Control

A fully designed pattern scanner can be used productively in real-time when it can automatically start its scanning without human supervision.

Pattern scanner offer many different ways to auto start itself.

You can set it to scan on a fixed time interval like every 5 minute.

You can set a specific time of the day to do the scan once like 1:00 PM everyday.

Basic Operations

Setup

To open the setup window for pattern scanner, choose **Setup** from the pop up menu.

Starting/Stopping Scanning

To start scanning, choose **Scan Now** from the pop up menu.

To stop scanning, choose **Stop Now** from the pop up menu.

You can schedule pattern scanner to scan on timer. For more information, refer to ***Scanning on Timer*** (on page 809).

Scanner Completion Report

When scanning is completed, pattern scanner returns a scanning report that looks like:

```
NET 53/RT 53/HIST 54/SYM 49200
```

This line provides valuable information to help you optimize the scanning performance. It is a summary report of how each type of filters has filtered the symbols.

`SYM` is the number of symbols left after the symbol filters (**Symbol** tab in pattern scanner setup) have been applied. In this example, there are 49200 symbols left.

`HIST` is the number of symbols left after lookup filters (**Lookup** tab in pattern scanner setup) have been applied. In this example, there are 54 symbols left.

`RT` is the number of symbols left after the current filters (**Current** tab in pattern scanner setup) have been applied. In this example, there are 53 symbols left.

`NET` is the number of symbols that are feed to indicator and visual pattern filters.

To optimize the scanning performance, you can reduce the number of symbols from right to left. The filters on the right hand side are more efficient then filters on the left hand side. When the number of symbols are reduced on the right, fewer symbols require the more time consuming filtering on the left.

Selecting an Input Symbol List

1 Open the setup window.

2 Press the **Options** tab.

3 Choose one of the ways of importing symbol list.

4 Apply the changes.

If you choose **All Current Symbols**, you are scanning the complete universe of symbols based on the data source context. If data source is your real time server, then the universe is all the symbols you are tracking in real time at the moment. If data source is Quotes Plus or TC2005, then it is all the symbols specified under the Group/Directory setting. If data source is a file based data format, then it is all the available files listed in the directory.

You can also import symbol list from another window, choose **Pattern Scanner**, **Quote Window** or **Top List** and enter the name of the window.

In general, **All Current Symbols** is practical for satellite feeds and file-based data sources. For Internet feeds, because the current set of symbols can be relatively small, it is preferable use other options.

Ranking Scanning Output

The scanning output is ranked. To change ranking settings:

1 Open the setup window.

2 Press the **Options** tab.

3 Modify the ranking settings under **Ranking**.

4 Apply the changes.

Processing Scanning Output

You can instruct the scanner to automatically save the scanning output to a file or send it to a quote window.

1 Open the setup window.

2 Press the **Options** tab.

3 Settings are under **Output**.

Another possibility is to send the output to another pattern scanner, and triggers that pattern scanner. This is called chain scanner. Refer to ***Introduction to Chaining*** (on page 815) for more information.

Scanning on Timer

To set up pattern scanner to scan on timer:

1 Open the setup window.

2 Press the **Schedule/Misc** tab.

3 Enable timer scanning by checking the **Timer** option.

4 Fill in the settings under **Timer**.

5 Apply the changes.

Timer is only active only within the time period specified. To scan outside of the specified time, you must manually start scanning.

Scanning Completion Notice

Pattern scanner can notify you by sound, pop up window or alert when scanning is done.

To set up:

1 Open the setup window.

2 Press the **Schedule/Misc** tab.

3 The settings are under **Completion Notification**.

4 Apply the changes.

Data Source

You need to tell pattern scanner where the data is coming from. To set up:

1 Open the setup window.

2 Press the **Data** tab.

3 The settings are under **Source**.

4 Apply the changes.

If **Data Type** is set to **RT Server**, you will be scanning using your real-time data feed. This means the scanning result will reflect the latest changes in real-time data. If historical data is required, it will be download automatically from the real-time feed.

When scanning use **RT Server**, for each symbol, the latest data is requested from the real-time data feed. If you use Internet data feed, you should read ***Scanning with Internet Data Feed*** (on page 834). The section will tell you what to expect and how to improve scanning performance.

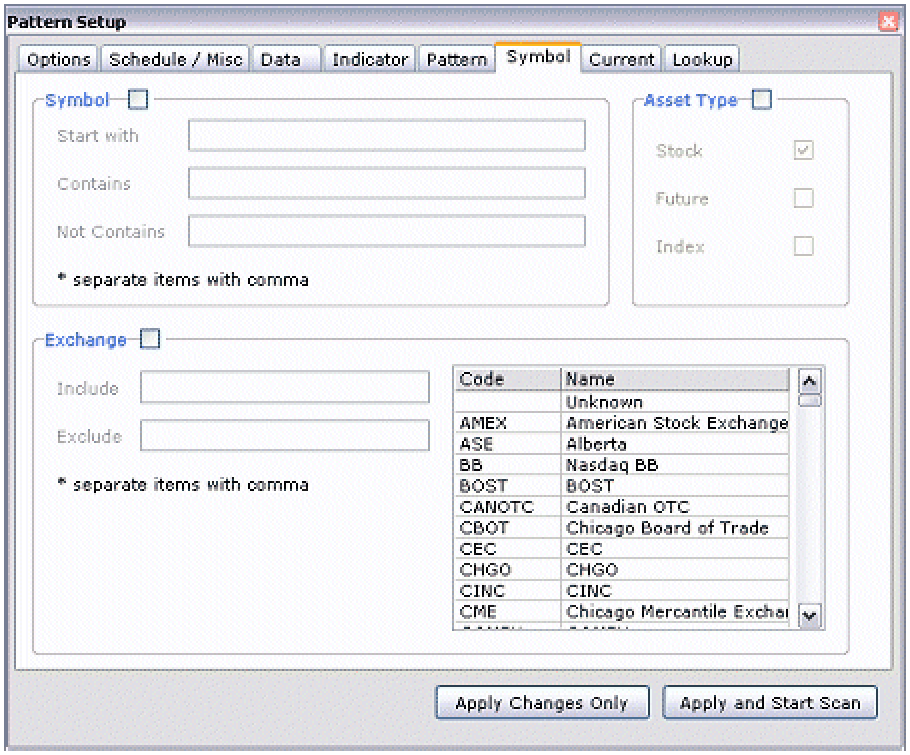
For other **Data Type** settings, refer to ***Scanning without Real Time Server*** (on page 836).

Symbol Filter

Symbol filter will filter out symbols that do not match the given description. Symbol filtering can greatly improve the scanning speed since it performs the filtering before any real-time processing. Symbol filter is most useful for satellite feed users who wish to filter out symbols (such as mutual fund data) in a full market scan.

Availability of exchange and asset type filter is data feed dependent. Some data feed does not provide exchange and asset type information.

To set up symbol filter, press the **Symbol** tab in pattern scanner setup.



Symbol ☐

Asset Type ☐

Stock ☑

Future ☐

Index ☐

Exchange ☐

Code	Name
	Unknown
AMEX	American Stock Exchange
ASE	Alberta
BB	Nasdaq BB
BOST	BOST
CANOTC	Canadian OTC
CBOT	Chicago Board of Trade
CEC	CEC
CHGO	CHGO
CINC	CINC
CME	Chicago Mercantile Exchai

Symbol

This filter only accepts symbols with specific character pattern. You can enter multiple patterns separated by either space or comma in all three edit areas

Exchange

You can enter exchange codes here as filters. A list of the exchange codes are listed as a reference.

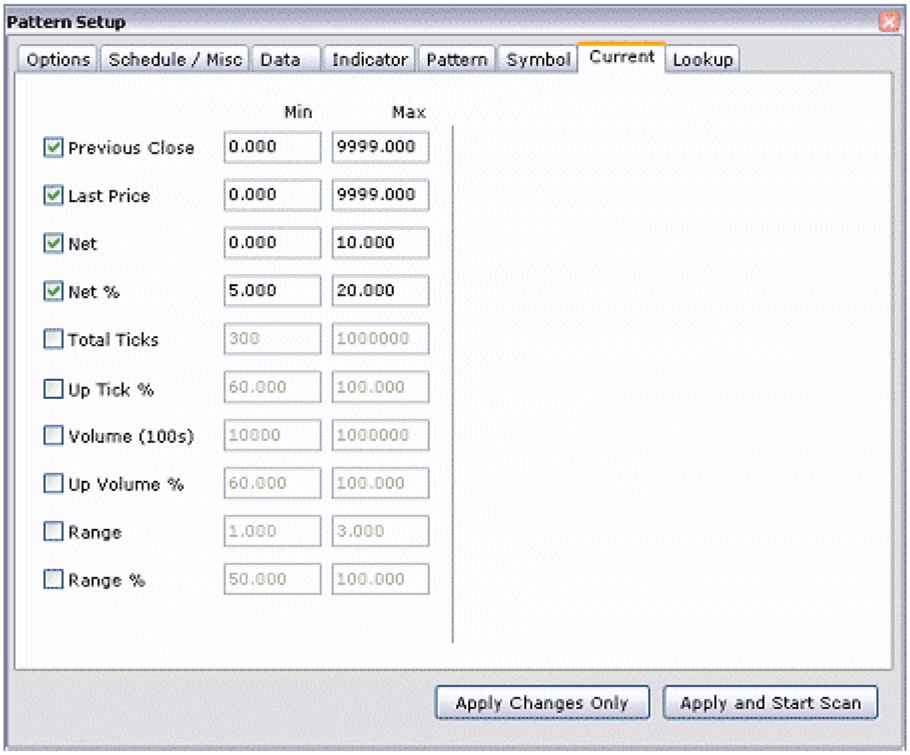
Asset Type

You can choose to accept only certain types of asset.

Current Filter

Current filters perform filtering based on current data fields. For example, last traded price and volume. Current filters are high performance filters than can quickly weed out unwanted symbols.

Current filters are setup under the **Current** tab in pattern scanner setup.



	Min	Max
☑ Previous Close	0.000	9999.000
☑ Last Price	0.000	9999.000
☑ Net	0.000	10.000
☑ Net %	5.000	20.000
□ Total Ticks	300	1000000
□ Up Tick %	60.000	100.000
□ Volume (100s)	10000	1000000
□ Up Volume %	60.000	100.000
□ Range	1.000	3.000
□ Range %	50.000	100.000

Caution with Tick Related Filters

The **Total Ticks**, **Up Tick %** and **Up Volume %** filter's behavior depend on whether you have the real-time lookup table enabled (the setting is under **Lookup DB** in server setup). If real-time lookup table is enabled, pattern scanner uses the real-time lookup table to resolves these fields for filtering. If real-time lookup table is not enabled, pattern scanner relies on the real-time data vendor to supply values for these fields for filtering.

The real-time lookup table uses an up-down tick definition to calculate **Up Tick%** and **Up Volume%**. If you do not use the real-time lookup table, you should note that there is no commonly agreed definition for these values among data vendors. You should find out from your data vendor whether an up-down-side or an up-down definition is used.

CPU Utilization

You can set refine the impact of CPU load caused by the pattern scanner:

1 Open the setup window.

2 Press the **Schedule/Misc** tab.

3 Choose one of the **Priority** under **CPU Utilization**.

4 Apply the changes.

If you set **Priority** to highest, pattern scanner will complete its task in the shortest time but may affect a slow computer, making you feel like the mouse is less responsive.

If you set **Priority** to lowest, pattern scanner will complete its task in the longest time and your computer will be more responsive to other tasks. If you are doing some after hours scanning while using NeoTicker® to browse through charts, etc. You may want to use this setting as the scanner performance is less critical.

Historical Pattern Scanning

You can specify a last date. When criteria are evaluated, data passed the last date will not be considered. This way, you can specify criteria such as finding all stocks that have a slowd larger than 70 on September 30, 2003.

To specify the end date:

1 Open the setup window.

2 Press the **Data** tab.

3 The settings are under **Last Day**.

4 If you choose **Most Recent Data**, the pattern scanner will scan to the most recent data, i.e. real-time scanning. If you choose **Specific Date**, the pattern scanner will only scan to the date specified, i.e. historical scanning.

5 Apply the changes.

Saving Scanning Result

When you save the pattern scanner window, your current scanning result can be saved along with the pattern scanner. To enable/disable this feature:

1 Open Pattern Scanner setup.

2 Under **Schedule/Misc** tab, in **Last Scanning Result** box, choose **Save with settings/Discard on close**.

Sending Scanning Result to Excel

You can specify an Excel workbook and worksheet as a destination to send scanning result. To enable this feature:

1 Open Pattern Scanner setup.

2 Under **Options** tab, in **Output** box, choose **Excel** and specify **Workbook** and **Worksheet**.

Sorting Scanning Result

You can sort scanning result. Simply press the header of the column you want to sort in scanning result.

Eliminating Real-time Symbol Usage in Scanning

Pattern scanning can temporary increase your real-time symbol limit. This is caused by pre-filtering and the final quote of scanning result. If your setup is close to your symbol limit, pattern scanning can cause problems.

You can eliminate usage of real-time symbols if you turn off the following features:

- In pattern scanner setup, under **Options** tab, turn off **Always Include Last Price Column**. After you turn off this option, quote price and net will not be included in the scanning result. So your scanning cannot use these fields for ranking.
- In pattern scanner setup, under **Symbol** tab, turn off all filtering under this tab.
- In pattern scanner setup, under **Current** tab, turn off all filtering under this tab.
- In pattern scanner setup, under **Lookup** tab, turn off all filtering under this tab.

In another words, you can perform pure indicator or pattern scanning, based on single or multiple symbols, but you cannot use filtering to help you eliminate scanning candidates.

Launching External Application After Scan

You can launch an external application after scanning completes:

To set up:

1. Open the setup window.
2. Press the **Schedule/Misc** tab.
3. Under **Completion Notification**, specify the application.
4. Apply the changes.

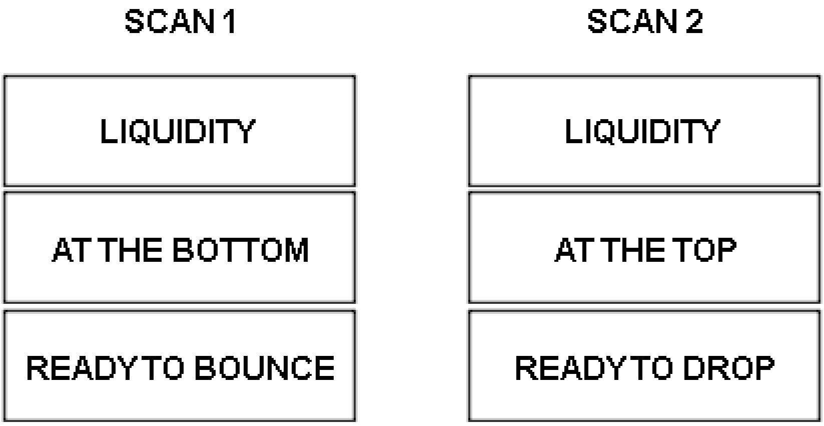
You can set the scanner to output to a symbol list file (under setup's **Options** tab). The external application can then read the symbol list files as input.

Introduction to Chaining

One of the most powerful feature of pattern scanner is its ability to chain with another scan. Why would anyone wants to chain their pattern scanners together? There are two answers. First is easier to update and improve. Second is speed.

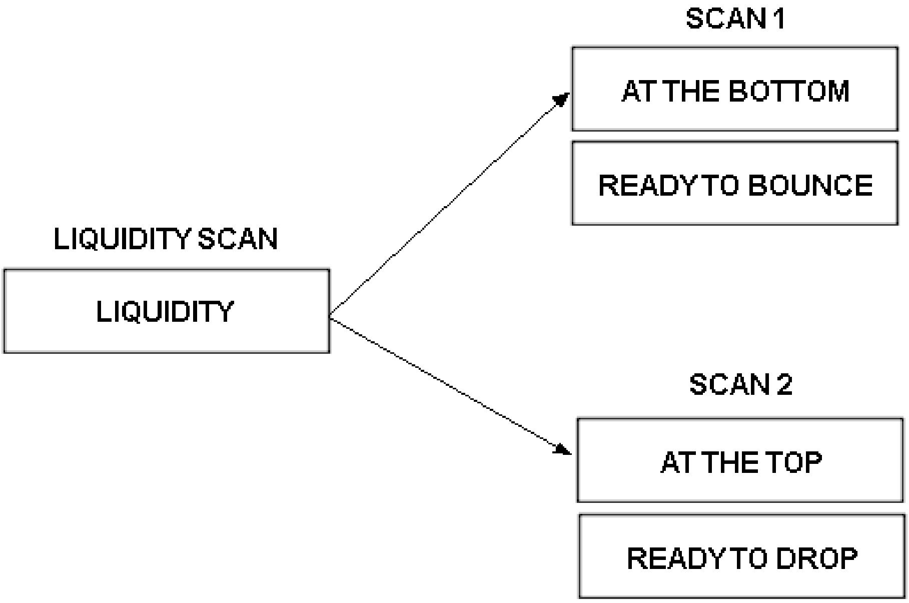
Easier To Update

For example, you designed a scan to look for tradable symbols based on a set of indicators to qualify a stock is liquid enough, at its bottom and ready to bounce. Then you also have this other pattern scanner to look for tradable symbols with the same basic set of indicators to qualify a stock is liquid enough, at its top, and ready to drop.



If you are going to adjust your basic rules of liquidity, say from at least 3 million shares traded in the morning to 5 million shares traded, then you will need to modify both scanners to get them updated with your latest criteria change.

If instead you have a pattern scanner set up to scan for liquidity only. And then the 2 tradable scans we have talked about are changed to scan from the results of this liquidity scan, you do not need to modify the 2 scans that look for tradable symbols as once the liquidity scan is modified, so will the results of the 2 scans.



You can think of scans as building blocks that you can easily combine together to perform any scanning task you want. It is always easier to build complex scans from simpler ones.

Speed

Using the same example from previous paragraphs, the liquidity scan has actually archived yet another goal.

The liquidity scan reduced the symbols to a smaller subset of the original list which makes the 2 scans spending less time to look for what you want. Plus the calculation of the indicators related to defining liquidity is calculated only once instead of the original way of doing that twice, thus saving computation time as well.

Forward Chaining

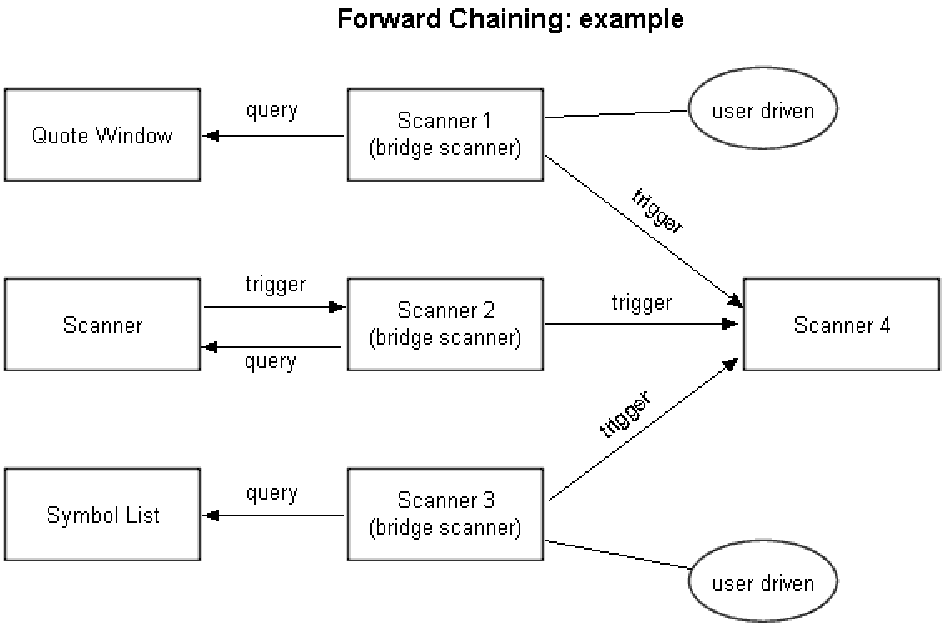
Forward chaining is referring to the method of driving through a list of pattern scanners one by one. Where after each scan, the resulting list of symbols is reduced, and then the next scanner is invoked to further reduce the number of symbols.

This method ensured the next pattern scanner always uses the most updated result from the current pattern scanner. Thus this method is good for criteria that are highly dependent on the coexistence of the conditions you have put into each pattern scanner.

For better reuse of system resource, there is a trick that you can employ to reuse scanning result from a separate pattern scanner which is not part of the chain.

This technique is called a bridge scanner which does nothing but just copying the input symbols into its output. In this case, you specify this scan to use the result from another pattern scanner like the liquidity scan and do not set any rules to filter the symbols. This way, all the symbols will then be available for the next pattern scanner in the chain.

So, you can have a few basic or core pattern scanners that keep monitoring the symbols that fulfill your basic tracking requirement. Then use a bridge scanner to start a chain of scanners to locate your symbols.



Lets go through the forward chaining example in the diagram above. The dialog shows that how Scanner 4 is being forward chained from Scanner 1, Scanner 2 and Scanner 3. In all three cases, Scanner 4 always get the latest result as input.

Scanner 1 is user driven (e.g. you manually start it). When Scanner 1 starts, it gets the input symbols from a quote window. Once Scanner 1 is done, it forward chain to Scanner 4 by sending the output to Scanner 4 and trigger Scanner 4 to start scanning.

Scanner 2 is a bridge scanner. Scanner 2 is triggered by another scanner. Scanner 2 does not do any scanning, and directly triggers Scanner 4.

Scanner 3 is user driven. When Scanner 3 starts, it loads the symbols from a symbol list. Once Scanner 3 is done, it forward chain to Scanner 4 by sending the output to Scanner 4 and trigger Scanner 4 to start scanning.

How to Specify Forward Chaining

Forward chaining is specified at scanner that triggers the next scanner. For example, in the forward chain Scanner 1 to Scanner 4, the forward chaining is specified in Scanner 1. In this example, you would:

1 Open the setup window of Scanner 1.

2 Press the **Options** tab.

3 Enable the option **Pass Result to Next Scan**.

4 Choose Scanner 4 in **Pattern Scanner to Launch**.

5 Apply the changes.

6 Make sure Scanner 4 is not automatically triggered by a timer.

Backward Chaining

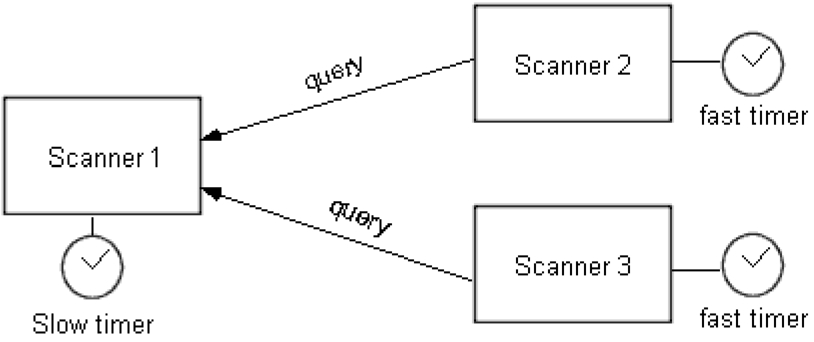
Backward chaining is referring to the method of reading the scan results from another pattern scanner as input.

This method does not ensure that the symbol list obtained from the pattern scanner is the latest result. Thus the 2 pattern scanners should not have strong coexistence relationship in their conditions.

Backward chaining is useful when the source symbol list generated is from a higher time frame filtering.

For example, you have a pattern scanner that filters on conditions based on data source of 30-min bars and the output is used as input to a pattern scanner that works on 5-min bars. Ideally, you want the 5-min bars to scan at least every 5 minute. However, it is not necessary for the 30-min bars to be scan this often. Backward chaining would allow you to use a slower timer on the 30-min bars and a faster timer on the 5-minute bars, reducing unnecessary scanning on the 30-min bars.

Backward Chain Example



The figure above show examples of backward chaining. Both Scanner 2 and Scanner 3 are backward chained to Scanner 1.

First Scanner 1 will start scanning on a timer, and produces scanning output.

Scanner 2 and Scanner 3 will use their own timer to start scanning. Scanner 2 and Scanner 3 will pull the output from Scanner 1 as their input for scanning.

How to Specify Backward Scanning

Let's use Scanner 1 and Scanner 2 in the example above. You will:

1 In Scanner 1, open the setup window.

2 Press the **Schedule/Misc** tab.

3 Check the **Timer** option.

4 Specify the time that you want Scanner 1 to be active.

5 Specify the scanning interval.

6 Apply the changes.

7 In Scanner 2, open the setup window.

8 Press the **Schedule/Misc** tab.

9 Check the **Timer** option.

10 Specify the timer that you want Scanner 2 to be active.

11 Specify the scanning interval.

12 Press the **Options** tab.

13 Under **Input**, check **Pattern Scanner**, and choose Scanner 1.

14 Apply the changes.

Tip: You may want to manually start the scanner with a slower timer. This way, you can ensure some scanning result is available for the scanner with a faster timer.

Scanning Speed Optimization

To enhance your scanning speed, you can first construct your idea in a all-in-one pattern scanner first. Then you can redesign the pattern scanner into multiple scanners to save time. There are a few basic techniques you may find useful.

Use Scan Results From Previous Trading Day

If your criteria include setups based on previous day, then define a new pattern scanner to do that first and save the result into a symbol list. Then you can easily reuse this list with your real-time pattern scanners as the input.

Use Basic Filters If Possible

Whenever possible, use basic filters to reduce the need to invoke pattern matcher and indicator computation. As both operations are more time consuming than basic filters.

Load Less Number Of Days

Check your indicators to see if you really need to load that many days of data. For example, you are filtering on the 5 period simple moving average, then there is no need to load more than 2 days of data at all.

The setting is under **Data** tab in setup window.

Reduce Indicator Bar Calculation

For intra-day data, even if you load only one day of data, there can be many minute bars and second bars. If you know you only need a few indicator bars to ensure accuracy (some indicator depends on previous bars), you can reduce re-calculation to only the necessary bars.

For example, it is safe to re-calculate only 10-15 bars for a 5-period simple moving average. The extra bars provide a safety net to handle invalid bars.

The setting is under **Indicator** tab in setup window.

Break Multiple Filtered Indicators Into Multiple Scans

If you find that you need to have multiple filters on multiple indicators within the same scan, you can re-arrange the pattern scanner into 2 or more forward chained scans. Remember that during each scan, the number of symbols left are drastically reduced thus saving you time in the indicator computation.

Make Slow Scans Available Through Backward Chaining

If you have some very slow scans due to its nature of heavy computation. You can make these scans work on a timer basis so you will have its scan results available for other pattern scanners to use.

Multiple Symbols Per Iteration Usage

NeoTicker® supports the unique feature of scanning for criteria based on multiple symbol combinations. This capability is very useful for many different type of applications such as spread scanning.

Spread Scanning

If you are a spread trader, you will want to find spread opportunities based on your pre-defined criteria. Before NeoTicker®, there is no way one can scan for spread conditions at all and you will need to monitor the pre-defined spreads say 20 to 30 of them all on pre-created charts in case one of them is matching your criteria. This classic way wastes both time and effort but is not that efficient as you will probably missed many spread combinations that is happening at the moment with your condition fulfilled. Plus, most classic software do not even handle multiple symbols within a chart anyway!

To scan for spreads in the pattern scanner, remember three rules of thumbs.

1. If possible set the multi symbol tab filter of scan only once per symbol combination. This way, a spread based on say ABC and DEF is scanned once only and the subsequent combination of DEF and ABC will be skipped automatically for you. You save a lot of time to scan for more symbol combinations.
2. Break your criteria into chain scans and keep the first scan as simple as possible. Remember that calculation based on indicators or pattern matching is slow. If you can limit your criteria to simple spread amount or percentage calculation based on say 2 days of daily data only, your first scan will be extremely fast and enable you to narrow down the choices in a chain scan.
3. If you know your spread combinations well and do not need to use the automated generated symbol combinations, then use your pre-defined spread list. To define a spread list is very similar to defining a symbol list except each line contains more than one symbol.

Dynamic Portfolio Optimization

If you trade a basket of symbols, then one of the very important tools for optimizing your system performance is to optimize the portfolio. You can easily do that by throwing in a basket of symbols through a symbol list to a pattern scanner. Then set the number of symbols per iteration to your basket size. You can then add your trading system indicator into the pattern scanner to start scanning. You can change the indicator output to anything you like to optimize on, say, net profit amount, win / loss ratio, etc.

If you like to filter the trading system results, you can save your performance data into separate plots of the indicator you have created. Then you can easily refer to different plots of the indicator using the built-in indicator PlotValue to indirectly access your trading system's information.

Correlation Driven Trading System or Option Trading Strategies

If your trading model is based on correlation behavior and need to pick trading target based on other securities, then you will want to use the multi symbol per iteration feature. Unlike the spread trading scenario, you do not want to look for trading opportunities based on trading 2 instruments at the same time. Instead, you could be looking for trading combinations based on 3 to 5 securities and would exercise your orders on 1 to 2 of them. Option traders looking for special option combinations mostly will find this setup very useful.

Specifying Scan Multiple Symbols Per Iteration

1 Open the setup window.

2 Press the **Options** tab.

3 Choose either **Predefined Multiple Symbol List** or **Symbols per Iteration**.

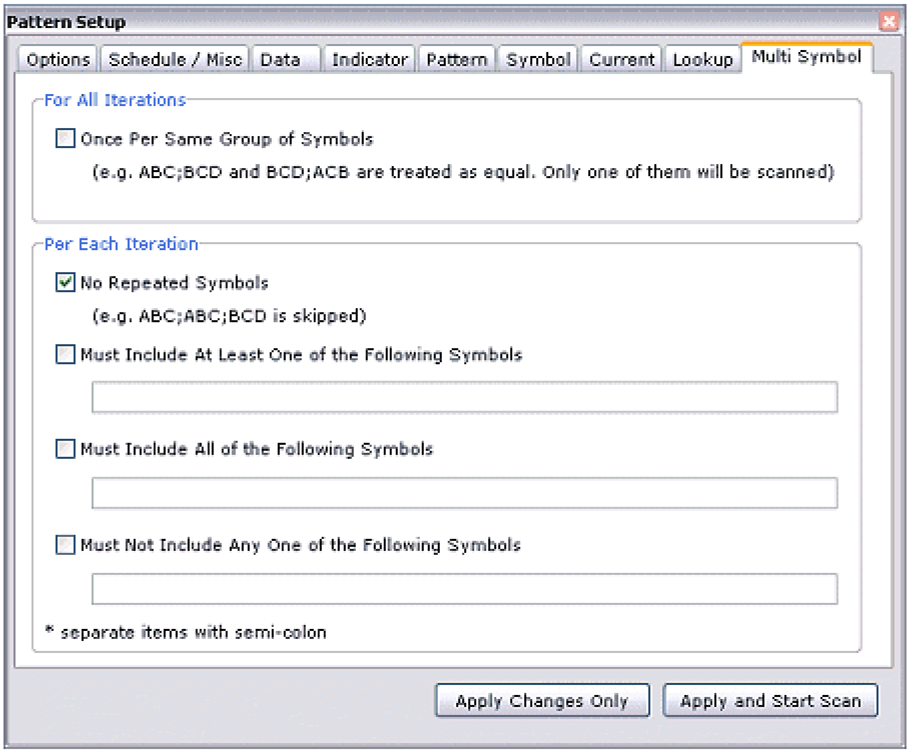
4 Apply the changes.

If you choose **Predefined Multiple Symbols List**, then you can specify a file containing listing of symbol combinations. To define a multiple symbol list file, you simply add your symbols to separate lines in the text file, with each line representing one iteration of scan. Within each line, separate the symbols with the semicolons. The following is a few lines of a multiple symbol list:

```
IBM;MSFT
CSCO;DELL
AAPL; MSFT
DELL; APPL
```

If you choose **Symbols Per Iteration**, pattern scanner will automatically auto generate all the combinations based on the symbols in a regular symbol list.

Additional multiple symbol settings are under the **Multi Symbol** tab in the setup window. The tab appears only after you select one of the multiple symbol option.



For All Iterations

□ Once Per Same Group of Symbols

(e.g. ABC;BCD and BCD;ACB are treated as equal. Only one of them will be scanned)

Per Each Iteration

☑ No Repeated Symbols

(e.g. ABC;ABC;BCD is skipped)

□ Must Include At Least One of the Following Symbols

□ Must Include All of the Following Symbols

□ Must Not Include Any One of the Following Symbols

* separate items with semi-colon

Setting	Meaning
Once Per Same Group of Symbols	Check this if the order of the symbols is not important. Only one combination of the symbols will be scanned.
No Repeat Symbols	This option is checked by default. Check this option if you do not want to scan a combination of symbols than contains repeated symbols. For example, if you are scanning a spread, you do not want to waste time scanning a symbol against itself.
Must Include At Least One of the Following Symbols	Check this option and provide a list of symbols separated by semicolon will force the pattern scanner to return results that have at least one of the symbols in the list.
Must Include All of the Following Symbols	Check this option and provide a list of symbols separated by semicolon will force the pattern scanner to return only results that include all of the symbols in the list.
Must Not Include Any One of the Following Symbols	Check this option and provide a list of symbols separated by semicolon will force the pattern scanner to skip results that include any one of the symbols in the list.

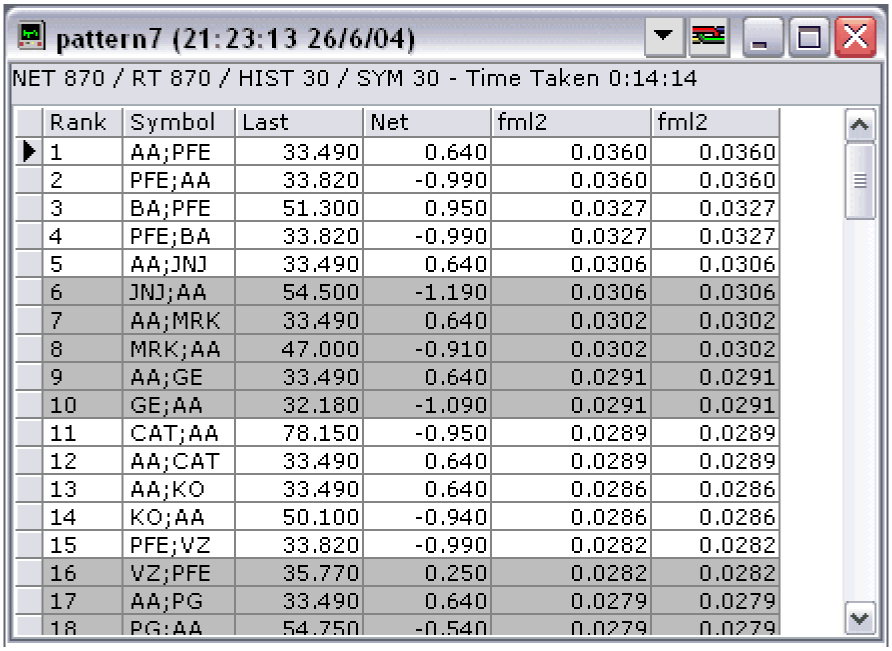
Indicator Setting with Multiple Symbol Scanning

If you provide 2 symbols per iteration, the two symbol will be available as **Data1** and **Data2** under the **Link** setting of the indicator. For example, if you want to use formula to calculate the maximum net percentage spread between two symbols, you will create a formula2 indicator in pattern scanner, with the following formula entered as parameter for **plot1**.

```
$pd1 := prevDClose(0,data1);
$net1 := (data1.close - $pd1) / $pd1;
$pd2 := prevDClose(0, data2);
$net2 := (data2.close - $pd2) / $pd2;
abs($net1 - $net2)
```

Like all indicator scanning, remember to check the settings under the **Data** tab. The example scan works on minute level.

When we apply this example to the stocks in Dow Jones 30 index, pattern scanner will produce a result like:



pattern7 (21:23:13 26/6/04)

NET 870 / RT 870 / HIST 30 / SYM 30 - Time Taken 0:14:14

Rank	Symbol	Last	Net	fml2	fml2
1	AA;PFE	33.490	0.640	0.0360	0.0360
2	PFE;AA	33.820	-0.990	0.0360	0.0360
3	BA;PFE	51.300	0.950	0.0327	0.0327
4	PFE;BA	33.820	-0.990	0.0327	0.0327
5	AA;JNJ	33.490	0.640	0.0306	0.0306
6	JNJ;AA	54.500	-1.190	0.0306	0.0306
7	AA;MRK	33.490	0.640	0.0302	0.0302
8	MRK;AA	47.000	-0.910	0.0302	0.0302
9	AA;GE	33.490	0.640	0.0291	0.0291
10	GE;AA	32.180	-1.090	0.0291	0.0291
11	CAT;AA	78.150	-0.950	0.0289	0.0289
12	AA;CAT	33.490	0.640	0.0289	0.0289
13	AA;KO	33.490	0.640	0.0286	0.0286
14	KO;AA	50.100	-0.940	0.0286	0.0286
15	PFE;VZ	33.820	-0.990	0.0282	0.0282
16	VZ;PFE	35.770	0.250	0.0282	0.0282
17	AA;PG	33.490	0.640	0.0279	0.0279
18	PG;AA	54.750	-0.540	0.0279	0.0279

It shows MCD and MO in combination produces the largest net percentage spread. In a spread trading strategy that hypothesize such spread will reduce in time, you will be shorting one while longing the other.

If you choose to send multiple symbols to the currently chained windows, chart windows with multiple data will have them replaced by the symbols in the listed order.

Using Formula in Pattern Scanner

When using pattern scanner, one of the most complex task is setting up the indicator filters. As individual indicators do return calculated values and can be compared to your designated numeric constants, taking the next step to construct complex condition is quite difficult and hard to master.

To tackle this issue, you can use the built-in indicator formula, formula2 and formula3 to write your conditions directly and avoid the complexity of combining indicators by hand.

Case 1

Lets look at the following scenario.

You are interested to scan the Nasdaq 100 for symbols that is trading above their 5-min 100 period simple moving average. Then you would like to rank the result by today's volume.

You can create a new pattern scanner window. And change the settings as follows,

1. Input symbol list set to Nasdaq 100
2. Use 5-min bar
3. Load 3 days of data
4. Rank by Volume in Descending order
5. Add indicator formula, edit the indicator and type in the following formula in the parameter named **Plot1**,

   ```
   c > average (c, 100)
   ```
6. Then set the filter condition to > 0.

That's all you need to do and you can start scanning right the way.

Notice that if you are setting up the indicators without formula, you will need to use three (3) indicators as follows,

- average, period = 100, link to close
- ref, offset = 0, link to close
- diff, link1 to ref, link2 to average

and then you need to set filter condition for diff to > 0.

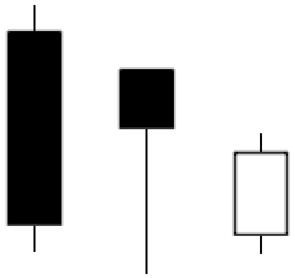
The two style will return exactly the same result, so choosing one will depend on your personal preference.

Case 2

Here is another example on how you can utilize the formula indicator to save time in constructing price pattern scans.

Say you are interested to scan the exact same setup as in Case 1 except the condition is changed to a price pattern called Three River Bottom.

The pattern looks like this.



Basically, to recognize the pattern, you need to verify the following,

- 2 bars ago is a long down bar
- 1 bar ago is a down harami with lower low
- The current bar is a short up bar below the close of 1 bar ago

You can use the following formula to recognize the pattern.

```
Avgrange (2, data1, 5) < (H (2) - L (2)) and    ' part 1
O (2) > C (2) and
L (1) < L (2) and   ' part 2
O (1) > C (1) and
C (1) > (H (1) + L (1)) / 2 and
L > L (1)  and   ' part 3
```

```
   C > O and
   C < C (1)
```

Typing this complete formula into the formula indicator parameter entry could be difficult for some. If you could like to reuse this pattern in other function window then you can create an indicator called ThreeRiverBottom with the exact formula above using the Script Editor. Then you can simply apply this new indicator in the pattern scanner to recognize the pattern.

Indicator Scanning

Pattern scanner can scan with multiple indicators. Indicators can depend on each other just like in charts to allow you to set up complex scanning based on indicators.

Enabling Indicator Scanning

To setup multiple indicator scanning:

1 Open the pattern scanner setup window

2 Press the **Indicator** tab in the pattern setup window.

3 In order to to scan with indicators, you need to check the **Enable Indicators** option..

4 Press the **Data** tab.

5 Verify trading time and holiday list information.

6 Verify bar information, e.g. 5-min if you want to scan using 5-minute bars.

7 Verify **Days to Load** information to ensure you have enough data for indicator calculation.

Settings that Affect Indicator Calculation

For indicator scanning, you need to tell pattern scanner information such as bar size, days of data to load, trading time, etc.

To set up:

1 Open the setup window.

2 Press the **Data** tab.

3 Bar size settings are under **Format**.

4 Trading time and holiday list information settings are under **Data Setting**.

5 Apply the changes.

You can use **Quick Fill** to fill in **Data Setting**.

Adding, Editing, Copying Indicator

To add or edit an indicator:

1 Press the **Indicator** tab in Pattern Scanner Setup.

2 Press the **Add** button to add indicator. You can edit an existing indicator by pressing the **Edit** button. Adding an indicator to the pattern scanner is almost identical to adding an indicator to charts. The difference is in the pattern scanner, the only data series available is the symbol that is currently being scanned.

Indicator can depend on another indicator. For example, you can scan with a moving average applied to a MACD indicator.

To copy an indicator:

1 Press the **Indicator** tab in Pattern Scanner Setup.

2 Click on the indicator you want to copy to select it.

3 Press the **Copy** button. A copy of the indicator will be added to the list of indicators.

Ranking

When you have multiple indicators, you can specify one of them as the ranking indicator by checking the **Rank** field in **Indicator** tab in pattern scanner setup. The ranking indicator is used to rank the scanning result when rank by indicator is used. Rank by indicator is set under the **Options** tab.

Turning Indicators On and Off

In pattern scanner setup, under **Indicator** tab, the **On** field lets you turn on/off individual indicators during scanning. This allows you experiment on different scanning techniques.

IMPORTANT: if one indicator depends on another indicator, and you turn off the indicator that is being depended on, then indicator scanning cannot perform properly on the dependent indicator.

Indicator Filtering

You can enable indicator filtering. By turning on a filter, you can filter out items that do not have indicator values that match the filter.

To enable indicator filtering:

1 Open pattern scanner setup window.

2 Press the **Indicator** tab.

3 Each indicator can use up to two conditions: **C1** and **C2**. A symbol will be filter out if the indicator value does not match the conditions specified.

Loading Indicator Setup from a Time Chart

You can load the indicator setup from a time chart. Loading indicator setup from a chart is very useful when you want to use a complex indicator setup to scan. You can construct the indicator setup in a time chart, experimenting on different symbols and parameter settings before you perform the actual scanning.

When you load a setup from a chart, information such as indicator dependency and parameters are loaded. The data series of the chart are replaced by the default data series of the indicator.

Loading indicator setup form a time chart does not change the time frame information under the **Data** tab. You should verify that the **Data** tab contains the proper settings for the scanning you intend to perform.

To load a setup from a chart:

1 In a time chart, construct the indicator setup.

2 Save the time chart.

3 In pattern scanner setup window, press the **Indicator** tab.

4 Press the **Load From Chart** button and choose the time chart.

5 Verify the information under **Data** tab.

6 Apply the settings.

Multiple Symbols Per Iteration Scanning Concerns

Normally pattern scanner perform single symbol per iteration scan. If you are performing multiple symbols per iteration scan (such as spread scan), you can specify multiple data series in the time chart, one for each symbol. Load from chart will automatically map the data series to the multiple symbols.

Improving Indicator Scanning Performance

Internal indicator and formula indicator evaluates faster than script-based indicators. Therefore, if you have a particular indicator scanning setup in mind, try constructing the setup with multiple internal indicators or formula before writing an indicator script for scanning.

To tell whether an indicator is internal, formula, or script-based, open the Indicator Manager by choosing **Program>Indicator Manager** from the main window.

Many indicators do not require a complete calculation of the data. You can control the number of bars to be calculated in **Calculation Preference** under the **Indicator** tab in pattern scanner setup.

For example, for a 10-period moving average, it is sufficient most of the time to calculation 15 to 20 bars (The extra bars provide a safety net for invalid bars).

Displaying Indicator Values

You can choose to display the indicator values in the scanning result.

1 Open Pattern Scanner setup.

2 Press the **Indicator** tab.

3 Select the indicator you want the value to display in scanning result by checking the **Out** check box.

Scanning with Internet Data Feed

If you use an Internet data feed, there are some important considerations you should take into when you use pattern scanner. This section is not applicable to users of IDS and off-line mode.

Limitations

There are several limitations you must aware when you scan with Internet data feed:

1 Internet data feed provider does not broadcast full market symbols. You must directly or indirectly provide a symbol list to scan from.

2 When NeoTicker® requests real-time data for a symbol, there is a delay between making the request and the data becomes available. The delay makes real-time scanning with large number of symbol (several thousands plus) impractical for Internet feed users. You should consider using another pattern scanner that relies on off-line data (disk cache, EOD file, etc) to filter out most symbols before feeding the symbols to a pattern scanner that works directly off a real-time server.

3 The delay is not uniform for all fields. Time sensitive fields such as last price are usually available before less time sensitive fields such as previous day close.

All Current Symbols

One way to use the pattern scanner is to scan from all current symbols. In the context of Internet data feed, all current symbols are the symbols that NeoTicker® has made request to. This is the list of symbols that are currently in the quote window, time and sales windows, charts, etc.

You can pre-load a symbol list to the current symbols in the server setup. The main advantage is instant access because the pre-loading is done during morning restart.

To learn how to pre-load a symbol list, refer to ***Tutorial: Turbo Charging Data Loading Speed*** (on page 205).

Non-Uniform Data Field Delays

Some data fields receive data much later than others. This is especially true when you retrieve large number of symbols. When you set up the scanner, you should avoid using slow fields to filter.

In general fields that change rapidly in real-time are updated faster.

Data Feed with Very Low Symbol Limit

For speed, pattern scanner will be connecting to a certain number of symbols, and release the symbols as the scanning progress.

If your data subscription has a very low symbol limit, you may want to set up pattern scanner to release symbol more often so it does not break your symbol limit.

1. In pattern scanner setup, press **Schedule / Misc** tab.
2. Settings are under **Data Vendor Symbol Limit Special Handling**. Typically you will want to set the number to well below your symbol limit because you want to leave some symbols for charting, etc.

Scanning without Real Time Server

Pattern scanner allows you to scan with with a data source other than your real-time server. Examples include EOD data from Quotes Plus, TC2005, Internet, Disk Cache and any file-based format. Some of the data source offers unique advantage such as long EOD historical data and very fast access compared.

EOD scanning is very much like real-time scanning with some limitations.

Scanning with Quotes Plus or TC2005 requires you to be a subscriber. Quotes Plus and TC2005 are very suitable for scanning as its data access is extremely fast.

Setup

1 In the pattern scanner's **Data** tab, choose anything but **RT Server**.

2 You should verify the **Days to Load** and **Time Frame**. For EOD formats, you want to make sure the number of days are sufficient for end-of-day scanning and you use at least a daily time frame for scanning.

3 When you scan without a real-time server, the basic filters are not available as they rely on the real-time server. You can scan with indicators or patterns. You can simulate many of the real-time filters behavior by filtering indicators such as absolute value or plot value.

4 You must specify a symbol list.

Characteristics of Different Sources

Source	Characteristics
Quotes Plus	Very fast. Has commodities data. EOD only.
TC2005	Very fast. EOD only.
Internet EOD Data	EOD only.
Disk Cache	Intraday and EOD. Very fast. It uses your current disk cache as input. So you need data cache.
Metastock	EOD only. File-based.
CSI, CSIM	EOD only. File-based.
Text(CSV, comma and space separated)	EOD and intraday. File-based.

Scanning with Visual Patterns

In pattern scanner setup, the **Pattern** tab lets you specify a visual pattern file to scan with. To create a visual pattern file, zoom into a region in a time chart and choose **Save Pattern** from the time chart's pop up menu. The price action from the first data series is then saved in a visual pattern file.

For reliable visual pattern scanning, it is advised you use a pattern is at least 20 bars long.

Pattern

Specifies a file that contains a visual pattern for scanning

Pattern Preview

You can click in this area to load the visual pattern in the area for you to preview

Option

The type of pattern matching algorithm to use. Currently, the **Technical** model is available. You can provide a cut off level using the Confidence %.

Field Selection

By default, visual pattern uses all data fields for pattern matching. You can limit the data fields to the selected fields.

Lookup Tab

Settings under this tab are for backward compatibility only and are no longer supported. If you are setting up a new scan, use indicator scanning.

Filters under the **Lookup** tab compare real-time data with end of day historical information to perform filtering. These filters depend on the EOD lookup table. To use these filters, you must have the EOD lookup table enabled under the Server tab in the main window.

Filters in the **Lookup** tab depend on real-time data and are not available for end of day scanning.

Protecting Your Work

Function Window Write Protection

You can write protect a function window (charts, quote windows, etc) to guard against accidental changes:

1 In the function window, right click to open pop up menu.

2 Enable **Write Protected**.

3 The function window will prompt you to save before applying write protection. You must save the window before write protection can be applied.

4 To disable write protection, use the pop up menu and choose **Write Protected** again.

Once a window is write protected, you can still make changes to the window. For example, you can still add indicators to a write protected chart. However, NeoTicker® will refuse to save the changes. Therefore, you can simply close the window and re-open it to undo the changes.

Locking Indicators, Windows and Templates

If other people can access your computer, you may want to consider protecting your indicator, window and template from other users by locking these files. Once locked, indicator, window and template files are in an encrypted form only NeoTicker® understands. These files can still be used in NeoTicker® with no restriction, but other people cannot understand the logic.

Here, locked work is intended to be used by you only so the workflow described is simplified. If you want to lock your work for deploying to a customer of yours, you should read the chapter ***Third Party Developer Guide*** (see "Guide To Protection" on page 1659), which provides a complete view of the matter.

Creating a Product

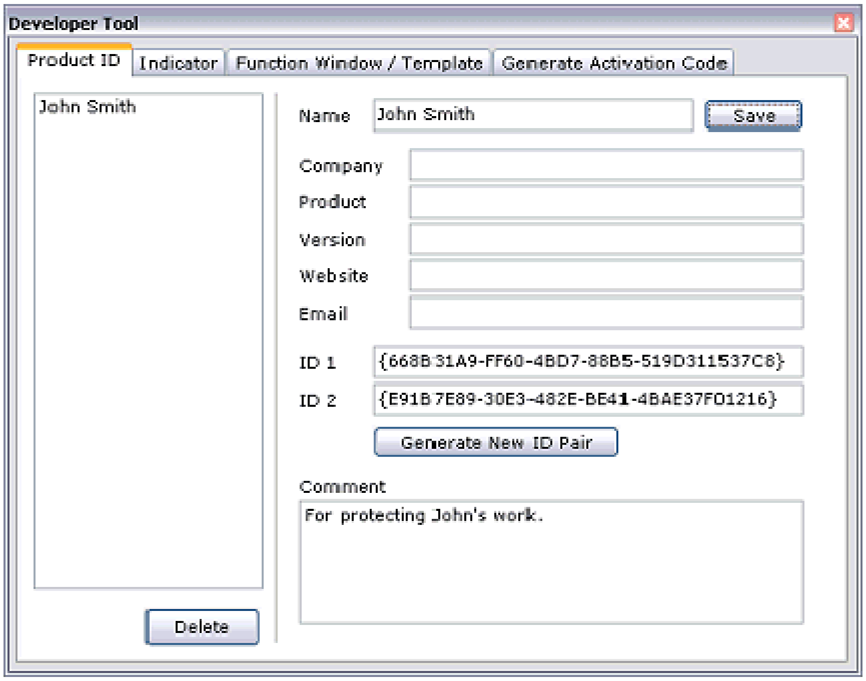
You need to create a product to lock your work. As an individual user, the product is used to uniquely identify the works created by you. Unlike commercial developers, you only need to create to a single product for all your works.

1. In main window, choose **Tool>Developer Tool** to open Developer Tool.
2. In Developer Tool, press the **Product ID** tab.
3. For **Name**, enter your name.
4. Leave **Company**, **Product**, **Version**, **Website**, **Email** blank.
5. For **Comment**, you can enter some descriptive text (optional).
6. Press **Generate New ID Pair** button, two ID strings are generated in **ID1** and **ID2**.
7. Press **Save** button.
8. Whenever you need to lock your work, you will need to choose the product you have just saved.

ID strings are critical to encryption/authorization. Therefore, you should make sure:

- You keep back ups of ID strings (print it out if necessary).
- Only you should have access to ID strings.

The figure below shows a product created by John Smith.



John Smith

Name: John Smith

Company:

Product:

Version:

Website:

Email:

ID 1: {668B31A9-FF60-4BD7-88B5-519D311537C8}

ID 2: {E91B7E89-30E3-482E-BE41-4BAE37F01216}

Comment: For protecting John's work.

Locking Indicators

To create the locked version of an indicator.

1. Open Developer Tool.
2. Press **Indicator** tab.
3. Under **File**, choose the indicator file you want to lock.
4. Under **Prod Id**, choose the Product ID you've created.
5. Make sure **Require Activation Code** is disabled.
6. Under **Password**, enter a password of your choice. Type the password again under **Confirm Pwd**. Password is required if you want to view/edit a locked indicator.
7. Press the **Create Locked Version** button.

Tip:

- Password is not saved with your Product ID or NeoTicker® session for security reason. You should memorize it, or write it down and keep it in a safe place.

Locking Windows and Templates

To protect a window/template, a locked version of the window/template is created. The locked version is an encrypted version of the original window/template. The locked version has the following properties:

- The locked file is not human readable, so it is a form of Source Code Protection.
- Once the file is opened by NeoTicker® as a window/template, you cannot read any formula in the window/template. For example, if a chart contains an fml indicator, the formula in the fml indicator is not visible.
- If the locked file is saved again, it will be saved in the same protected form as the original.

To create the locked version of an window/template:

1. Open Developer Tool.
2. Press **Function Window / Template** tab.
3. Under **Type**, choose the type of window/template you want to lock, e.g. chart, quote, etc.
4. If you are locking a function window, choose **Window**. If you are locking a template, choose **Template**.
5. Under **File**, choose the window/template file you want to lock.
6. Under **Prod Id**, choose the Product ID you've created.
7. Make sure **Require Activation Code** is disabled.
8. Under **Password**, enter a password of your choice. Type the password again under **Confirm Pwd**. Password is required if you want to view/edit a locked window/template.
9. Press the **Create Locked Version** button.

Tips:

- Window/template locking is a weak form of protection. If someone opens the window/template, he/she can simply duplicate the settings manually to reverse engineer your work. Therefore, you must create settings that you deem valuable using formula or a locked indicator, both of which are not visible from anyone other than yourself.
- Password is not saved with your Product ID or NeoTicker® session for security reason. You should memorize it, or write it down and keep it in a safe place.
- If your chart contains an indicator you write, and the indicator takes formula as a parameter. When you design the indicator, make sure the parameter is designated as a formula, not a string. This ensures NeoTicker® blocks view of the formula.

Unlocking Your Work

To unlock an indicator:

1. Open Developer Tool.
2. Press **Indicator** tab.
3. Under **Unlock Indicator**, enter **File** name of the locked file, **Prod ID** and your **Password**.
4. Press **Recover Original Version** button.

To unlock window/template:

1. Open Developer Tool.
2. Press **Function Window / Template** tab.
3. Under **Unlock Window / Template**, enter **Type**, choose **Window** or **Template**, enter **File** name of the locked file, **Prod ID** and your **Password**.
4. Press **Recover Original Version** button.

Quote Memo Operation Guide

Introduction

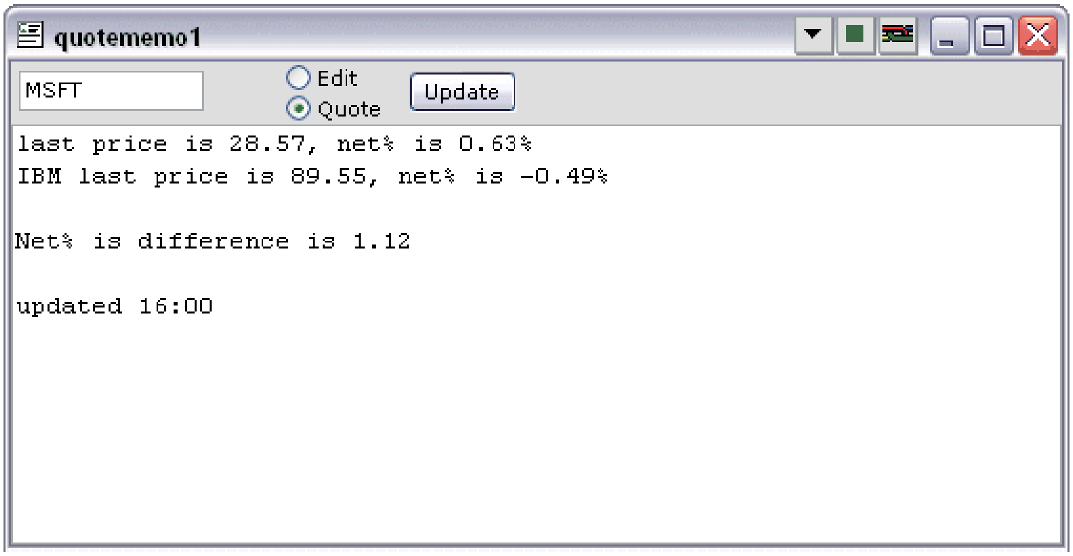
Quote memo lets you type free form memo in a function window.

Because quote memos are function window, they are part of window grouping. This makes quote memo very handy for taking notes for different groups.

Quote memo also allows you to evaluates quote fields, formulas, etc. You can use quote memo as a free form quote window.

To create a quote memo, press the quote memo tool button .

The following figure is an example of a quote memo:



```
last price is 28.57, net% is 0.63%
IBM last price is 89.55, net% is -0.49%

Net% is difference is 1.12

updated 16:00
```

Edit vs. Quote

Quote memo works in two modes: Edit and Quote. You can switch between modes either in the panel or use pop up menu.

In Edit mode:

- You can freely type anything
- Do not pull intraday quotes
- Do not evaluate formula

In Quote mode:

- Memo is read only
- Intraday quotes are pulled
- Formulas are evaluated

Therefore, if you intend to use quote memo as a simple memo, you can use Edit mode. Quote mode can act as write protection.

If you intend to use quote memo to evaluate quotes and formulas, you should use Edit mode to enter quotes and formulas, then you Quote mode to evaluate.

Hiding/Showing Top Panel

Top panel in quote memo can be hidden. To hide/show top panel, choose **Top Panel Visible** from pop up menu.

Background Color and Font

You can adjust quote memo background color and font from pop up menu. Choose **Background Color** or **Font**.

Foreground color is the font color, you can adjust foreground color by changing font color.

Saving to File

Text in quote memo can be saved directly to a text file by choosing from the pop up menu, **Save to Text File**.

Save to text file will save whatever text that is current in the memo. Thus you can save the tags in Edit mode, and the quote result in Quote mode.

Quotes and Formulas in Quote Memo

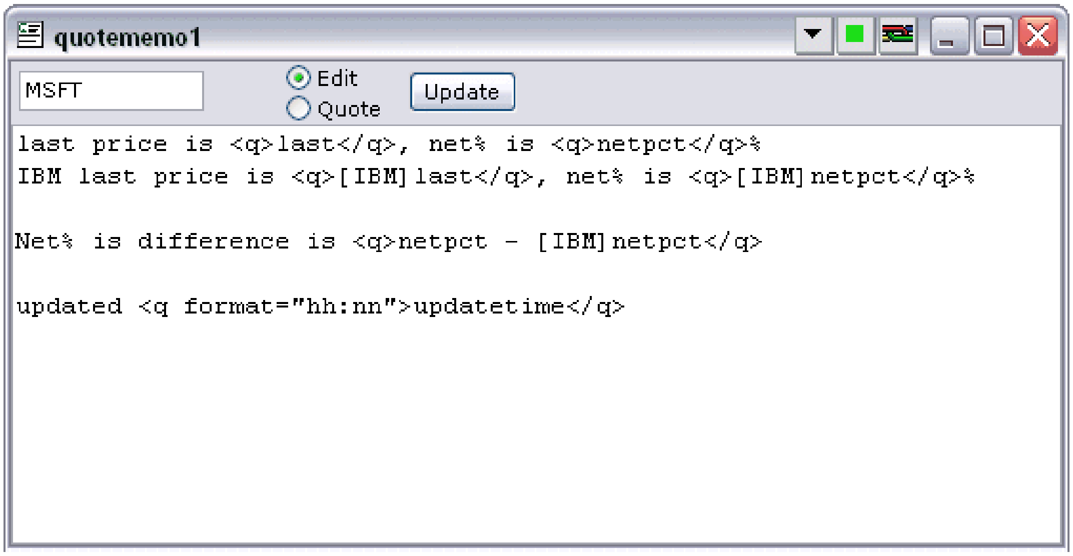
Quote memo shares the same quote fields as quote window, e.g. you can get the last price, net, update time etc. In addition, you can evaluate formulas. Quotes and formulas can be placed anywhere in the quote memo, provided a single quote/formula resides in a single line.

The symbol field in the top panel specifies the primary symbol of quote memo.

To enter a quote or formula:

1. Change to Edit mode
2. Type the quote field, surrounded by the tags: `<q>` and `</q>`
3. Change to Quote mode

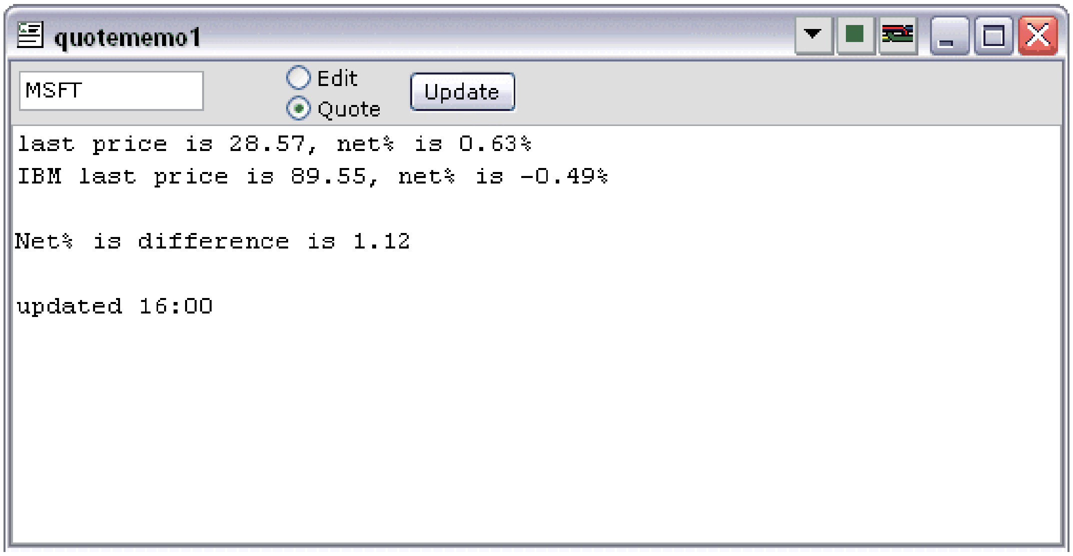
The following figure is an example of quote memo with quotes and formulas, in Edit mode:



MSFT

```
last price is <q>last</q>, net% is <q>netpct</q>%
IBM last price is <q>[IBM]last</q>, net% is <q>[IBM]netpct</q>%

Net% is difference is <q>netpct - [IBM]netpct</q>

updated <q format="hh:nn">updatetime</q>
```

The following figure is the same quote memo, in Quote mode:



```
last price is 28.57, net% is 0.63%
IBM last price is 89.55, net% is -0.49%

Net% is difference is 1.12

updated 16:00
```

The following quotes and formulas are valid in the example above:

Quote/Formula	Meaning
`<q>last</q>`	Last price of primary symbol
`<q>netpct</q>`	Net % of the primary symbol
`<q>[IBM]last</q>`	Last price of IBM
`<q>[IBM]netpct</q>`	Net % of IBM
`<q>netpct - [IBM]netpct</q>`	Difference between primary symbol's Net % and IBM's Net %
`<q format="hh:nn">updatetime</q>`	Quote update time, with optional formatting parameter

Formatting

In the <q> tag, you can specify an optional parameter to format the value. The syntax is:

<q format=*format_string*>

Format_string is a quoted string, specifying the formatting rule.

To format a value, use a format string similar to "0.000". The following table illustrates some examples:

Format String	Meaning
`"0.00"`	2 decimal places
`"0.000"`	3 decimal places
`"0"`	No decimal places

To format time, use a format string that is compatible with the default time format string. The following table illustrates some examples:

Format String	Meaning
`"hh:nn:ss"`	Hour minute second
`"MMM dd"`	Month and day, e.g. May 15
`"yyyy/mm/dd"`	Year, month, day, e.g. 2003/05/15

For the complete specification of time format, refer to the section ***Setting a Default Date Time Format*** (see "Default Date Time Format" on page 503).

Special Tag to Display Primary Symbol

You can use the special tag `<s>` to display the primary symbol. This tag is particularity useful when you hide the top panel but still want to see the primary symbol displayed.

Indicator Calculation

EOD version does not support indicator calculation in quote memo.

Formulas in quote memo can evaluate indicator as well. Indicator wizard is provided in the pop up menu to help you specify indicators.

The indicator format is the same as quote window. The following is an example of an moving average calculation:

```
    <q>average(0,M1,10)</q>
```

Note that for different markets, e.g. commodities vs. stocks, you may need to specify time frame for indicator calculation.

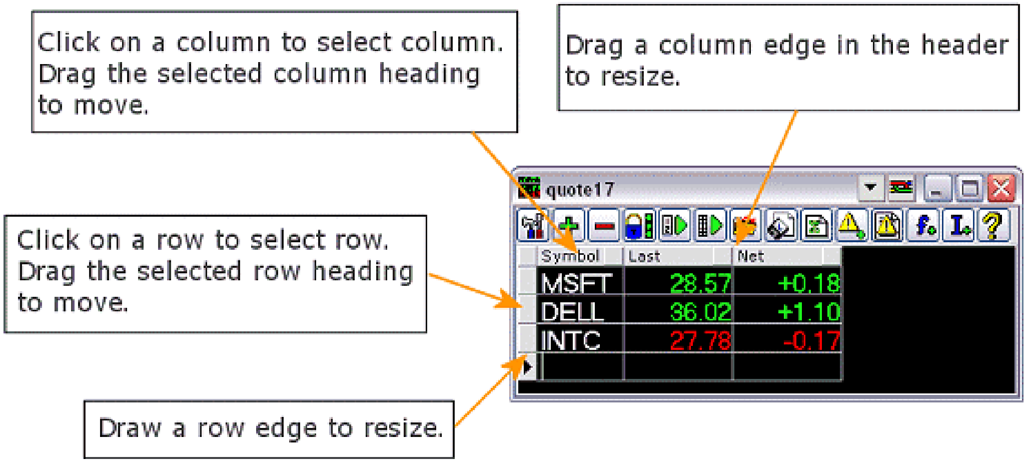
Quote Window Operation Guide

Quote Window Setup

To setup quote window, open the quote window setup window by either:

- Choose **Setup** in the pop up menu, or
- Press the Setup button .

Re-arranging and Resizing Rows and Columns



Click on a column to select column. Drag the selected column heading to move.

Drag a column edge in the header to resize.

Click on a row to select row. Drag the selected row heading to move.

Draw a row edge to resize.

Symbol	Last	Net
MSFT	28.57	+0.18
DELL	36.02	+1.10
INTC	27.78	-0.17

You can customize the look of the quote window by dragging the edges and cells.

To resize a column, point the mouse to the column edge in the column heading and drag.

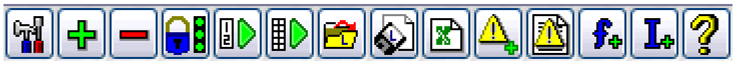
To resize all rows, point to the row edge in the row heading and drag.

To reorder a column, click on the column heading to select it, release the mouse button, click on the column heading again and drag to the desired position.

To reorder a row, click on the row heading to select it, release the mouse button, click on the row heading again and drag to the desired position.

Quote Window Tool Buttons

Quote windows have a set of tool buttons to help you access the commonly used functions in quote window.



The tool buttons are supplement to the quote window pop up menu. Not all the functions in the pop up menu can be found in the tool button. In fact, you can hide and show the tool buttons by choosing **Show Tool Buttons** and **Hide Tool Buttons** from the pop up menu.

For users who do not want the tool buttons displayed, they can set this up in the user preference.

1 Choose **Program>User Preference** in the main window to open the user preference.

2 Disable the **Show tool buttons for new window option** in the **Quote Window** group. Once the option is disabled, new quote window will no longer has the tool window on as a default.

What the Buttons Do

Open the Quote window setup window

Add a Row

Delete a Row

Lock sorting in the quote window

Start/Stop Timer Sort (auto ranking)

Start/Stop Row Filtering

Imports the symbols from a symbol list

Exports the symbols as a symbol list to a file

Export to Excel

Add a Quote window alert

Shows the list of alerts associated with this Qutoe window

Adds a formula column and opens the column properties window for editing. Not available in NeoTicker® EOD.

Adds an indicator column. An indicator column is simply a formula column. The Indicator Wizard will be opened to help you quickly specify an indicator into the indicator column. Not available in NeoTicker® EOD.

Open help.

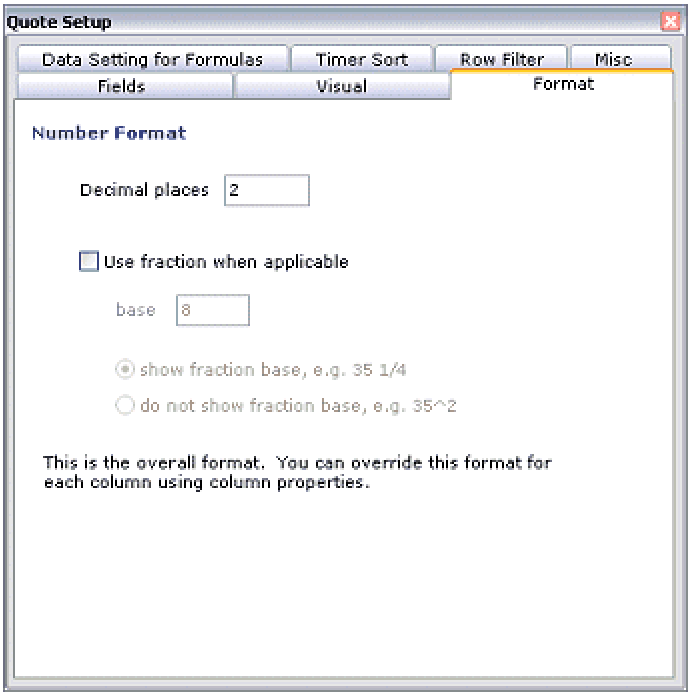
Formatting Values

You can format the display of quotes. There is a common setup for the formatting rules for the whole quote window. In addition, you can specify formatting rules for a particular column.

To set up the formatting rules for the whole quote window:

1 Open the setup window by choosing **Setup** from the pop up window

2 Press the **Format** tab.



Number Format

Decimal places 2

□ Use fraction when applicable

base 8

● show fraction base, e.g. 35 1/4

○ do not show fraction base, e.g. 35^2

This is the overall format. You can override this format for each column using column properties.

For values that have a decimal display, you can specify the number of decimal places. Some values can be displayed in fractions (useful for commodities and bonds). If you check the **Use fraction when applicable**, these values will be displayed as fractions. You can specify the base for the fraction.

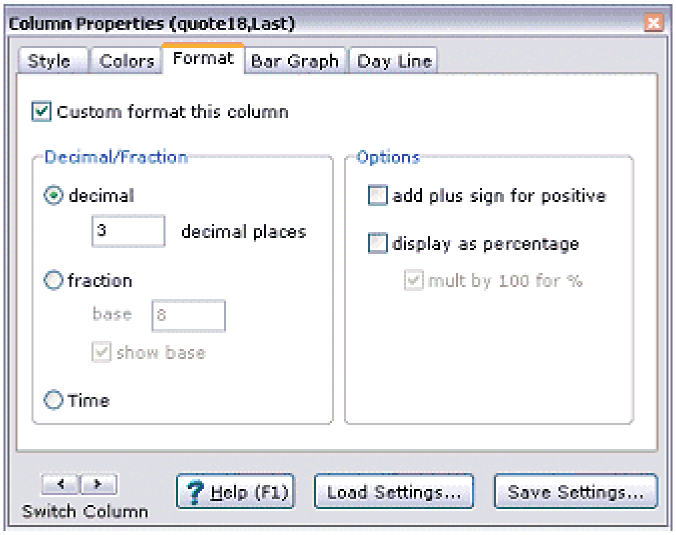
To set up the format rules for a specific column, use quote column properties:

1 Click on the column you want to format, and click the right mouse button to open the pop up menu.

2 Choose **Setup Column Properties**

3 Press the **Format** tab.

4 Check the **Custom format this column** check box. This tells the quote window this column has its own set of formatting rules. Then you can set the formatting rules for the column.



☑ Custom format this column

Decimal/Fraction

● decimal

3 decimal places

○ fraction

base 8

☑ show base

○ Time

Options

□ add plus sign for positive

□ display as percentage

☑ mult by 100 for %

Quote Window Level Formatting Options

Option	Meaning
Decimal places	Number of decimal places. Note: Some columns such as volume by default does not display decimal places at all
Use fraction when applicable	Columns that are displayed as decimal can be displayed as fractions (e.g. prices, net change). If you check this box, these values are displayed as fractions, useful for futures and commodities.
base	Base for fractions.
show fraction base	If on, 35.25 is displayed as 35 1/4
do not show fraction base	If on, 35.25 is displayed as 35^2, assuming that the base is 8.

Column Level Formatting Options

The **Format** tab in column properties lets you set up the formatting rules specific to the column. If you do not specify formatting rules for the column, the values in the column are formatted according to the formatting rules of the quote window.

The formatting rules specified in column properties are "stronger" than the quote window formatting rules. If you specified column properties formatting rules, it will take precedence over the quote window formatting rules. Also, quote window formatting rules try to take into consideration the type of data to be displayed before applying formatting rules. For example, under the quote window formatting rules, decimal places are never added to volume data, while under the column properties formatting rules, decimal places can be added to volume data if you choose to.

Option	Meaning
Custom format this column	Specify formatting rules for this column. If this option is not set, the column follows the general formatting rules of the quote window.
decimal, **decimal places**	Format this column as decimals with the specified decimal places
fraction, **base**, **show base**	Format this column as fraction with the specified base. If show base is set, the base is shown. For example, 35.25 is displayed as 35 1/4; otherwise 35.25 is displayed as 35^2, assuming that the base is 8.
add plus sign for positive	Add a "+" sign to positive numbers

display as percentage	Add a percentage sign to the formatted values
mult by 100 for %	If displayed as percentage, multiple the values by 100. For example, 0.02 will be displayed as 2%.

Rule-based Coloring

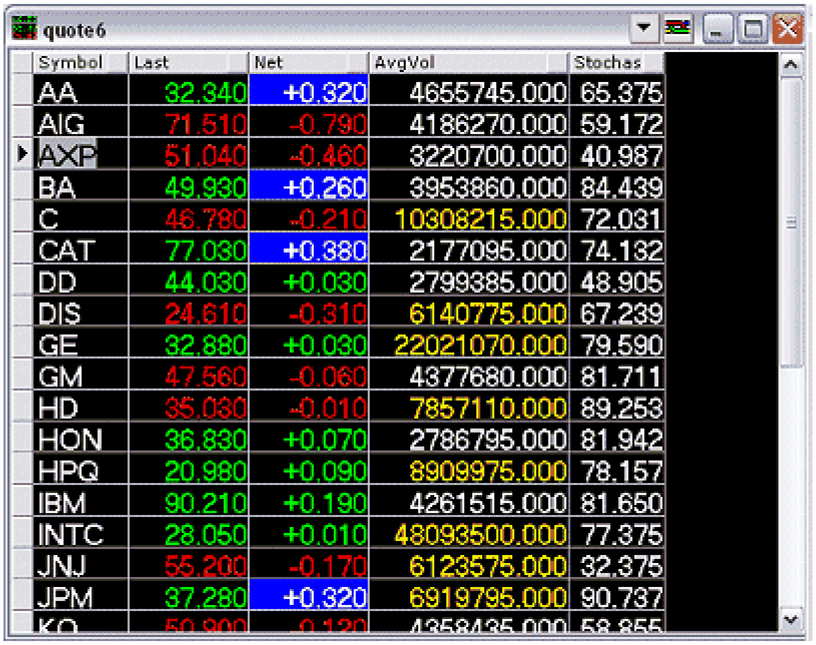
You can set the coloring rules for a column such that each cell in the column is colored according to the rules.

To setup, open the column properties for the column and press the **Color** tab.

For example, see ***Setting up Rule-based Coloring for a Column*** (on page 129).

Quote Window Formula

One of the most powerful feature in quote window is Quote Window Formula. When using Quote Window Formula in a column, the formula is applied to each symbol in the quote window. So you can perform indicator calculations and advance analytics in quote window.



Symbol	Last	Net	AvgVol	Stochas
AA	32.340	+0.320	4655745.000	65.375
AIG	71.510	-0.790	4186270.000	59.172
AXP	51.040	-0.460	3220700.000	40.987
BA	49.930	+0.260	3953860.000	84.439
C	46.780	-0.210	10308215.000	72.031
CAT	77.030	+0.380	2177095.000	74.132
DD	44.030	+0.030	2799385.000	48.905
DIS	24.610	-0.310	6140775.000	67.239
GE	32.880	+0.030	22021070.000	79.590
GM	47.560	-0.060	4377680.000	81.711
HD	35.030	-0.010	7857110.000	89.253
HON	36.830	+0.070	2786795.000	81.942
HPQ	20.980	+0.090	8909975.000	78.157
IBM	90.210	+0.190	4261515.000	81.650
INTC	28.050	+0.010	48093500.000	77.375
JNJ	55.200	-0.170	6123575.000	32.375
JPM	37.280	+0.320	6919795.000	90.737
KO	50.900	-0.120	4358435.000	58.855

There is series of tutorials to get you started using quote window formula. See:

Tutorial: Quote Window Formula 1 (see "Tutorial: Quote Window Formula 2" on page 145)

Tutorial: Quote Window Formula 2 (on page 145)

For more advance topics, see:

Formula in Quote Windows (on page 331).

Formulas can include indicators, which are affected by time frame settings and number of bars loaded using historical data. See ***Specifying the Data Settings for Indicator Calculation*** (on page 859) and ***Changing the Number of Bars Available for Indicator Calculations*** (on page 859).

Scanning with Quote Window

Quote Window is a great tool for scanning. By using features such as Quote Window Formula and Auto Ranking, quote window is capable of handling scan jobs that involves several hundred symbols.

Follow the tutorial ***Tutorial: Creating a Great Quote Window*** (on page 125) to see how this can be done.

For large scanning jobs, use Pattern Scanner instead.

Re-using Column Settings in Another Column

The column properties (choose **Setup Column Properties** from the pop up menu) allow you to save and load settings.

1 Double click on the column you want to save settings. This opens Column Properties dialog.

2 Press the **Save Settings** button in Column Properties. Give it a file name and save.

3 Open Column Properties dialog for quote window you want to re-use the settings. Press the **Load Settings** button. Load the setting file.

This is useful when you want to reuse the coloring, formatting rules and formulas in another column. Simply save the settings of the column you want to reuse and load the settings in the column you want to apply the settings to.

This feature is especially useful for coloring and formatting rules because these settings can be tedious to set up.

Specifying the Data Settings for Indicator Calculation

Users who are familiar with charts know that indicators can behave differently with different data settings. By data settings, we are referring to the trading days, trading time, holidays information. In charts, the data settings are set in the chart manager, under the **Data Settings for Formulas** tab.

Indicators are also being calculated in quote windows when you have indicators in formula. To change data settings for a quote window:

1. Choose **Setup** from the pop up menu to open the quote window setup.
2. Press the **Data Setting for Formulas.**
3. You can choose a pre-defined time frame (**Predefined Time Frame**), or manually specify the settings (**Customized**).

Changing the Number of Bars Available for Indicator Calculations

Quote window uses RAM Cache for indicator calculation. By default, RAM Cache stores 250 bars of indicator bars for back referencing. The number of bars will increase within the NeoTicker® session as trading data comes in. Therefore, 250 bars are sufficient for many usages. However, you may to want guarantee a minimum number of bars available. For example, if you want to use "close(280, M1)" in your formula and want the result of the calculation to show up as soon as possible. You want to change the bar per series minimum from 250 to 280.

To modify this value:

1. Open the cache manager by choosing **Manager>Cache** from the main window.
2. In the cache manager, press the **RAM Cache** tab and press the **Option** button.
3. In the RAM Cache Option window, press the **Bars Per Series** tab.
4. Change the values under the **Minimum** to increase or decrease the bars available as a minimum.

Tracking Portfolio

This section is for positions you enter manually into quote window. If you want to track position you have in your brokerage account or Trade Simulator, see ***Position Information as Quote Fields and Formula Functions*** (on page 782).

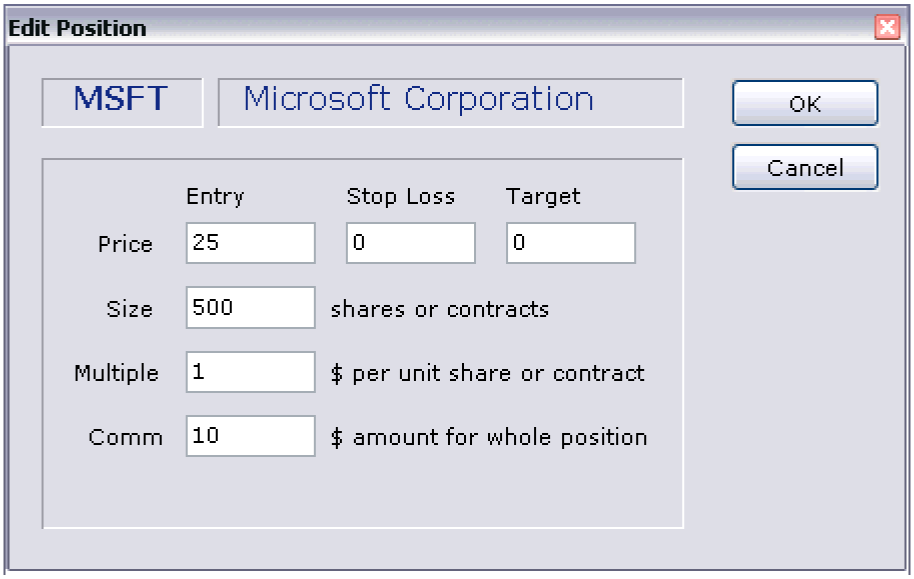
You can use the quote window to help you track your portfolio. This section is an example of setting up portfolio tracking.

The following fields are available for displaying position information in quote window. They can be added just like any other quote window fields.

PosComm	Commission cost for the position
PosCost	Cost of position. This equals PosPrice * PosSize * Multiple + PosComm
PosFromStopLoss	Current price from stop loss price
PosFromTarget	Current price from target price
PosPL	The profit/loss of the stock. This equals PosValue - PosCost
PosPL%	The profit/loss of the stock. This equals PosPL / PosCost
PosPrice	Entry price for the position
PosSize	Size of position
PosStopLoss	Stop loss price for the position
PosTarget	Target price for the position
PosValue	Value of the stock. This is the last price of the stock times PosSize

To enter/edit a position, right click on the symbol you want to edit to open the pop up menu and choose **Edit Position**.

The Edit Position window will be opened.

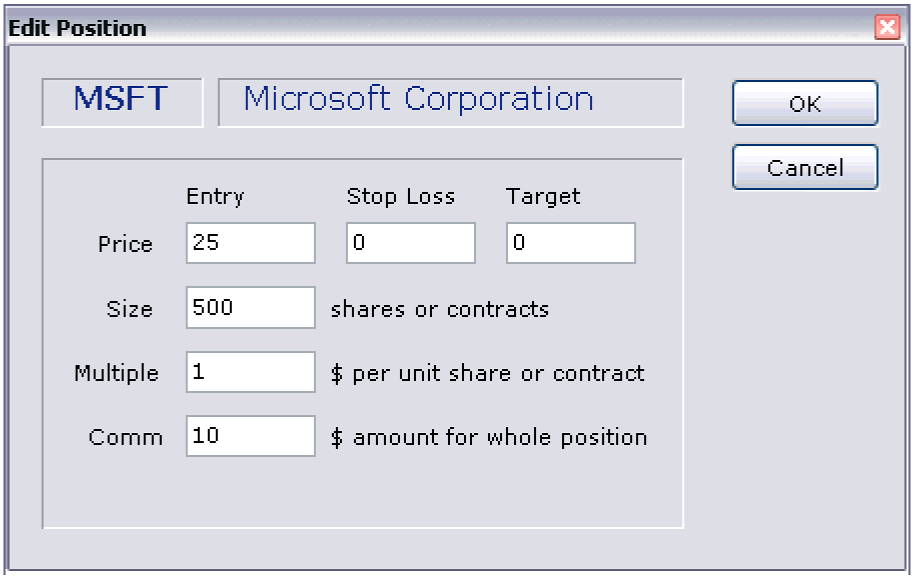


MSFT Microsoft Corporation

	Entry	Stop Loss	Target
Price	25	0	0
Size	500	shares or contracts	
Multiple	1	$ per unit share or contract	
Comm	10	$ amount for whole position	

Note: The **Multple** field is a multiplier for options and futures contracts where each position represent multiple points. Leave **Multiple** as 1 if the symbol is a stock.

Edit Position Window Options



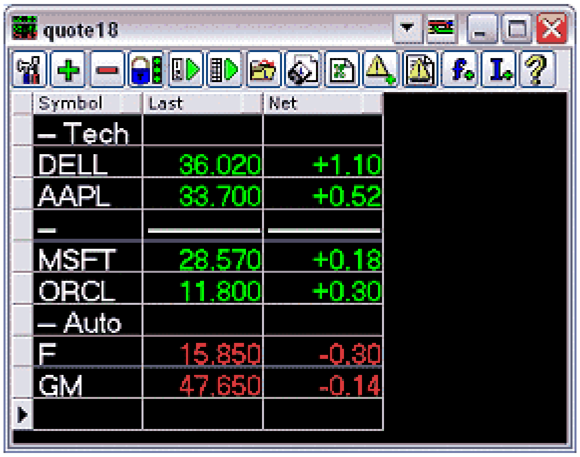
MSFT Microsoft Corporation

	Entry	Stop Loss	Target
Price	25	0	0
Size	500	shares or contracts	
Multiple	1	$ per unit share or contract	
Comm	10	$ amount for whole position	

Setting	Meaning
Entry Price	Entry cost of the position (PosPrice)
Stop Loss Price	Stop loss price you have in mind (PosStopLoss)
Target Price	Target price you have in mind (PosTarget)
Size	Number of units (shares/contracts) you purchased (PosSize)
Multiple	Multiplier to be used for options and futures. For stocks, **Multiple** should always be 1
Comm	Commission cost for the position (PosComm)

Vertical Sections

You can enter special symbol into the quote window for formatting purpose.



Symbol	Last	Net
– Tech		
DELL	36.020	+1.10
AAPL	33.700	+0.52
–		
MSFT	28.570	+0.18
ORCL	11.800	+0.30
– Auto		
F	15.850	-0.30
GM	47.650	-0.14

Vertical Separator

You can enter two minus sign (--) as a symbol. Two minus sign as a symbol tells quote window to treat the row as a separator line.

Section Heading

If you enter some words after the two minus sign (e.g. `-- HI TECH`), quote window will treat the row as a section header and display it as it is.

Sorting

When you sort a column, the sorting will happen within a section, i.e. sorting will not cause symbols in one section to mix with symbols in another section.

You can turn off this behavior by:

1 Open quote window set up.

2 Under **Misc** tab, choose **Ignore Section**.

Special Symbols

If you enter the following symbols into quote window, these symbols will perform computation rather than getting quotes from data service.

Special Symbol	Meaning
`--`	Section separator
`=SUM`	Sums up column values in a section

Timer Sort (Auto-Ranking Rows)

You can automatically rank the rows by a column's value. For example, you can auto arrange on Net % to monitor which symbol has the highest/lowest net percentage change. The ranking is automatically changed as net percentage changes.

To auto-arrange:

1 Enable the timer sort feature by either pressing the timer sort button or choose **Start Timer Sort** in the pop up menu.

2 Press the sort button on the column heading to rank. Press the button again to revert ranking direction.

You cannot modify the quote window when timer sort is on. You must turn off timer sort before modifying.

Adjusting Sorting Interval

The ranking happens every 10 second by default. You can adjust the this interval by:

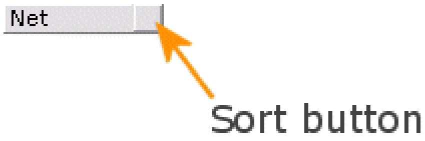
1 Open quote window's setup window.

2 Press the **Timer Sort** tab.

3 Change the interval and press **Apply** button.

Sorting Quote Columns

You can sort a quote column by pressing the sort button in a column.



Sort button

You can un-sort a column by choosing **Undo Sort** in the pop up menu. When undoing timer sort (auto ranking), quote window will revert back to the original order before timer sort happens.

Row Filtering

You can filter row by criteria. When a row is filtered, it is not displayed. Calculation for the row continues. When the criteria becomes true, the row will be shown again.

To specify the criteria for row filtering:

1 Open quote window's setup window.

2. Press the **Row Filter** tab.
3. Enter the criteria under **Condition Formula**. A row is displayed only if the formula evaluates to true for the symbol in this row. You can use any quote window formula, including indicator evaluation. The following are valid condition formula:

   ```
   last > 30
   ```

   ```
   RSIndex(0,M1,10) > 50
   ```
4. Press the **Apply** button to indicate the formula is used as a criteria for row filtering.

You will need to enable row filtering for the feature to work. To enable, either:

- Check the **Enable Row Filter** option under **Row Filter** tab in quote window setup, or
- Press the row filtering tool button ▶ , or
- Choose **Start Row Filtering** in quote window's pop up menu.

When row filtering is on, you cannot modify the quote window. You must stop row filtering first before modifying.

NeoTicker® EOD does not support indicator evaluation in row filtering. You can evaluate indicators in row filtering condition.

Below explains the different row filtering options.

Enable Row Filtering

Enable row filtering.

Maintain Background Calculations Only

The is the default state to turn off row filtering.

The formula for row filtering is being updated by quote window constantly. However, the result of the formula is not applied on the rows.

This enables row filtering to take into effect immediately once you enable it.

Disable Row Filtering

The formula for row filtering is not updated. So when you enable row filtering, the formula must be applied to each symbol individually and it takes time for row filtering to stabilize.

Use this option to disable row filtering if you have a complex row filtering formula that can consume large amount of resources.

Sorting Lock

Turning on sorting lock will prevent the quote window from being sorted, i.e. rows getting automatically re-arranged. This is a useful feature when you do not want to accidentally sort the rows and mess up the order of the quote window.

To turn on sorting lock, either:

- Press the Lock Sorting tool button 🔒, or
- Choose **Lock Sorting** in the pop up menu.

When sorting lock is on, you cannot use any of the sorting feature such as the sort button on the column heading, or timer sort. You can still manually arrange the rows by dragging and dropping the rows, enter new rows and delete rows.

Available Fields

When you reference a field in quote window formula, replace % with Pct (e.g. use NetPct when referencing Net% in formula). If a fields starts with a number, add the character f at the beginning For example, the field 52WeekHigh can be used in the formula language using the name `f52WeekHigh`.

Quote window supports the following fields:

Field	Meaning
52WeekHigh	52 Week High. See ***Fundamental Data Support*** (on page 885).
52WeekLow	52 Week Low. See ***Fundamental Data Support*** (on page 885).
AccDnTick	Total number of down ticks of the day, accumulated after the symbol is added to the quote window
AccDnTick%	Percent of ticks that are AccDnTicks
AccDnVol	Total down volume for the day, accumulated after the symbol is added to the quote window
AccDnVol%	Percent of volume that are AccDnVols
AccNetTick	AccUpTick - AccDnTick
AccNetVol	AccUpVol - AccDnVol
AccTikRatio	AccUpTick divided by AccDnTick
AccTikTrin	TRIN indicator modified for real-time use. The definition is (AccUpVol/AccUpTick) / (AccDnVol/AccDnTick)
AccUpTick	Total number of up ticks of the day, accumulated after the symbol is added to the quote window
AccUptick%	Percent of ticks that are AccUpTicks
AccUpVol	Total up volume for the day, accumulated after the symbol is added to the quote window
AccUpVol%	Percent of volume that are AccUpVols

AccVolRatio	AccUpVol divided by AccDnVol
Ask	Current ask
AskExch	The exchange of the indicated Ask Price
AskSize	Current ask size
AskTradedVol	Volume of trades traded at ask price or higher
AskTrades	# of trades traded at ask price or higher
AvgPrice	(High + Low + Close) / 3
BATick10	The net trade at ask (+1) / trade at bid (-1) tick value of the last 10 ticks
BATick16	The net trade at ask (+1) / trade at bid (-1) tick value of the last 16 ticks
Bid	Current bid
BidAsk	Returns both bid and ask prices in compact form
BidAskMid	Middle between bid and ask price
BidAskSize	Return both bid and ask sizes in compact form
BidAskSizeSpread	The spread between the bid and ask size
BidAskSpread	The spread between the bid and ask price
BidExch	The exchange of the indicated Bid Price
BidSize	Current bid size
BidTick	The bid-tick indicator that indicator whether the bid price is an up bid
BidTradeVol	Volume of trades traded at ask price or lower
BidTrades	# of trades traded at ask price or lower
BPosAvgEntry	Average position entry price (brokerage). See ***Position Information as Quote Fields and Formula Functions*** (on page 782).
BPosSize	Average position size (brokerage). See ***Position Information as Quote Fields and Formula Functions*** (on page 782).

ColorCode	For display/assigning color codes. See ***Color Coding Symbols*** (on page 882).
DayClose	Regular trading hours closing price
DBRange	The range from HH to LL
DBRange%	The close position relative to the DBRange. DBRange% equals 100 if close equals to HH
Delayed	Whether the symbol is delayed (data feed specific)
DividendPerShare	Dividend Per Share. See ***Fundamental Data Support*** (on page 885).
DividendYield	Dividend Yield. See ***Fundamental Data Support*** (on page 885).
DnFromClose	Returns 1 for Last price less than PrevClose
DnFromCloseVol	Returns volume for Last price less than PrevClose, 0 otherwise
DnFromMidpoint	Returns 1 for Last price less than MidPoint
DnFromOpen	Returns 1 for Last price less than Open
DnTick (*)	Total number of down ticks of the day
DnTick% (*)	% of ticks that are DnTicks
DnVol (*)	Total down volume for the day
DnVol% (*)	% of volume that are DnVols
EarningPerShare	Earning Per Share. See ***Fundamental Data Support*** (on page 885).
EBITDA	EBITDA. See ***Fundamental Data Support*** (on page 885).
Exchange	The exchange information assigned to the symbol either by the user or the datafeed vendor

Flags	The condition on which the symbol is in. The correctness of this information depends on the datafeed vendor 'U' - UPC11830 called 'H' - Halted 'F' - Fast Market 'C' - Market Closed 'S' - Split 'X' - X-Div
FirstLog	First log of symbol. See ***Symbol Log Tool Operation Guide*** (on page 934).
Formula	Custom formula column refer to Custom Formulas for detail
High	The day high
HH	The Highest High value of the lookup database
Last	The last traded price
LastLog	Last log of symbol. See ***Symbol Log Tool Operation Guide*** (on page 934).
Low	The day low
LL	The lowest low value of the lookup database
Marked	For marking symbols. See ***Marking Symbols*** (on page 881).
MarketCap	Market Capitalization. See ***Fundamental Data Support*** (on page 885).
MidPoint	(High + Low) / 2
Name	Company name or description of the symbol
Net	Net change of last traded price from previous closing price
Net%	Percentage change from previous closing price
NetBidAskTradedVol	AskTradedVol - BidTradedVol
NetBidAskTrades	AskTrades - BidTrades
NetOpen	Net change since open of the day

NetOpen%	Percentage change from open price
NetTick (*)	UpTick - DnTick
NetVol (*)	UpVol - DnVol
Notes	User's personal notes on the symbol
Open	The open price
OpenInt	Open interest
PERatio	P/E Ratio. See ***Fundamental Data Support*** (on page 885).
PosComm	Commission for the position
PosCost	Position cost. This equals PosSize * PosPrice * Multiple
PosFromStopLoss	Returns the difference between the stop loss price and the current price. The value is positive if the stop loss is not reached yet
PosFromTarget	Returns the difference between the target price and the current price. The value is positive if the target is not reached yet
PosPL	Position profit/loss. This equals PosValue - PosCost
PosPL%	Position profit/loss percentage
PosPrice	Entry price for the position
PosSize	Position size
PosStopLoss	Stop loss price of the user defined position
PosTarget	Target price of the user defined position
PosValue	Value of the stock. This is the last price of the stock times PosSize
Prev	Previous closing price
PriceBookRatio	Price Book Ratio. See ***Fundamental Data Support*** (on page 885).

PriceSalesRatio	Price Sales Ratio. See ***Fundamental Data Support*** (on page 885).
Range	Today's range from High to Low
Range%	Today's close relative position in Range. Range% equals 100 if close equal to high
ShortRatio	Short Ratio. See ***Fundamental Data Support*** (on page 885).
Tick	Total number of trades for the day
Tick10	The net up (+1) / down (-1) tick value of the last 10 ticks
Tick16	The net up (+1) / down (-1) tick value of the last 16 ticks
Ticks10	The net up (+1) / down (-1) / side (0) tick value of the last 10 ticks
Ticks16	The net up (+1) / down (-1) / side (0) tick value of the last 16 ticks
TikRatio (*)	UpTick divided by DnTick
TikTrin (*)	TRIN indicator modified for real-time use. The definition is (UpVol/UpTick) / (DnVol/DnTick)
TradeExch	The exchange on which the last trade in record is traded at
TradeVol	Volume of the last trade. In some literature, this field is referred as trade size
UDTick	A character pattern of the latest 16 trades. Each trade is represented by either "+" for an UpTick (or a SideTick of an UpTick); "-" represents a DnTick (or a SideTick of a DnTick). The latest tick is the last character at the right side
UDSTick	Similar to UDTick with Sidetick represents by "=" character
UD	Returns 1 for uptick and -1 for downtick
UDS	Returns 1 for uptick, -1 for downtick 0 for sidetick
UpdateTime	The time when the quote is last updated

UpFromClose	Returns 1 for Last price greater than or equal to PrevClose
UpFromCloseVol	Returns volume for Last price greater than or equal to PrevClose, 0 otherwise
UpFromMidpoint	Returns 1 for Last price greater than or equal to MidPoint
UpFromOpen	Returns 1 for Last price greater than or equal to Open
UpTick (*)	Total number of up ticks of the day
Uptick% (*)	% of ticks that are UpTicks
UpVol (*)	Total up volume for the day
UpVol% (*)	% of volume that are UpVols
Vol	Total volume of the day
VolK	Total volume of the day in thousands
VolRatio (*)	UpVol divided by DnVol

(*) If you use an Internet data feed, there are special considerations before you can use these fields. See the ***Up Down Tick Related Fields for Internet Data*** (on page 876) section.

Quote Window Visual Settings

To adjust the visual settings such as color, font, row height of a quote window:

1 Open the quote window setup.

2 Press the **Visual** tab.

Grid Settings

Setting	Meaning
Row Height	Row height for all rows
Col Width	Column width for all columns
Rows	Number of rows. When you increase the number of rows, blank rows will be created. When you decrease number of rows, rows will be removed from the bottom of the table first
Allow New Rows	Whether you can add new rows directly in the grid
Heading On	Whether to show the quote headings
Window Border	Whether to show window border
Row Bar On	Whether to show row bar. Row bar is the left hand side display that indicates which row is active.
Tool Bar On	Whether to show the tool bar
Vertical Scroll Bar	Whether to display vertical scroll bar
Horizontal Scroll Bar	Whether to display horizontal scroll bar
Show Grid	Grid lines setting

Colors

Here are what the colors mean:

Color	Application
Font	Normal font color
Background	Background color
Up	Color when Net >= 0
Down	Color when Net < 0

New High Color to use if the security is making a new high comparing to the high of the previous trading day

New Low Color to use if the security is making a new low comparing to the low of the previous trading day

NH Period Color to use if the security is making a new high against the trading days stored in the EOD lookup table

NL Period Color to use if the security is making a new low against the trading days stored in the EOD lookup table

Grid Lines Grid lines color

New High, **New Low**, **NH Period** and **NL Period** all require the use of the EOD lookup table because they use historical information. You must make sure the EOD lookup table is enabled in server setup before the quote window can use these colors.

Writing a Note for a Symbol

You can write a note for a row by using the **Notes** field.

1. Add a column to the quote window by choosing the **Notes** field.
2. Double click on the cell under the **Notes** field.
3. An editor is open to let you enter notes for the row.

Specifying Trading Time and Holiday List in Quote Window

In quote windows, holiday list and trading time affect formula calculations that uses indicators.

1. Open the setup window of the quote window.
2. Press the **Data Settings for Formulas** tab.
3. Set the trading time and holiday list.
4. Press the **Apply** button.

You can use **Quick Fill** to fill in the trading time and holiday. The settings for **Quick Fill** is defined by time frame manager. Refer to ***Time Frame Manager*** (on page 1225) for more information.

Up Down Tick Related Fields for Internet Data

Read this section if you use an Internet data provider and want to use the following quote window fields. This section is not applicable if you use NeoTicker® with IDS:

The following quote window fields are related to up down ticks:

DnTick
DnTick%
DnVol
DnVol%
NetTick
NetVol
TikRatio
TikTrin
UpTick
Uptick%
UpVol
UpVol%
VolRatio

Internet data providers do not directly supply information about up down ticks. When you use an Internet feed, quote window calculates these fields using its real-time lookup table. In order for these fields to show values, you have to turn on the real-time lookup table.

Turning on real-time lookup table is a slow operation because all ticks have to be loaded. You should not set this option during trading hours.

Steps to turn on real-time lookup table:

1 Choose **Program>Server Setup**

2 Press the **Lookup DB** tab.

3 Check the **Enable RT Lookup DB option**.

4 Open the set up window for the quote window.

5 Press the **Misc** tab.

6 Enable the **Use RT Lookup values instead of quotes returned by the server** option.

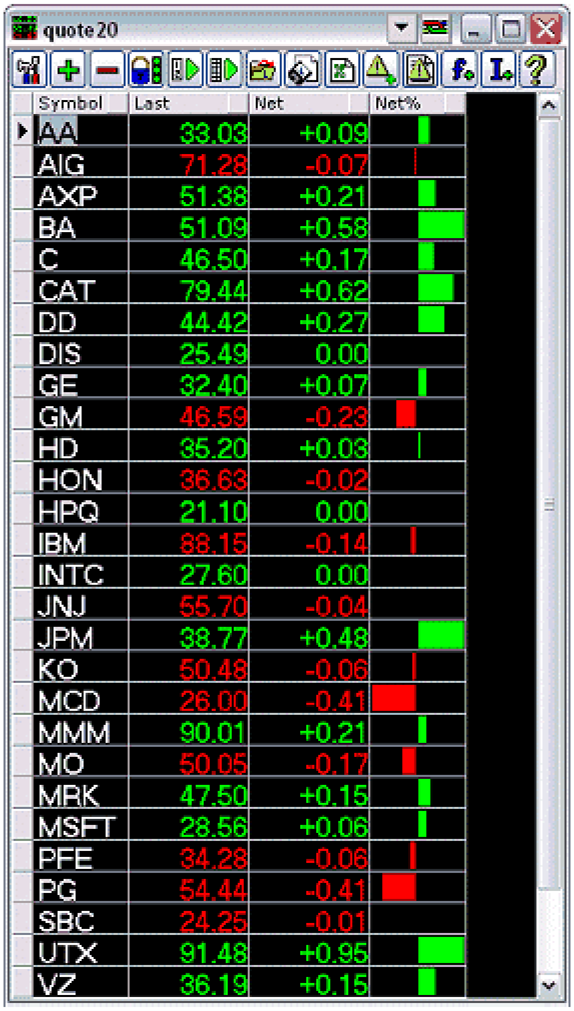
Because turning on these options can be very slow, you may want to consider using the accumulated version of these fields as an alternative. The accumulated version starts to collect up down tick information after a symbol is added to the quote window. Using the accumulated version will significantly improve performance.

The accumulated version of these fields are:

AccDnTick
AccDnTick%
AccDnVol
AccDnVol%
AccNetTick
AccNetVol
AccTikRatio
AccTikTrin
AccUpTick
AccUptick%
AccUpVol
AccUpVol%
AccVolRatio

Bar Graph

You can display a column as bar graph. In the following figure, net% is displayed as bar graph.



Symbol	Last	Net	Net%
AA	33.03	+0.09	🟩
AIG	71.28	-0.07	🟥
AXP	51.38	+0.21	🟩
BA	51.09	+0.58	🟩
C	46.50	+0.17	🟩
CAT	79.44	+0.62	🟩
DD	44.42	+0.27	🟩
DIS	25.49	0.00	
GE	32.40	+0.07	🟩
GM	46.59	-0.23	🟥
HD	35.20	+0.03	🟩
HON	36.63	-0.02	
HPQ	21.10	0.00	
IBM	88.15	-0.14	🟥
INTC	27.60	0.00	
JNJ	55.70	-0.04	
JPM	38.77	+0.48	🟩
KO	50.48	-0.06	🟥
MCD	26.00	-0.41	🟥
MMM	90.01	+0.21	🟩
MO	50.05	-0.17	🟥
MRK	47.50	+0.15	🟩
MSFT	28.56	+0.06	🟩
PFE	34.28	-0.06	🟥
PG	54.44	-0.41	🟥
SBC	24.25	-0.01	
UTX	91.48	+0.95	🟩
VZ	36.19	+0.15	🟩

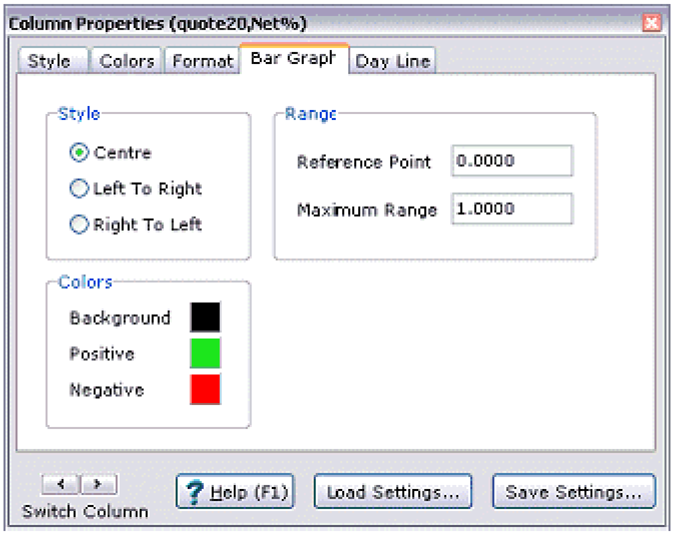
To set up bar graph:

1 Double click on column to open Column Properties dialog.

2 Press **Style** tab.

3 Choose **Bar Graph**.

Bar graphs options are under the **Bar Graph** tab in Column Properties.



Style

● Centre

○ Left To Right

○ Right To Left

Colors

Background ■

Positive 🟩

Negative 🟥

Range

Reference Point 0.0000

Maximum Range 1.0000

Style - Drawing style of the bar graph

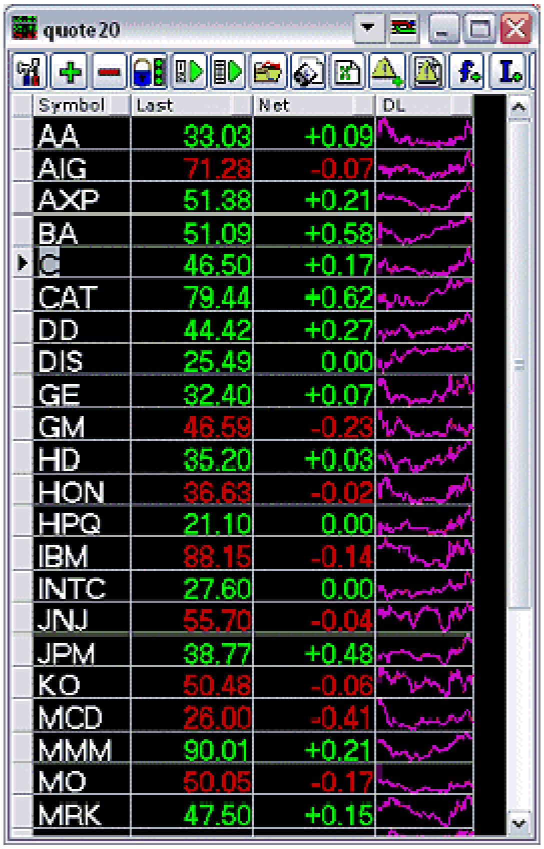
Colors - Colors of bar graph

Reference Point - The splitting point to determine positive/negative value. For example, for stochastic, you want to set this to 50.

Maximum Range - The value at maximum of the graph. For example, if the field returns 0-50, set this value to 50.

Day Line

You can display a column as day line. Day line ignores the column values, but use the column to display a mini 1-day intraday chart inside the cells. In the following figure, DL is a day line column.



Symbol	Last	Net	DL
AA	33.03	+0.09	
AIG	71.28	-0.07	
AXP	51.38	+0.21	
BA	51.09	+0.58	
C	46.50	+0.17	
CAT	79.44	+0.62	
DD	44.42	+0.27	
DIS	25.49	0.00	
GE	32.40	+0.07	
GM	46.59	-0.23	
HD	35.20	+0.03	
HON	36.63	-0.02	
HPQ	21.10	0.00	
IBM	88.15	-0.14	
INTC	27.60	0.00	
JNJ	55.70	-0.04	
JPM	38.77	+0.48	
KO	50.48	-0.06	
MCD	26.00	-0.41	
MMM	90.01	+0.21	
MO	50.05	-0.17	
MRK	47.50	+0.15	

To set up day line:

1 Create a new formula column.

2 In Column Properties dialog, set the formula to 0. This is simply a dummy value as day line column will ignore it.

3 Press **Style** tab.

4 Choose **Day Line.**

Day Line options are under the **Day Line** tab in Column Properties.

Entering Symbols

When you enter symbol into quote window, there are two possible behavior:

- Quote appears after you press ENTER key
- Quote appears as you type the symbol

The former is used for data feeds that impose a symbol limit (i.e. Internet feed, broker feed). This set up reduces the amount of "junk symbols" that are counted towards your symbol limit.

The latter is used for data feeds that do not have symbol limit (i.e. satellite feed). This set up provides instant feed back on quotes.

You can configure quote window to behave either way:

1 In main window, choose **Program>User Preference**.

2 Press **Quote Window** tab.

3 Toggle **Accept symbol while you type**.

Quote Window Copy and Paste

You can copy and paste quote window contents.

To copy:

1 Highlight the rows you want to copy. To highlight multiple rows, hold the SHIFT key while clicking on the row bar.

2 Press CTRL-C

To paste:

1 Press CTRL-V. When pasting to a quote window, only the symbols are pasted. When pasting to an external program such as Excel, all the cell values are pasted.

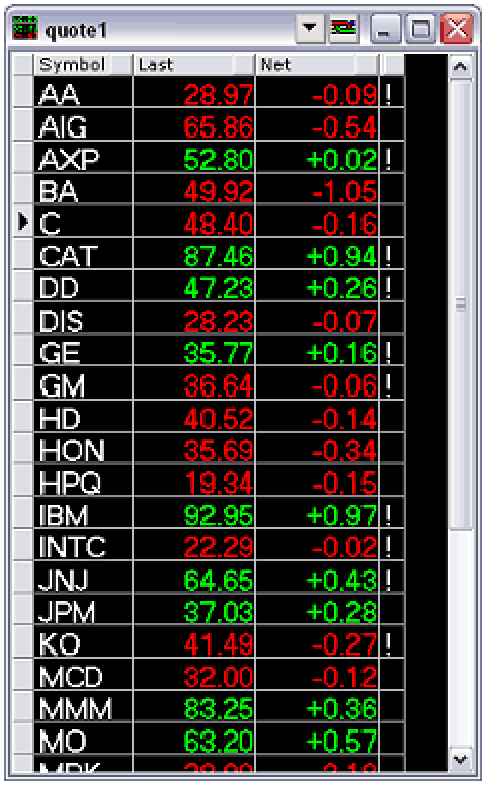
Marking Symbols

You can mark symbols for group operations. To mark symbols, you need to include the `Marked` quote field in the quote window.

1 Use pop up menu or set up window to add `Marked` field to quote window.

2 To mark a symbol, click on the cell under the `Marked` field. Marked symbols are shown with an exclamation mark under the `Marked` column.

3 To perform group operation, use one of the operation in pop up menu, under the **Marked Symbols** sub menu. For example, you can delete all marked symbols from the quote window.



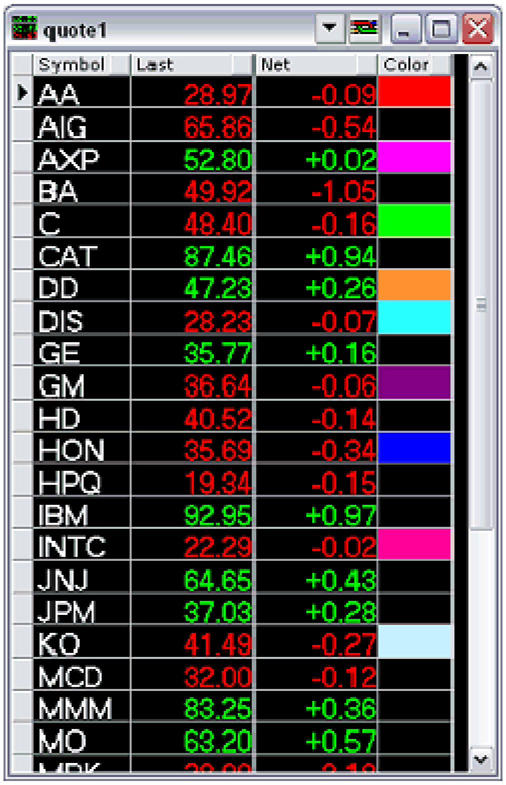
Symbol	Last	Net	
AA	28.97	-0.09	!
AIG	65.86	-0.54	
AXP	52.80	+0.02	!
BA	49.92	-1.05	
C	48.40	-0.16	
CAT	87.46	+0.94	!
DD	47.23	+0.26	!
DIS	28.23	-0.07	
GE	35.77	+0.16	!
GM	36.64	-0.06	!
HD	40.52	-0.14	
HON	35.69	-0.34	
HPQ	19.34	-0.15	
IBM	92.95	+0.97	!
INTC	22.29	-0.02	!
JNJ	64.65	+0.43	!
JPM	37.03	+0.28	
KO	41.49	-0.27	!
MCD	32.00	-0.12	
MMM	83.25	+0.36	
MO	63.20	+0.57	

Color Coding Symbols

You can assign color code to symbols. To color code symbols, you need to include the `ColorCode` quote field in the quote window.

1 Use pop up menu or set up window to add `ColorCode` field to quote window.

2 To assign a color code, click on the cell under the `ColorCode` field. A color dialog will be opened to let you select the color.



quote1

Symbol	Last	Net	Color
AA	28.97	-0.09	🟥
AIG	65.86	-0.54	⬛
AXP	52.80	+0.02	🟪
BA	49.92	-1.05	⬛
C	48.40	-0.16	🟩
CAT	87.46	+0.94	⬛
DD	47.23	+0.26	🟧
DIS	28.23	-0.07	🟦
GE	35.77	+0.16	⬛
GM	36.64	-0.06	🟪
HD	40.52	-0.14	⬛
HON	35.69	-0.34	🟦
HPQ	19.34	-0.15	⬛
IBM	92.95	+0.97	⬛
INTC	22.29	-0.02	🟥
JNJ	64.65	+0.43	⬛
JPM	37.03	+0.28	⬛
KO	41.49	-0.27	⬜
MCD	32.00	-0.12	⬛
MMM	83.25	+0.36	⬛
MO	63.20	+0.57	⬛

Adjusting Update Speed

Quote window updates periodically. By default, it uses a conservative update speed to avoid overloading slower computers. If you use a fast computer, you can increase the update speed for faster updates. Alternatively, if you have a very complex quote window involving many quote formulas, you can reduce the update speed to reduce the load on your computer. Based on our testing, most 2.0GHz computers can handle maximum update speed with no problem.

To adjust update speed:

1 Open quote window setup using either the quote window setup button or pop up menu.

2 In quote window setup, press the **Misc** tab.

3 Adjust the speed under **Update Speed**. The new update speed will take effect right away.

Receive Symbol List Option

When quote window receives a symbol list (e.g. from pattern scanner), it can carry out certain actions depending on the receive symbol list option.

To set receive symbol list option:

1. Open quote window setup using either the quote window setup button or pop up menu.
2. In quote window setup, press the **Misc** tab.
3. Choose the option under **Receive Symbol List Option.**

Possible actions are:

Option	Action
Sender Defaults	Let sender decides what action to take. By default, sender will replace the sy window.
Replace All Symbols	Replace the symbol in quote window with those in the symbol list.
Append New Symbols Only	Append to quote window symbols that are in the symbol list, but currently n window.

Fundamental Data Support

Fundamental data support is data vendor dependent. Also, fundamental fields are broadcasted by data vendors, they may not update in real-time.

eSignal

Available fundamental fields - 52WeekHigh, 52WeekLow, DividendPerShare, DividendYield, EarningPerShare, PERatio, PriceBookRatio.

Quote.com/QCharts

Available fundamental fields - DividendPerShare, DividendYield, EarningPerShare, MarketCap, PERatio, ShortRatio.

DTN.IQ/IQFeed

Available fundamental fields - 52WeekHigh, 52WeekLow, DividendPerShare, DividendYield, PERatio.

Report Window Operation Guide

Report window is a reporting facility for scripts. The report window is the destination where scripts can send text information to.

Scripts can send text messages to specific report window for many purposes. If you are developing an indicator, the report window is useful for debugging. If you are conducting a system test, you can use the report window to display the results.

Creating Report Window

To create a report window, either:

- Press the report window button REPORT, or
- Select **Window>New>Report** from the main window.

Basic Operations

To set or change any setting in the report window, you can use the pop up menu. To access the pop up menu, right click on the report window.

The following are features available under the pop up menu.

Item	Function
Max Lines	To set the maximum number of lines the report window can take. When Max Lines is exceeded, the report window removes the top most lines to stay within the limit
Font	Shows the font setting dialog. You can change the font type, style, color, etc.
Background	For setting the background color
Stick to Top	Forces the report window always showing the beginning part of the report data
Stick to Bottom	Forces the report window always showing the ending part of the report data
Normal	When the report window is not forced to show a specific region of the report data, it will act like a normal text window and stay at whatever is showing
Insert Blank Line	Inserts a blank line to the text.
Clear	Clears all the report data

File Name Associated with Report Window

By saving at least once to a specific file, a report window can save this file name with other settings, allowing you to re-save to the same location next time.

If you have specified such file name for a report window, scripts can then control the report window to save its content to this preset file name.

RTD / DDE Link Manager

This section is not about DDE data feed.

The RTD / DDE Link Manager manages the RTD and DDE links that NeoTicker® serves. In this case NeoTicker® is acting as a data server, while another program, e.g. Microsoft Excel, is acting as the client.

RTD (Real-Time Data) is Microsoft's latest technology to send real-time data from a server application such as NeoTicker® to Excel.

DDE (Dynamic Data Exchange) is an older technology to send data between application. It is supported by Excel and other third party applications.

We suggest users who want to use NeoTicker® with Excel to set up links using RTD in Excel. You should use DDE only for backward compatibility with applications that do not support RTD.

RTD is available only on Excel 2002 or later. See ***Excel Troubleshooting*** (on page 890) if you encounter connection problems in Excel.

Accessing RTD / DDE Link Manager

To access the RTD / DDE Link Manager, choose from main menu **Manager>RTD / DDE Link.**

Excel Samples

Excel samples are provided under the `Samples\Excel` directory within the NeoTicker®'s program folder.

The following example files are provided:

- `RTD Example.xls` example of using RTD with Excel 2003.
- `DDE Example.xls` example of using DDE with Excel.

Excel Troubleshooting

If the RTD / DDE links are not updating, check the following items:

- In Excel, under **Tools>Options>General**, **Ignore other applications** is unchecked.
- In Excel, under **Tools>Options>Calculation**, **Update Remote References** is checked.
- In Excel, under **Edit>Links**, all the links are set to **Update**, **Automatic**.

RTD Specific

You need Excel 2002 or later for RTD to work.

For Excel 2002, you need Service Pack 3 or later. You will to turn macro security to medium or lower (in Excel, choose **Tools>Macro>Security**, then set the security in the dialog).

How It Works and Topic Reference

With RTD / DDE, you can use NeoTicker® as a server to provide data and indicator calculations to Excel.

How It Works (DDE)

In Excel (or other client application), you will use an Excel formula that access the data from NeoTicker®. The formula tells Excel how to get data from NeoTicker® by specifying three items - Server Name, Topic Name, and Item Name.

Consider the following Excel formula:

```
=neoticker|q!'MSFT,last'
```

In this example, the Excel formula will request the last price of `MSFT` from NeoTicker®, where `neoticker` is the Server Name, `q` is the Topic Name, `'MSFT,last'` is the Item Name.

The Server Name `neoticker` is fixed. It tells Excel where to get the data.

The Topic Name `q` tells NeoTicker® to return a quote and the Item Name `'MSFT,last'` tells NeoTicker® what to return. We will discuss the supported Topic Names in later sections.

How It Works (RTD)

RTD is similar to DDE. You also need to supply the Server Name, Topic Name and Item Name. In Excel, you use the `RTD` function to get data from NeoTicker®.

Consider the following Excel formula:

```
=RTD("ntrtd.server",,"q","MSFT","last")
```

Similar to the DDE example, this formula also request the last price of `MSFT` from NeoTicker®. The differences are you use the `RTD` function to make the request, and the name for NeoTicker RTD server is ntrtd.server.

The main advantage of using RTD over DDE is you can easily use a reference cell instead of a string for request. For example, the following is a valid formula:

```
=RTD("ntrtd.server",,"q",RC1,R1C)
```

Note that Server Name is case sensitive, but NeoTicker® can process Topic and Item name in upper and lower case.

Quote Topic

Quote Topic returns a quote. It is requested with the Topic Name `q`. Each Quote Topic request returns text of comma separated values.

The related format for the Item Name is as follows,

```
Symbol,Field
Symbol,Field1,Field2,…,FieldN
```

`Symbol` is simply the symbol that you want to track.

`Field, Field1, … FieldN` are any quote fields (see ***Quote Window, Available Fields*** (see "Available Fields" on page 867)), up to 32 fields.

Here are some examples in Excel.

To get the symbol IBM's last price in Excel, type the following DDE formula in the cell:

```
=neoticker|q!'IBM,Last'
```

In RTD:

```
=RTD("ntrtd.server",,"q","IBM","Last")
```

To get IBM's open high low last of the day, select four (4) cells in a row and type the following DDE formula in to the first cell,

```
=neoticker|q!'IBM,open,high,low,last'
```

and to save the formula, you need to press CTRL-SHIFT-ENTER instead of the regular ENTER because Excel consider this as an array formula.

In RTD, use four formulas to retrieve open, high, low, last separately.

Quote Topic is updated in Version 4.1 of NeoTicker®. The old field `TradeVolK` is no longer supported.

Formula Topic

Formula Topic request NeoTicker® to do quote formula calculation and send the result to Excel in real-time. So you can reuse NeoTicker®'s indicator framework in Excel.

Formula Topic is requested with the Topic Name `f`. Each Formula Topic request returns a text value.

The format for the Item Name is as follows,

```
Symbol,TimeFrame_Type,Formula
```

`Symbol` is simply the symbol that you want to track.

`TimeFrame_Type` is one of the pre-defined names you have within the time frame manager. This will provide all the trading time and holiday information about the price bars that you want to generate.

`Formula` is a quote formula similar to those used in the quote window.

Here are some examples in Excel:

You want to obtain the latest value of the 100-period simple moving average of 5-minute IBM data using the "US Stock" time frame, simply type the following (DDE)

```
=neoticker|f!'IBM,US Stock,average (m5, 100)'
```

If you want the previous bar values (DDE)

```
=neoticker|f!'IBM,US Stock,average (1, m5, 100)'
```

If you need the difference between two exponential moving averages:

```
=neoticker|f!'IBM,US Stock,xaverage (m15,20) - waverage
(m30,50)'
```

The above formulas in RTD:

```
=RTD("ntrtd.server",,"f","IBM","US Stock","average (m5, 100)")
=RTD("ntrtd.server",,"f","IBM","US Stock","average (1, m5,
100)")
=RTD("ntrtd.server",,"f","IBM","US Stock","xaverage (m15,20) -
waverage (m30,50)")
```

Table Topic (DDE)

Table Topic returns a table of quotes. It is requested with the Topic Name `rt`. Each topic table is a table of price records. Each record is composed in one of the following ways (see ***Controlling Table Topic Format*** (on page 896))

```
DateTime,Volume,Open,High,Low,Close
DateTime,Open,High,Low,Close,Volume
```

The table format is in Excel's special xlTable format which allows very fast data transfer from NeoTicker® to Excel. If you are using a customized application to receive the data, you can refer to Microsoft Excel's specification of the xlTable format for more information on how to process such data.

The format of the related Item Name is as follows,

```
Symbol,TimeFrame,BarsToLoad,TimeFrame_Type
```

`Symbol` is the symbol you want to track.

`TimeFrame` is a short form of the time frame you want to track.

Period	Size	TimeFrame short form
Tick	10	`T10`
Second	15	`S15`
Minute	5	`M5`
Daily	1	`D1`

`BarsToLoad` specifies the number of records to return.

`TimeFrame_Type` is one of the pre-defined names you have within the time frame manager. This will provide all the necessary trading time and holiday information about the price bars that you want to generate.

Here are some DDE examples using Excel:

You want to obtain the last 100 bars of 5-minute IBM data using the “US Stock” time frame. Then select 100 rows by 6 columns. At the first selected cell, type the following:

```
=neoticker|rt!'IBM,M5,100,US Stock'
```

Since this is an array formula, you need to press CTRL-SHIFT-ENTER to enter the formula.

To obtain the latest 250 days of daily data of IBM using the “US Stock” time frame, select 250 rows by 6 columns, then type the following formula into the first cell selected cell:

```
=neoticker|rt!'IBM,D1,250,US Stock'
```

Press CTRL-SHIFT-ENTER to enter the formula.

Table Topic (RTD)

In RTD, you need a formula for each cell. You can use the following formula:

```
=RTD("ntrtd.server",,"RT","IBM","M5","100","US Stock",0,0)
```

To obtain the value for a cell. The last two numbers identify the row number and field index in the table. See the RTD example file for a complete example.

Formula Table Topic (DDE)

Formula Table Topic is requested with the Topic Name `ft`. Each topic formula table is a column of indicator values.

The column format is in Excel's special xlTable format which allows very fast data transfer from NeoTicker® to Excel. If you are using a customized application to receive the data, you can refer to Microsoft Excel's specification of the xlTable format for more information on how to process such data.

The format of the related Item Name is as follows,

```
Symbol,BarsToLoad,TimeFrame_Type,Formula
```

`Symbol` is the symbol you want to track.

`BarsToLoad` specifies the number of historical values to return.

`TimeFrame_Type` is one of the pre-defined names you have within the time frame manager. This will provide all the trading time and holiday information about the price bars that you want to generate.

`Formula` is a quote formula similar to those used in the quote window.

Here an example in Excel:

You want to obtain the last 100 bars of 5-minute IBM simple moving average using the "US Stock" time frame. Select 100 rows by 1 columns. At the first selected cell, type the following,

```
=neoticker|ft!'IBM,100,US Stock,average (m5, 20)'
```

Since this is an array formula, you need to press CTRL-SHIFT-ENTER to enter the formula.

Formula Table Topic (RTD)

In RTD, you need a formula for each cell. You can use the following formula:

```
=RTD("ntrtd.server",,"FT","IBM","100","US Stock","average (m5,
20)",0)
```

To obtain the value for a cell. The last number identifies the row number in the table. See the RTD example file for a complete example.

Checking Status

The **Status** tab in RTD / DDE Link Manager provides a summary of the links that NeoTicker® are serving.

Resetting Links

In case of lost connections, you will can reset the RTD / DDE connections:

1. Open DDE Link Manager.
2. Press the **Reset Links** button.

Enabling/Disabling RTD / DDE Link

1. Open DDE Link Manager.
2. Press the **Setting** tab.
3. Toggle the **Enable RTD / DDE Link** option.

Change the Serving Interval

1. Open RTD / DDE Link Manager.
2. Press the **Setting** tab.
3. Change the settings under **Timer Interval**. Each interval is for a topic served.

Change Table and Formula Table Chronological Order

1. Open RTD / DDE Link Manager.
2. Press the **Setting** tab.
3. Choose either **Chronological** or **Reverse Chronological** order.

Controlling Table Topic Format

1. Open RTD / DDE Link Manager.
2. Press the **Setting** tab.
3. Choose **Volume-Open-High-Low-Close** or **Open-High-Low-Close-Volume**.

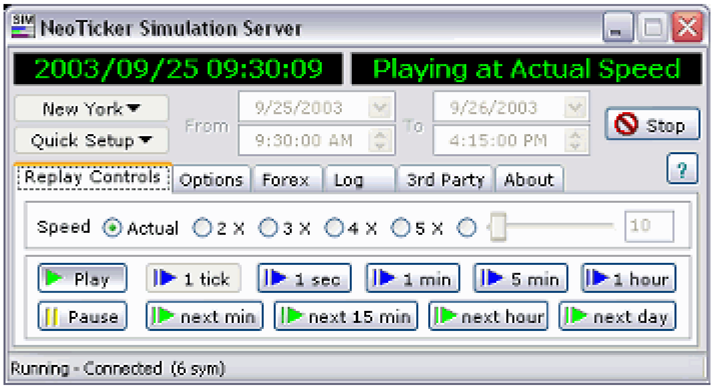
Simulation Server Operation Guide

Simulation Server is an environment simulation of intraday trading. When you use Simulation Server, NeoTicker® replays all ticks, bids and asks as if they are coming from a real-time server. The ticks come from multiple symbols and are played to all windows, recreating the environment as if the ticks are receiving in real-time.

NeoTicker® is the first commercially available program in the world to provide a complete trading environment simulation.

Simulation Server is suitable for:

- Replaying a trading scenario
- Fast forward analysis of trades from a trading day
- Step through analysis of data and indicator behavior
- Trading training, especially if the training hours are not trading hours



NeoTicker Simulation Server

2003/09/25 09:30:09

Playing at Actual Speed

From 9/25/2003 9:30:00 AM To 9/26/2003 4:15:00 PM

Speed ● Actual ○ 2 x ○ 3 x ○ 4 x ○ 5 x ○ 10

Running - Connected (6 sym)

What is Required to use Simulation Server

Simulation Server works on a data server level. It replaces a normal real-time data feed server by serving data files to NeoTicker®.

Simulation Server requires tick, minute and EOD data to work. Because Simulation Server replaces a real-time server, there is no connection to a real-time server when Simulation Server is running. Therefore, data must be installed prior to running Simulation Server.

Technically, Simulation Server uses the data stored in NeoTicker®'s disk cache. So if you have been using NeoTicker® for a while, you will have some data already present in the disk cache. However, it is recommended that you install some kind of data manually into the disk cache prior to running simulation server. This ensures Simulation Server have access to complete set of data.

You can install data from:

- Sample data files come with NeoTicker® and on the NeoTicker® CD. This is the option to use if you do not want to subscribe to a data feed and do not mind limited number of symbols.
- Delay data feed subscription that provides historical tick, minute data and EOD data.
- Real-time data feed subscription that provides historical tick, minute data and EOD data.
- Data downloaded from TickQuest's Backfill Server.

Demo Version Restrictions

If you use the demo version of NeoTicker®, Simulation Server will automatically disconnect from NeoTicker® after 90 minutes of simulation time. This restriction does not apply on lease or full version of NeoTicker®.

How to Install Data for Simulation Server

There are different sources of data you can use:

- NeoTicker® comes with a sample data file. The symbol and history is limited.
- NeoTicker® CD comes with more symbols and longer history.
- Delay/Real-time data subscription. Each data vendor has different data availability. Check with the data vendor.
- Data downloaded from TickQuest's Backfill Server.

Installing data is a "one-time" operation. Once completed, you do not have to re-install data unless you want to run simulation on different symbols or different time period.

How to Install Sample Data File

NeoTicker® automatically installs sample data into your disk cache.

This is completely transparent. Sample data is ready to use once you start NeoTicker®.

The following sample data is automatically installed:

- The index symbols: $INDU, $SPX, $NDX, $TICK
- Stock symbols: MMM, AAPL
- S&P 500 e-mini symbol: ESD_F

All the symbols has complete tick data coverage for 2003/9/25 to 2003/9/26. Thus you can run Simulation Server on these two days for the symbols above.

How to Install CD Data

To install CD data, you can use Data Tool.

1. Put NeoTicker® CD into your CD-ROM drive.
2. Close down NeoTicker®. Starts Data Tool by choosing **Start>Programs>TickQuest>DataTool** in Windows.
3. Press the **Install CD Data Tool** button to open the Install CD Data Tool.
4. In Step 2 of of the Install CD Data Tool, choose the symbols you want to install data. For Simulation Server, you should choose the symbols that have all tick, minute and EOD data.
5. Press the **Finish** button.
6. Once the operation is done, you can close Data Tool.

For more information about Data Tool, refer to its own documentation.

How to Install Historical Data from Delay or Real-Time Data Feed

Because tick data is involved, retrieving data from data vendor is a timely operation. If many symbols across many days are requested, it can take several hours to complete this operation.

To install data from Delay or Real-Time Data Feed:

1. In NeoTicker®, make sure you are connected to a delay or real-time feed.
2. From main window, choose **Manager>Cache**. Cache manager will open.
3. Press the **Historical Data** tab in Cache Manager.
4. Press the **Options** button. Historical Data Retrieval Options dialog is opened.
5. Specify a symbol list under **Symbol Selection**. A symbol list is just a plain text file with each line containing a single symbol. You can use one of the pre-defined list, or you can easily create your own symbol list. Try to select/create a symbol list that contains only the symbols you plan to run in the simulation.
6. Make sure **Daily Data**, **Minute Data** and **Tick Data** options are all selected under **Data To Request**.
7. Press **OK** button to close Historical Data Retrieval Options dialog.
8. In Cache Manager, adjust the **From**, **To** settings for dates of data to request. Note that many data vendors do not store tick data for more than a few days. Consult with your data vendor to check.
9. Press **Request Data** button to start retrieving.

How to Download Data From TickQuest Backfill Server

To use TickQuest Backfill Server for Simulation Server, you must be running either a permanent or lease license of NeoTicker®. Demo version will not work.

To download data from TickQuest Backfill Server:

1. Go into no feed mode. From Main Window, choose **Program>Server Setup**. Choose **No Feed**. Press **OK** button and restart NeoTicker®.
2. Download the data as if they are from a real-time data service. See How to Install Historical Data from Delay or Real-Time Data Feed section above. Because NeoTicker® is now in no feed mode, downloading data will force NeoTicker® from Backfill Server, and not from your regular data vendor.
3. For current list of symbols supported by Backfill Server, see ***Backfill Server Webpage*** (http://www.tickquest.com/backfillserver.html).

Setting up Simulation Server

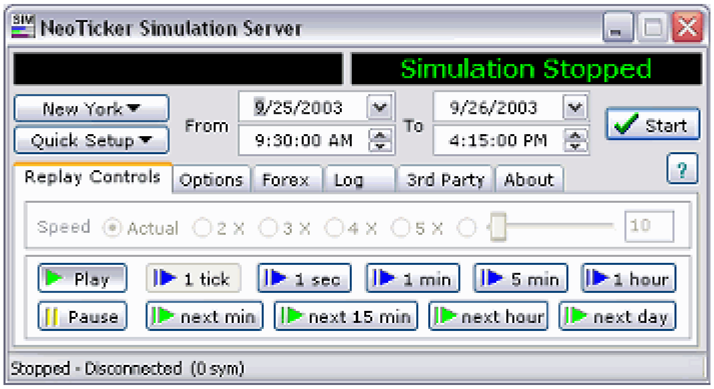
Once you have data ready, the next step is to set up NeoTicker® to use Simulation Server:

1. In NeoTicker®, choose **Program>Server Setup** to open server setup dialog.
2. Choose **Simulation**.
3. Press the **OK** button.
4. You will need to restart NeoTicker® for the change to take effect.

Running Simulation Server

Once you have Simulation Server setup, you can restart NeoTicker®.

Simulation Server is a separate program from NeoTicker®. NeoTicker®will launch Simulation Server automatically when NeoTicker® starts. When Simulation Server starts, you will see it:



NeoTicker Simulation Server

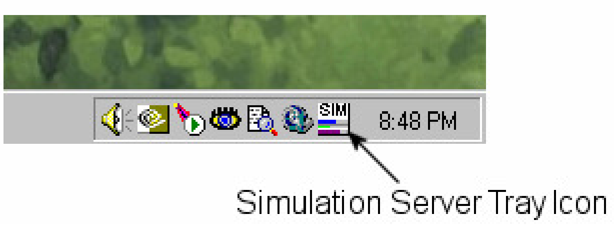
Simulation Stopped

From 9/25/2003 9:30:00 AM To 9/26/2003 4:15:00 PM

Speed ● Actual ○ 2 X ○ 3 X ○ 4 X ○ 5 X ○ 10

Stopped - Disconnected (0 sym)

Note: Simulation Server is minimized as a tray icon in the lower right corner of Windows task bar.



Before you start simulation, you need to set the start and end time of simulation:

1. In Simulation Server, set the start and end date time (the **From** and **To** settings). This is the period of time Simulation Server will be sending ticks to NeoTicker®.

The **Quick Setup** button can help you quickly set the start and end time. Under this button, you can choose **Sample Data** to use the pre-installed sample data. Simulation Server will then set itself to 2003/9/25.

2 By default, time zone is set to **New York** (EST). You can set to other city if you prefer your charts, quotes, etc to display in a different time zone. Data time stamp will be automatically adjusted to selected time zone.

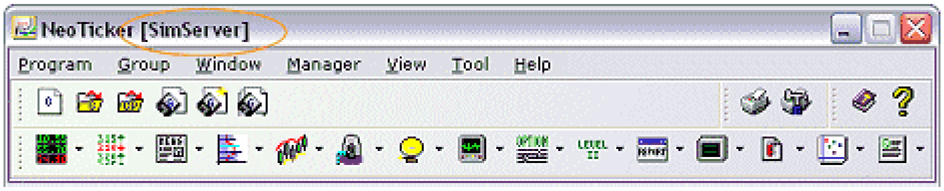
You need to make sure data is available in the time you specified. Remember that when you are running Simulation Server, data must already be present in your disk cache. NeoTicker® will not connect to your regular data vendor to obtain data.

Sample data is automatically installed for 2003/9/25 and 2003/9/26 for the symbols: $INDU, $SPX, $NDX, $TICK, MMM, AAPL, ESD_F.

The next step is to connect Simulation Server to NeoTicker®.

1. In Simulation Server, press the **Start** button.

After you press the **Start** button, NeoTicker® will automatically connects to Simulation Server. You can notice in Main Window, the title bar will have the words **SimServer** flashing.



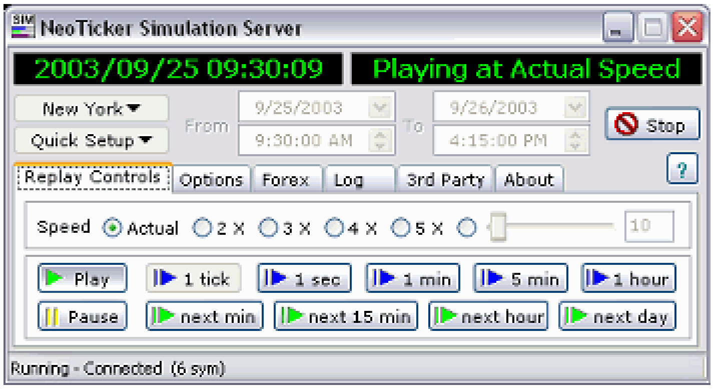
At this stage NeoTicker® is "frozen" at the time instance just before the simulation start time and no ticks are coming in. You can start adding charts, quote windows, etc. and the windows will load historical data. You can have multiple windows with multiple symbols open.

To start sending ticks:

1 In Simulation Server, press the **Replay Controls** tab.

2 Press the **Play** button.

When Simulation Server is sending ticks to NeoTicker®, NeoTicker® will behave like it's connected to a real-time server. You can expect features such as charts, volume distribution, bid ask data, etc. to work with Simulation Server. You can even place orders in Trade Simulator to practice simulated trading.



NeoTicker Simulation Server

2003/09/25 09:30:09 Playing at Actual Speed

From 9/25/2003 9:30:00 AM To 9/26/2003 4:15:00 PM

Speed ● Actual ○ 2 X ○ 3 X ○ 4 X ○ 5 X ○ 10

Running - Connected (6 sym)

Running Simulation at Different Speeds

To run simulation at different speeds:

1 In Simulation Server, press **Replay Controls** tab.

2 Choose either **2X**, **3X**, **4X**, **5X** or a customized speed.

In 2X speed, each second in real world time will corresponds to 2 seconds in simulation time. Thus 2X speed will play twice as fast as actual speed.

It is possible that simulation cannot play at the speed you specify. For example, if you choose 200X speed but your computer cannot catch up with the tick processing, Simulation Server will play at a slower speed than 200 times.

At 0X speed, Simulation Server will act as paused.

Skipping

Skipping works by fast forwarding to a specific time. It works similar to a chapter skip in a DVD player.

To skip a fixed time interval:

1. In Simulation Server, press **Replay Controls** tab.

2. Press either **1 sec**, **1 min**, **5 min** or **1 hour** button.

To skip to the next round time (e.g. 14:30:23 to 14:31:00):

1. In Simulation Server, press **Replay Controls** tab.

2. Press either **next min**, **next 15 min** or **next hour** button.

To skip a tick:

1. In Simulation Server, press **Replay Controls** tab.

2. Press the **1 tick** button.

You can only skip ticks if you have a single symbol in simulation. Otherwise, **1 tick** button is grayed out.

Pause vs. Stop

At anytime, you can press the **Pause** button in Simulation Server to suspend the tick sending. You can resume tick sending by pressing the **Play** button.

If you press the **Stop** button, the simulation will stop and NeoTicker® will disconnect from Simulation server.

Use pause if you want to temporary suspend tick sending. In pause mode, you can still use skipping feature and you can quickly resume tick sending.

Use stop if you want to adjust settings such as start date. These settings cannot be changed unless the simulation server has stopped.

Working with Trade Simulator

Trade Simulator simulates a real life broker. It has records of orders and transactions. The records persist over different NeoTicker® sessions. This presents a challenge when using Trade Simulator with Simulation Server, because it is not coherent to have trades in the future.

If you have Trade Simulator transaction that has a time stamp that is later than the current simulation time, when you place an order, NeoTicker® will give you the following choices:

- Backing up the current Trade Simulator data in a file, then clear Trade Simulator. You will be able to place orders right away. You may need to restore the data later.
- Leave data in Trade Simulator alone. You cannot place order until the simulation time passed the last transaction time in Trade Simulator.

You can also manually clear orders in Trade Simulator.

1. Choose **Manager>Trade Simulator** to open Trade Simulator.
2. Press **System Settings** tab in Trade Simulator.
3. Press **Reset Tracking** button to reset to default settings and clear all trades, or press **Clear All Trades** button to keep the current settings and clear all trades only.

Controlling Tick Arrival

All tick time stamps have resolution down to second only (e.g. 9:30:15, 9:30:16, etc). But in real-life, ticks arrive in different orders. Simulation Server provides options to let you control how ticks with the same time stamp are played.

The options here works up to 5x playback speed. For performance reason, when playback speed higher than 5x, all ticks with the same time stamp are played as a block.

To control tick arrival options:

1. If simulation is running, press the **Stop** button to stop it.
2. Press **Options** tab.
3. Check the tick arrival options.

In general, turning on the options will make tick arrival feels more natural. When these options are turned on, it takes more CPU time to process the ticks.

Randomize Tick Arrival Across Symbols

If multiple symbols have ticks with the same time stamp, the order in which the ticks are played is randomized.

Simulate Natural Tick Arrival in Sub-Second Time Frame

For a single symbol, if it has multiple ticks with the same time stamp, a random delay is introduced to make the ticks arrival spread out within the second.

Forex Tick Simulation

Forex data does not contains trade ticks, but you can ask Simulation Server to generate trade ticks based on bid/ask data.

1. In Simulation Server, press the **Forex** tab.
2. Press the **Edit Forex Symbol** List button to open a symbol list editor.
3. In the editor, enter symbols you want to generate trade ticks, one symbol per line. Press **OK** button.
4. Under **Trade Tick Option**, choose how trade tick is generated.

If **Bid** is chosen, a trade tick is generated with a bid tick. This is the usual convention.

If **Ask** is chosen, a trade tick is generated with an ask tick.

If **(Bid + Ask) / 2** is chosen, a trade tick is generated whenever a bid or an ask tick arrives.

Trouble Shooting

Simulation Server has a log page to help you track down problems:

1. Open Simulation Server.
2. Press the **Log** tab.

Limitations

Simulation Server is a simulation. There are subtle differences compared to a real-life server. Understanding the differences will help you use Simulation Server more effectively.

- Due to storage size requirement, Simulation Server cannot serve the following types of data: streaming news, options, Level 2, top list.
- Real-time data does not have sub-second time stamps. Thus if two ticks from different symbols has the same time stamp down to seconds, in Simulation Server, their tick arrival order is simulated. See ***Controlling Tick Arrival*** (on page 907).
- Real-time data does not have sub-second time stamps. Thus it will not be possible to provide slow motion playback.

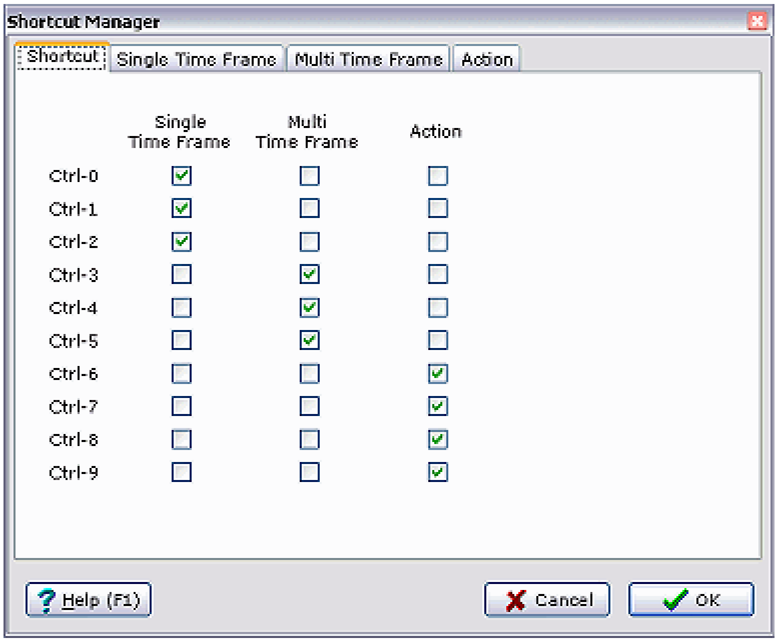
3rd Party Plugin

It is possible to use Simulation Server to send ticks to another application. Contact our sales team for more information.

Shortcut Manager Operation Guide

Shortcut Manager lets you define the keyboard shortcuts (CTRL-0 to CTRL-9).

To open the shortcut manager, choose **Manager>Shortcut** from the main window.

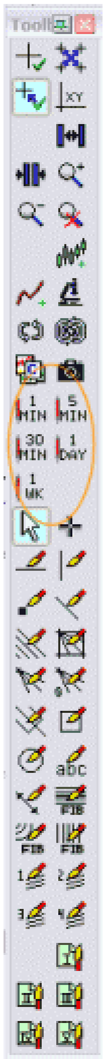


	Single Time Frame	Multi Time Frame	Action
Ctrl-0	☑	□	□
Ctrl-1	☑	□	□
Ctrl-2	☑	□	□
Ctrl-3	□	☑	□
Ctrl-4	□	☑	□
Ctrl-5	□	☑	□
Ctrl-6	□	□	☑
Ctrl-7	□	□	☑
Ctrl-8	□	□	☑
Ctrl-9	□	□	☑

Each shortcut key can map to one of the three action types - **Single Time Frame**, **Multi Time Frame** or **Action**. The Shortcut tab allows you to specify the mapping. For example, in the figure above, CTRL-0 to CTRL-2 are mapped to Single Time Frame; CTRL-3 to CTRL-5 are mapped to Multi Time Frame; CTRL-6 to CTRL-9 are mapped to Action.

You can use Single Time Frame and Multi Time Frame shortcuts only in time charts. You can use Action shortcuts anytime.

In addition to using keyboard, you can use the tool bar to fire a shortcut. The icons for shortcuts are illustrated in the figure below.



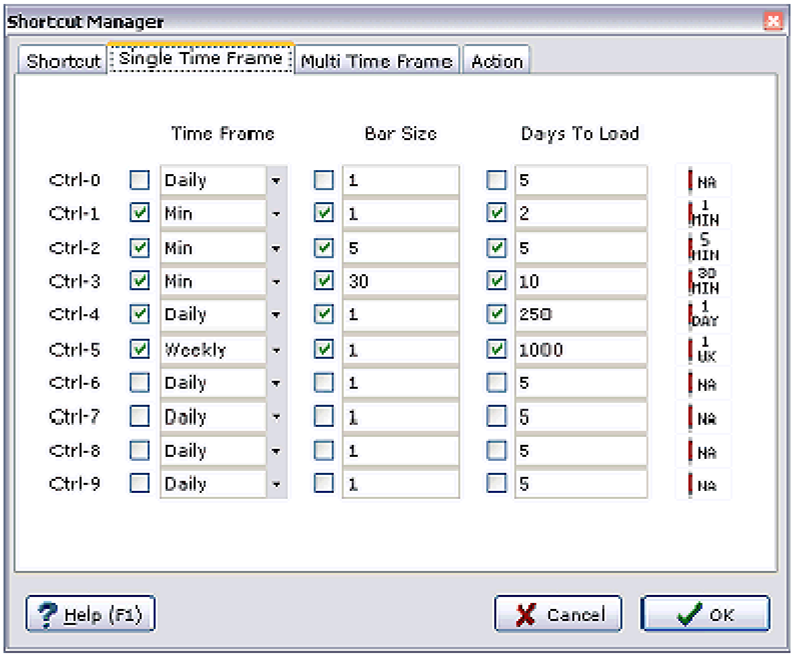
The shortcut icons on your tool bar may not look exactly the same as shown above, because the icons change dynamically according to the shortcut action.

Right clicking a shortcut button will open a pop up menu. From the menu, you can choose **Edit** to open Shortcut Manager.

The following sections explain the different types of shortcut actions.

Single Time Frame

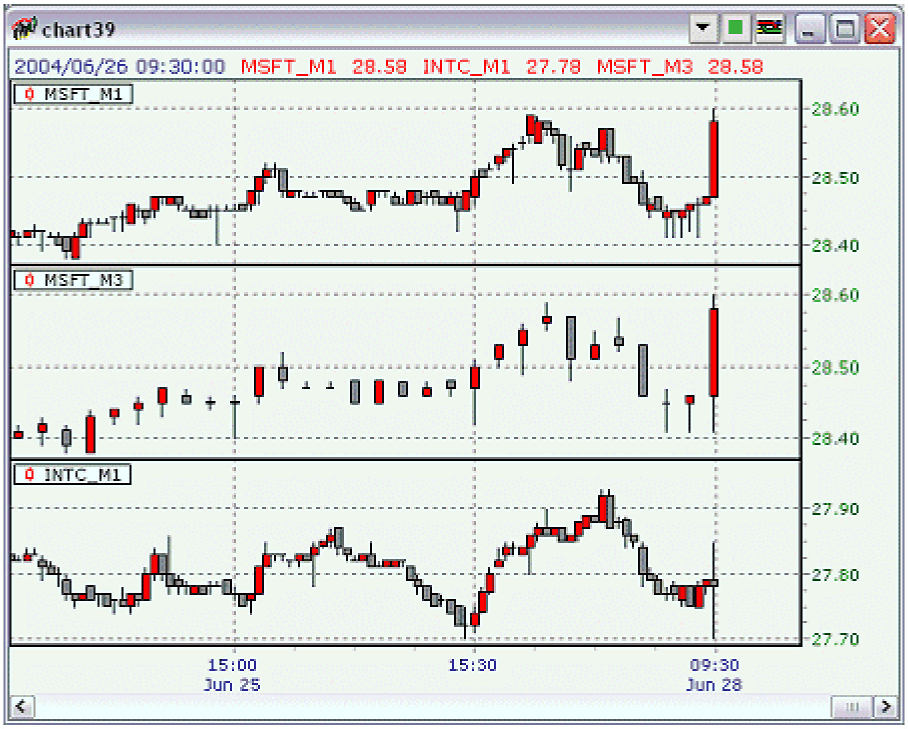
The purpose of Single Time Frame shortcuts is to quickly change one time frame to another. Use the **Single Time Frame** tab to configure the shortcuts.



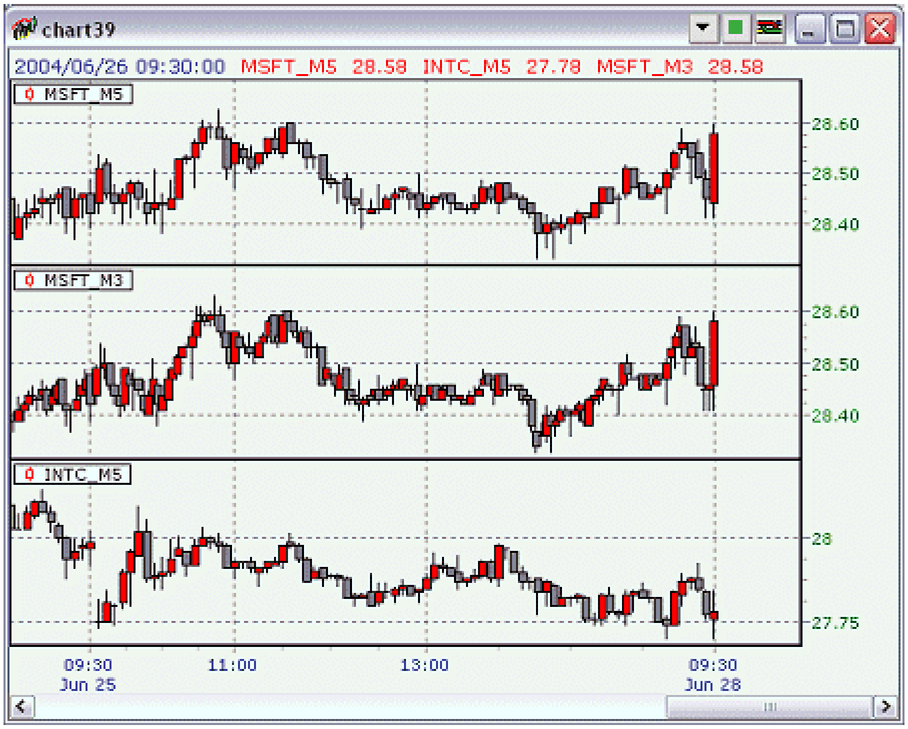
		Time Frame		Bar Size		Days To Load	
Ctrl-0	☐	Daily	☐	1	☐	5	NA
Ctrl-1	☑	Min	☑	1	☑	2	1 MIN
Ctrl-2	☑	Min	☑	5	☑	5	5 MIN
Ctrl-3	☑	Min	☑	30	☑	10	30 MIN
Ctrl-4	☑	Daily	☑	1	☑	250	1 DAY
Ctrl-5	☑	Weekly	☑	1	☑	1000	1 WK
Ctrl-6	☐	Daily	☐	1	☐	5	NA
Ctrl-7	☐	Daily	☐	1	☐	5	NA
Ctrl-8	☐	Daily	☐	1	☐	5	NA
Ctrl-9	☐	Daily	☐	1	☐	5	NA

When you use a Single Time Frame shortcut, the time frames for all data series that share the same time frame as the first data series are changed.

For example, in the following chart, there are three data series, 1-minute MSFT, 3-minute MSFT, 1-minute INTC.



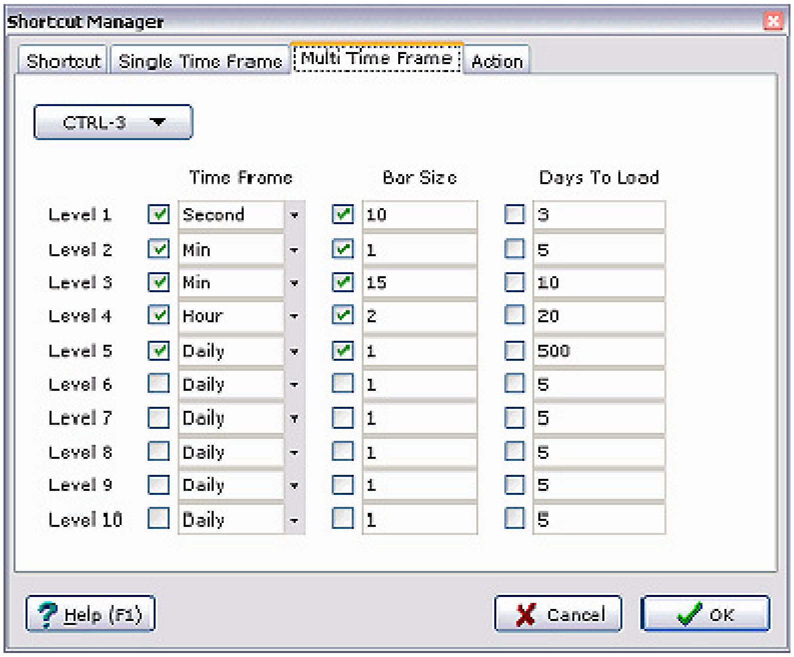
In this setup, CTRL-2 is configured to 5-minute, 5 days. So when you press CTRL-2 in the chart, the data series 1-minute MSFT changes to 5-minute MSFT. Also, because INTC is of the same time frame as MSFT 1-minute, the 1-minute INTC also changes to 5-minute INTC. In constrast, the time frame for 3-minute MSFT remains unchanged.



You can set time frame, bar size and days to load. Because the days to load is also set, these short cuts are very handy for quickly switching between intraday and end-of-day data.

Multi Time Frame

The purpose of Multi Time Frame shortcuts is to quickly change several time frames of the same symbol quickly. Use the **Multi Time Frame** tab to configure the shortcuts.

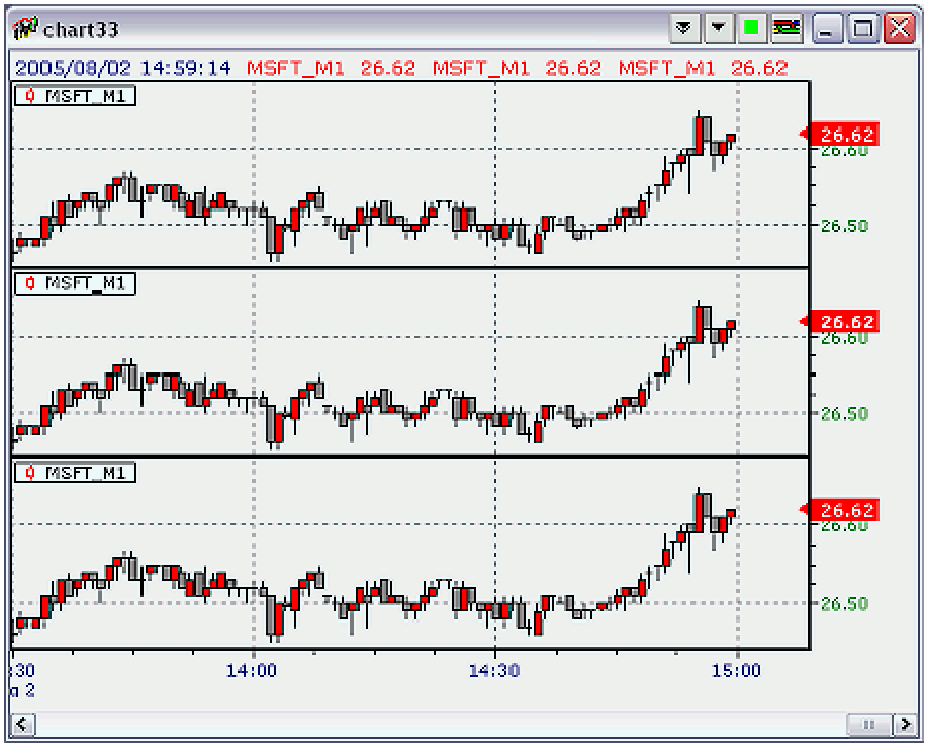


CTRL-3

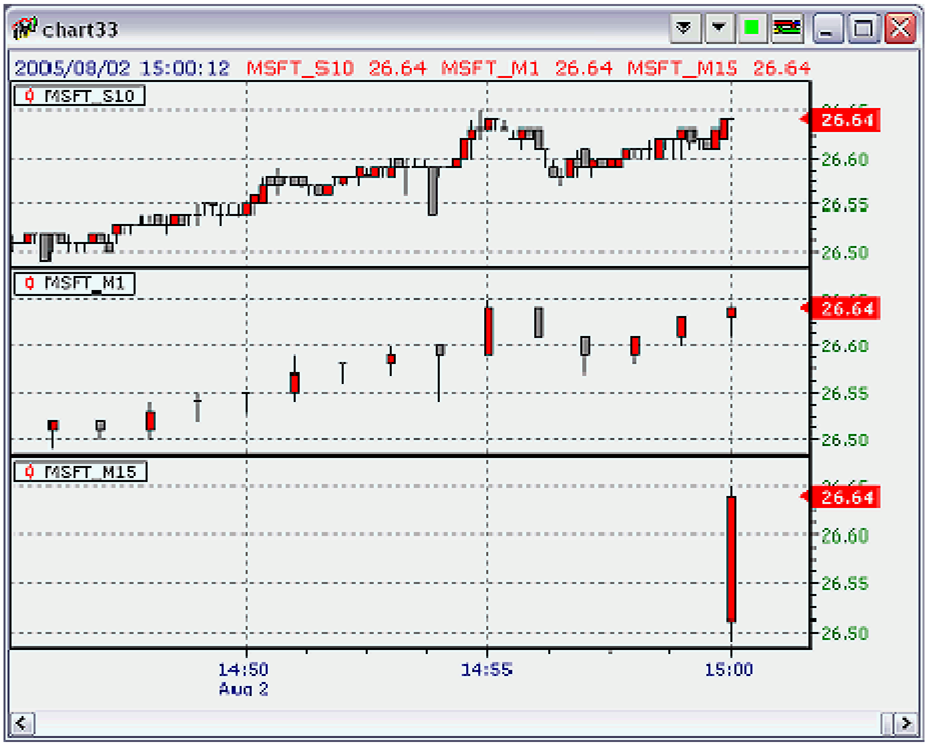
	Time Frame		Bar Size		Days To Load	
Level 1	☑	Second	☑	10	□	3
Level 2	☑	Min	☑	1	□	5
Level 3	☑	Min	☑	15	□	10
Level 4	☑	Hour	☑	2	□	20
Level 5	☑	Daily	☑	1	□	500
Level 6	□	Daily	□	1	□	5
Level 7	□	Daily	□	1	□	5
Level 8	□	Daily	□	1	□	5
Level 9	□	Daily	□	1	□	5
Level 10	□	Daily	□	1	□	5

The figure above shows the Multi Time Frame shortcut configuration for CTRL-3. What the configuration says is change the first occurrence ("Level 1") of a symbol to a 10-second time frame; second occurrence ("Level 2") of the symbol to 1-minute; third occurrence ("Level 3") of the symbol to 15-minute, and so on.

For example, the following chart contains three 1-minute MSFT series.

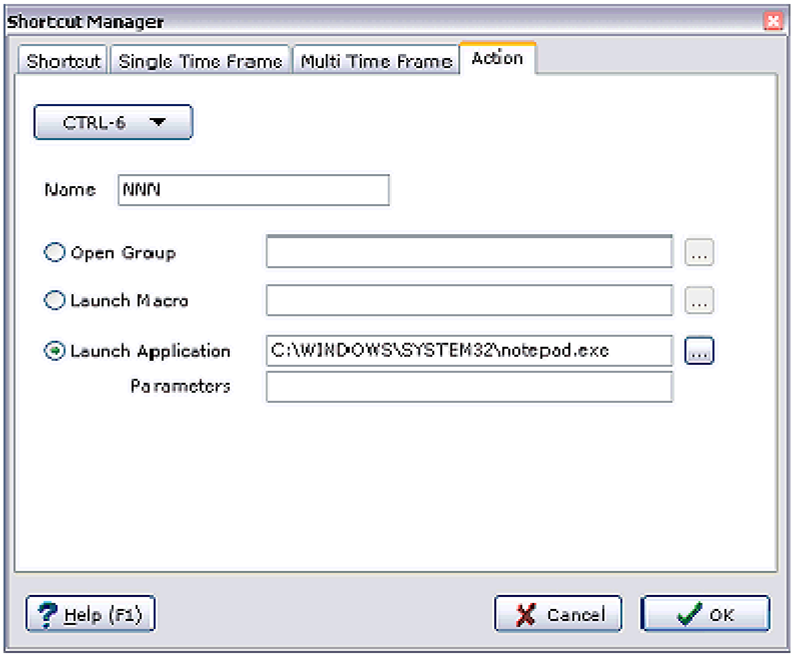


Pressing CTRL-3 will change the first series to 10-second, second series to 1-minute and third series to 15-minute.



Action

The purpose of Action shortcuts is to launch a specific action. Use the **Action** tab to configure the shortcuts.



CTRL-6

Name NNN

○ Open Group

○ Launch Macro

● Launch Application C:\WINDOWS\SYSTEM32\notepad.exe

Parameters

Name is an optional field to help you recognize the option. The first 3 characters of the value in the **Name** field will are displayed on the tool bar icon.

Open Group lets you specify an action that opens a group.

Launch Macro lets you specify an action that launches a macro. A macro is simply a script file that carries out certain actions. It is an OLE automation script (see ***OLE Automation*** (on page 653)).

Launch Application lets you specifies an action that launches an external application. In the figure above, CTRL-6 is configured to launch Notepad. **Parameters** are the optional parameter for the application. Parameter specification is identical to specifying parameters in a Windows shortcut.

Symbol Info Manager

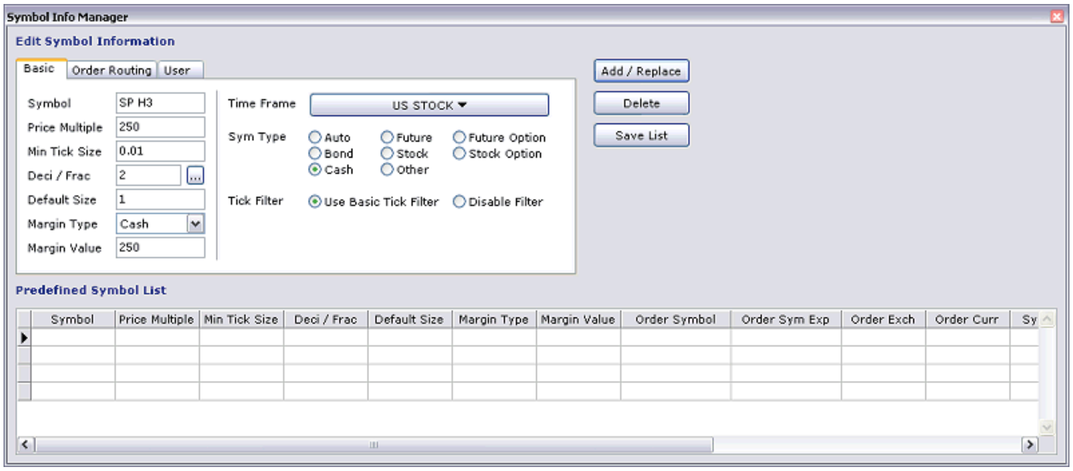
Symbol Info Manager let you define information of symbols. Trade Simulator, order forms, Backtest EZ, and other services in NeoTicker® can then use this information without you having to re-enter these data repeatedly.

For example, you can enter margin requirement for the symbol for trading systems that work on margin. For future contracts, information such as price multiples can be entered.

Basic Operations

Opening the Symbol Manager

To access the Symbol Info Manager choose from the main program window, **Manager>Symbol Info**.



Symbol: SP H3
Price Multiple: 250
Min Tick Size: 0.01
Deci / Frac: 2
Default Size: 1
Margin Type: Cash
Margin Value: 250

Time Frame: US STOCK

Sym Type: ○ Auto ○ Future ○ Future Option ○ Bond ○ Stock ○ Stock Option ● Cash ○ Other

Tick Filter: ● Use Basic Tick Filter ○ Disable Filter

Symbol	Price Multiple	Min Tick Size	Deci / Frac	Default Size	Margin Type	Margin Value	Order Symbol	Order Sym Exp	Order Exch	Order Curr	Sy

The top area is the Editor area for display and modification of symbol specifications.

Basic tab is for basic symbol information. These information is used by charting, trading systems, etc.

Order Routing tab is for information required to generate orders to send to your broker. These information is used when you are connecting to live broker, which may require additional information when placing orders (e.g. to which exchange orders are sent to, etc).

User tab lets you write down notes for a symbol. Information under this tab are not used by NeoTicker® and they are for your reference only.

The bottom area is the complete listing of all the pre-defined symbols.

Working on Symbols

Adding a Symbol

Fill in the Editor Area with a new symbol and its related symbol information. Then press the **Add/Replace** button to insert the new symbol into the **Predefined Symbol List**.

Editing a Symbol

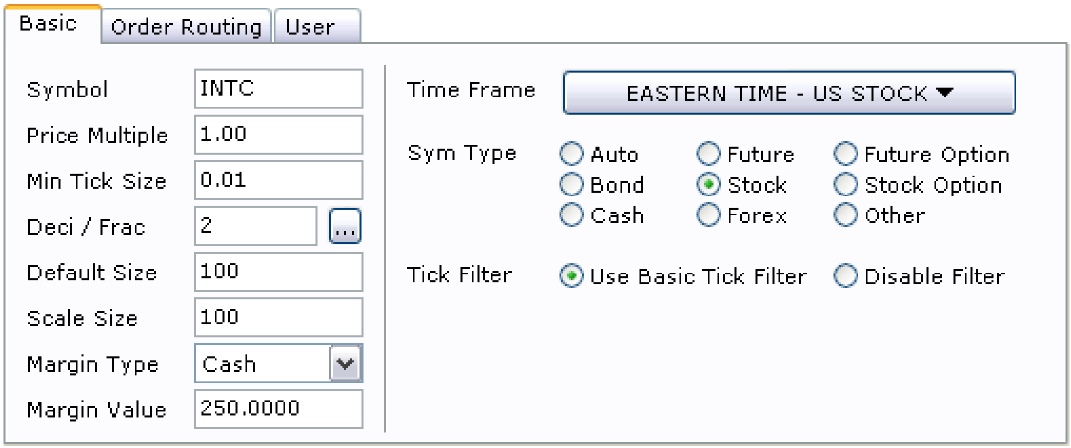
To edit an existing symbol, click on the symbol from within the **Predefined Symbol List**. The symbol and its data are then loaded into the Editor Area. You can make your changes and then press the **Add/Replace** button to save the updated information.

Deleting a Symbol

To delete a symbol, simply click on the symbol in **Predefined Symbol List**, and then press the **Delete** button.

Basic Tab and Related Fields

Basic tab is for basic symbol information. These information is used by charting, trading systems, etc.



Symbol: INTC

Price Multiple: 1.00

Min Tick Size: 0.01

Deci / Frac: 2 ...

Default Size: 100

Scale Size: 100

Margin Type: Cash

Margin Value: 250.0000

Time Frame: EASTERN TIME - US STOCK

Sym Type: ○ Auto ○ Future ○ Future Option ○ Bond ● Stock ○ Stock Option ○ Cash ○ Forex ○ Other

Tick Filter: ● Use Basic Tick Filter ○ Disable Filter

Symbol

Symbol is the unique identifier of the Symbol Info entry. There is only one entry per each symbol in the **Predefined Symbol List**. Symbol is used for matching the symbols you use in various services like Trade Simulator, Time Chart, etc.

Price Multiple

Price Multiple is the dollar value of the symbol based on absolute price movement of a single point. For example, stocks have a Price Multiple of 1 because each point move represents exactly $1. On the other hand, S&P emini contract has a Price Multiple of 50 because each point move represents $50.

Min Tick Size

Min Tick Size is the minimum change that the price of a symbol will fluctuate in. For example, stocks move in $0.01, thus Minimal Tick Size is 0.01. For S&P emini, Minimal Tick Size is 0.25.

Deci / Frac

Whether the symbol price is decimal or fraction. This setting affects trading system calculations (System Performance Viewer, System Monitor, fill price displayed in charts, etc).

Default Size

Default Size is the number of shares or contracts you usually trade with the particular symbol. For example, most day traders of stocks like to trade in 1000 shares per order. Thus the default size should be set to 1000. For most beginners who trade S&P emini, they trade 1 lot only, thus the default size should be 1.

Default Size is used by Trade Simulator and Order Forms to fill in the appropriate order size.

Scale Size

Scale Size is used for quick increment/decrement of size in order forms, trading systems, etc. For example, if you set MSFT **Default Size** to 1000 and **Scale Size** to 100, in an order form, MSFT will default to 1000 shares, and pressing the up/down buttons beside the order size will increment and decrement the order size by 100.

Margin Type

Margin Type is used by Trade Simulator and indicators with embedded trading system to calculate the margin required for any particular position.

For stock cash account, the **Margin Type** is **Cash** and the complete dollar value of an open position is used for the holding margin.

For stock margin account, the **Margin Type** is **Percent** and you can enter the percentage in **Margin Value** so that NeoTicker® can calculate the proper margin usage by the symbol.

For future account, **Margin Type** is a fixed **Amount** and you can enter the fixed amount in **Margin Value** for the calculation of the margin usage by the contract.

Margin Value

Margin Value is used together with Margin Type to calculate the cash usage by a position of the symbol.

Time Frame

The time frame for the symbol. Time frame is used in places like charting when information such as trading hours and holidays are required. For example, for a stock like MSFT, you should set the time frame to US Stock. If this information is not present, NeoTicker®'s ***Default Time Frame*** (on page 506) will be used.

Time Frame information is not shown in the bottom area of Symbol Info Manager.

Sym Type

Sym Type specifies the type of the symbol. This information is used throughout NeoTicker® when symbol type information is required. For example, pattern scanner can scan by symbol type.

Symbol Type information is not shown in the bottom area of Symbol Info Manager.

Tick Filter

Tick Filter is used for controlling real-time filtering of bad data. Depending on the data vendor, NeoTicker® may apply a tick filter to ticks coming from the vendor. This field allows you to customize this behavior for individual symbols.

For example, the Advance issue symbol for most stock markets fluctuate too much comparing to regular traded instruments. Thus if you are charting advance issue symbol, it is best you insert the symbol into the Symbol Info Manager and have the **Tick Filter** disabled.

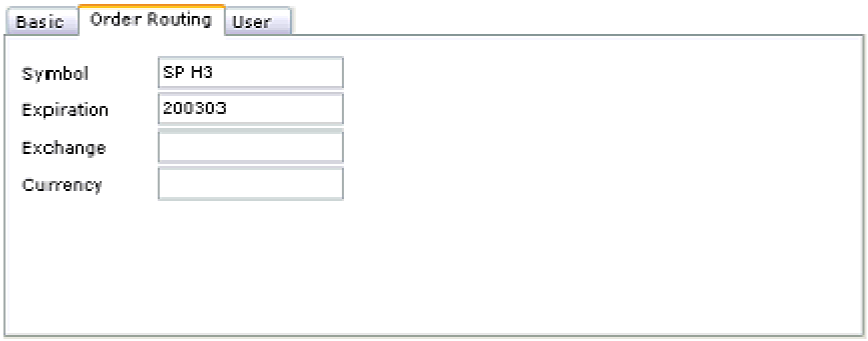
Order Routing Tab and Related Fields

Order Routing tab is for information required to generate orders to send to your broker. These information is used when you are connecting to live broker, which may require additional information when placing orders (e.g. to which exchange orders are sent to, etc).

Values for fields under this tab is broker dependant. Not all brokers required all of the fields listed here. We maintain a forum area to answer questions about order related information.

Order Interface Forum
(http://www.tickquest.com/forums/forumdisplay.php?forumid=30)

If you are not sure how to fill in order related information, please visit the forum and post a question there. It will be promptly answered.



Symbol	SP H3
Expiration	200303
Exchange	
Currency	

Symbol

Symbol specifies the symbol used when placing order. This field is useful when your data vendor and your broker use different symbology. This is a common issue with future contracts.

This field is listed under Order Symbol column in the bottom area.

Expiration

Some broker requires order expiration information when placing orders. **Expiration** is a text field that specifies this information. For example, 200303 specifies a 2003 March expiration for Interactive Brokers.

This field is listed under Order Sym Exp column in the bottom area of Symbol Info Manager..

Exchange

Exchange specifies which exchange the order should be placed, e.g. Globex.

This field is listed under Order Exch column in the bottom area of Symbol Info Manager.

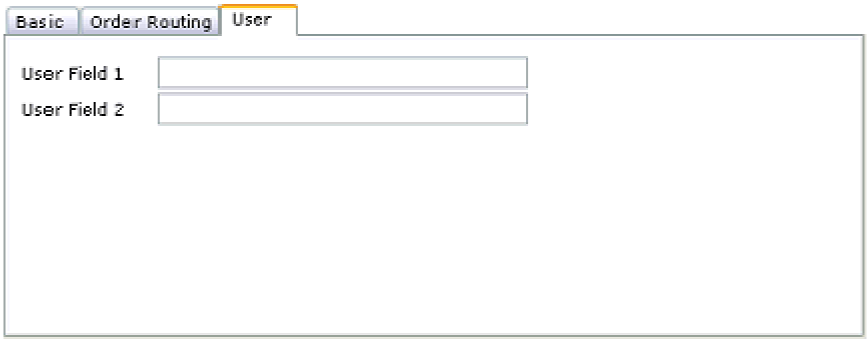
Currency

If an instrument (typically Forex) can be traded in multiple currencies, brokers may required an order to specify the settlement currency. This field is for specifying settlement currency.

This field is listed under Order Curr column in the bottom area of Symbol Info Manager.

User Tab and Related Fields

User tab lets you write down notes for a symbol. Information under this tab are not used by NeoTicker® and they are for your reference only.



User Field 1

User Field 2

User Field 1, User Field 2

You can use these fields to store information related the symbol. You can access field values by using the `NTLib.GetUserFields` function in scripts/IDL. See ***NTLib, Miscellaneous Routines*** (see "Miscellaneous Routines" on page 1624).

Save List

Normally, changes made to Symbol List Manager is saved when NeoTicker® exits. If you are entering many symbols, you may want to save your data entry from time to time in case of accidents like power outage, etc. Press the **Save List** button to save the complete set of Symbol Info data into your hard disk.

Symbol List Manager Operation Guide

Symbol list manager lets you quickly distribute symbols to different function windows. You can quickly go through many symbols and visualize them in function windows. This is especially handy for real-time monitoring of symbols.

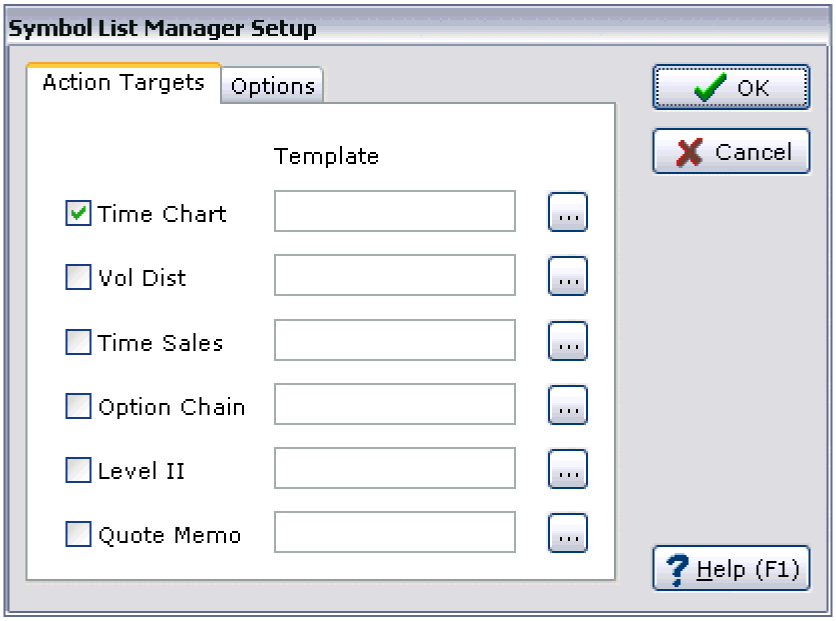
A tutorial is available to to help you getting start quickly on using symbol list manager: ***Tutorial: Monitoring Large Number of Symbols with Symbol List Manager*** (on page 189).

Controlling the type of Windows that Receive Distribution

1 Press the **Option** button in the symbol list manager. This opens the symbol list manager setup window.

2 Check the type of window you want to distribute symbol to.

3 Press the **OK** button.



☑ Time Chart

□ Vol Dist

□ Time Sales

□ Option Chain

□ Level II

□ Quote Memo

Creating New Windows

Press the **Create** button to create new windows from the symbols. You can control the template for window creation under the symbol list manager setup (press the **Option** button to open the setup window).

There is a safeguard to limit the number of windows in a group for window creation. The safeguard avoids the scenario of accidentally creating 500 windows when you are working on the S&P 500 list.

To change the safeguard limit:

1. Press the **Option** button to open the setup window.
2. Press the **Options** tab in the setup window.
3. Change the number for **Max windows per group**.
4. Press the **OK** button.

Custom Order for Symbol Distribution

If you prefer to distribute symbols from symbol list manager to a fixed set of function windows in a specific order, then you can set the custom distribution order of a window group.

1. Check the **Custom Distribute Order** option in symbol list manager.
2. Press the ▾ button to reveal the full symbol list manager.
3. Click the **Symbol List** button to open menu, and choose **View Custom Order**.
4. Open the group and window list window by choosing **Group>Group and Window List** from the main window.
5. On the group and window list window, click on the windows in the order that you want symbols to be distributed to.
6. Press the **Use Default Ordering** button on the symbol list manager.

Removing a Window from the Custom Order List

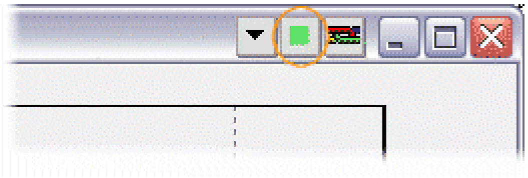
1. Click on the window you want to remove in the custom order list.
2. Press the DELETE key on the keyboard.

Append Missing

Press the **Append Missing** button to append the windows that are not currently listed within the distribution list. This can happen when you have added new windows to the group and you have not updated the list to include those new windows.

Distribute Override

Each function window has the distribute target property. This property is default to enable. If you disable distribute target, the window will no longer receive symbol distribution.



To turn on/off distribute target, either:

- Press the distribute target button on the title bar. A green button indicates the window is a distribution target. A red button indicates the window is not distribution target.
- Choose **Distribute Target** from the window's pop up menu, or
- Change the distribute target state in group and window list.

Distribute to Chain

Symbol list manager has a chain button to let you control which chain the symbols are distributed to. The chain button looks like this . Global chain is the default.

Refer to ***Window Chaining*** (on page 596) for more information on chaining.

Distributing the Symbols

Press the **Distribute** button. You will see the symbols being send to the windows that can receive symbol distribution (by default, all time charts). You can press the **Distribute** button again and more symbols will be distributed. By pressing the **Distributing** button, you can go through all the symbols quickly and visualize the symbols in the time charts.

Symbol list manager only distributes symbols to windows of the current group. You can create a group of windows that are specifically for distribution use.

Distribution Limit

The number of symbols distributed is limited. To change the setting:

1. Press the **Option** button to open the setup window.
2. Press the **Options** tab in the setup window.
3. Adjust **Max windows per group**

Press the **OK** button.

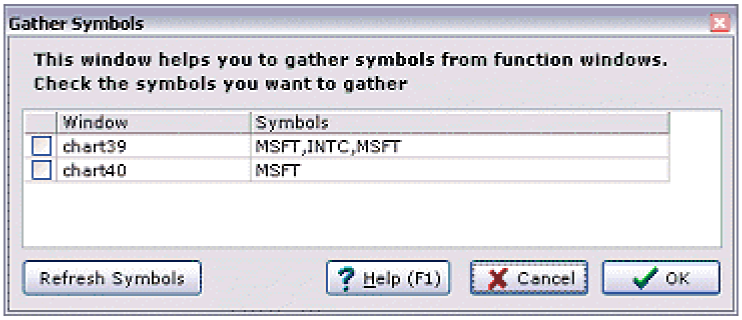
To help you match distribution target with distribution limit:

- Number of windows in the current group is shown as **Windows in group**.
- Number of windows that can act as a distribution target is shown as **Action targets in group**.

Gathering Symbols from Function Windows

One way to populate the Symbol List Manager is to use the symbol gathering function.

1. Expand Symbol List Manager.
2. If you are viewing custom order, press the **Custom Order** button to open menu, and choose **View Symbol List**.
3. Press **Symbol List** button to open menu, then choose **gather**.



Gather Symbols

This window helps you to gather symbols from function windows.
Check the symbols you want to gather

	Window	Symbols
□	chart39	MSFT,INTC,MSFT
□	chart40	MSFT

The symbols from all function windows will be listed. Check the option besides the window, and press the **OK** button to populate the symbol list manager.

Gather symbol will append the symbols to the current symbol list. Duplicated symbols are removed automatically.

Loading a Symbol List

Press the down triangle on the drop down menu. This lets you select one of the symbol list in NeoTicker®'s symbol list folder.

If you press the ▼ button, you will see the symbols displayed in the symbol list manager. You can add, remove, sort, clear or append the symbols. Changing the symbols will only affect the symbol list manager, but not the actual symbol list file.

Multiple Symbol Selection and Distribution

Normally when you distribute, the symbols are distributed such that all the function windows are filled or the max window per group is reached (changed in setup window). An alternative workflow is to multiple select the symbol by SHIFT clicking the symbols.

When multiple symbols are selected, only the selected symbol will be distributed.

Opening the Symbol List Manager

To open the symbol list manager, choose **Manager>Symbol List** from the main window.

Shown first is the minimized form of the symbol list manager. You can press the ▾ button to reveal all the options available in the expanded form.

Removing Duplicate Symbols

You can remove duplicate symbols by clicking on the **Symbol List** button to open pop up menu, and choose **Remove Duplicates**.

Reverse Distribution

You can distribute the symbols in reverse. To set up:

1. Press the **Option** button to open the setup window.
2. Press the **Options** tab in the setup window.
3. Turn on/off the option **Reverse Distribution**.
4. Press the **OK** button.

Turning off Last Symbol Repeat

Symbol list manager by default repeat the last symbol you distributed in the next distribution. This produces a "scrolling" feeling.

You can turn off the behavior by:

1. Open the symbol list manager by choosing **manager>symbol list** from the main window

2 Press the **Option** button

3 Press the **Options** tab in the option dialog

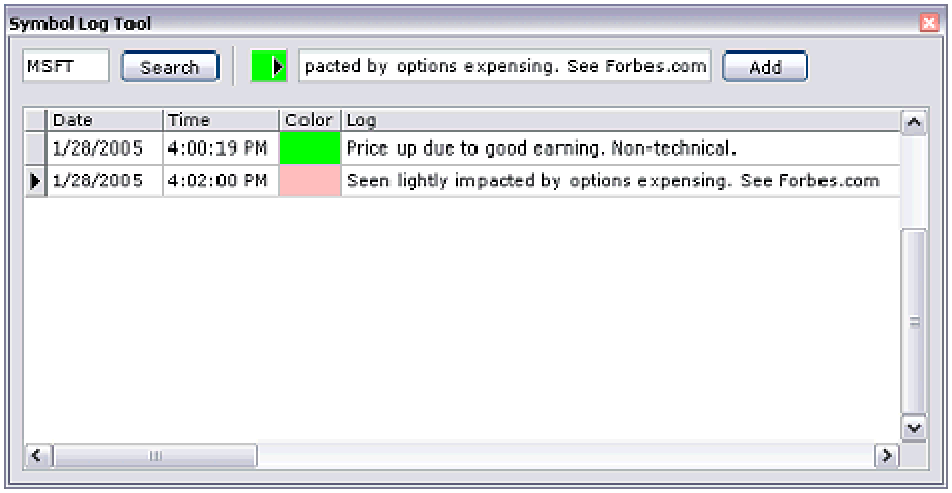
4 Check off **Keep Last Distributed Symbol**

5 Press the **OK** button

Symbol Log Tool Operation Guide

Symbol Log Tool is a tool to let you write down logs for symbols. A log is simple message about a symbol to help you remember something about a symbol. It can be anything - your trading notes, your thought about the symbol, etc. The logs are persist between NeoTicker® sessions.

To open Symbol Log Tool, from main window, choose **Tool>Symbol Log Tool**.



MSFT

pacted by options expensing. See Forbes.com

Date	Time	Color	Log
1/28/2005	4:00:19 PM	🟩	Price up due to good earning. Non-technical.
1/28/2005	4:02:00 PM	🟥	Seen lightly impacted by options expensing. See Forbes.com

Write Down a Log

To write down a log:

1 Type symbol in the Symbol field at upper left corner in Symbol Log Tool.

2 Enter the log in the Notes field.

3 Press **Add** button.

4 You can assign a color code to the log.

5 You can edit the log directly in the table.

Deleting Logs

To delete logs, right click on the table to open a pop up menu, and choose one of the deleting operation.

Searching

Logs are organized by symbols. To look for logs for a symbol, type the symbol in the Symbol field, then press the **Search** button.

Accessing Logs as Quote Fields

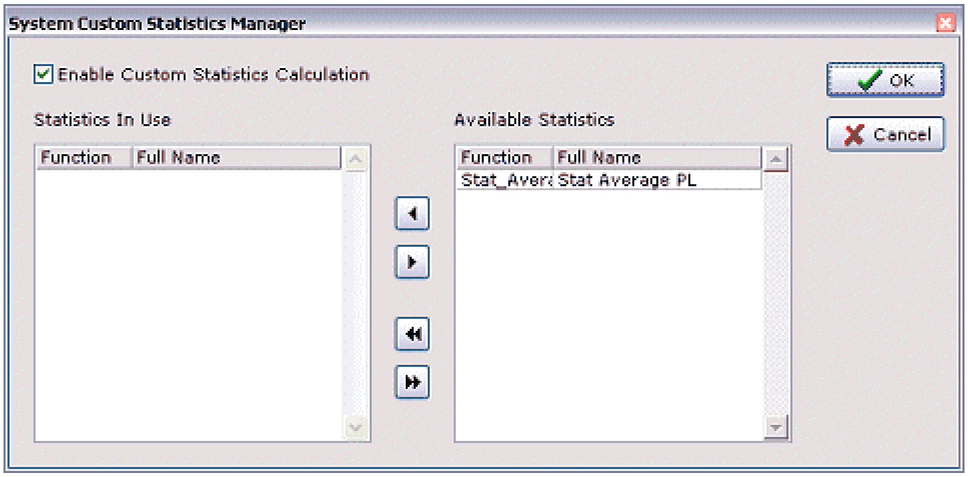
Quote fields are available to let you access logs in quote windows, dynamic windows, quote memos, etc.

Quote Field	Value
`FirstLog`	First log of the symbol.
`LastLog`	Last log of the symbol.

System Custom Statistics Manager Operation Guide

System Custom Statistics Manager lets you enable custom statistics calculation when you run a trading system. Custom statistics can be viewed by System Performance Viewer (***System Performance Viewe***r (on page 951)). Refer to ***Custom Statistics Programming Topics*** (on page 1515) for information on how to create custom statistics.

To open System Custom Statistics Manager, choose from main window, **Manager>System Custom Statistics**.



☑ Enable Custom Statistics Calculation

Statistics In Use

Function	Full Name

Available Statistics

Function	Full Name
Stat_Aver:	Stat Average PL

To enable custom statistics calculation, check **Enable Custom Statistics Calculation**.

Items listed under **Statistics In Use** are statistics that will be calculated when you run trading systems. Items listed under **Available Statistics** are custom statistics available for you to use.

You can press the ▶ and ▶▶ buttons to use a custom statistics. You can press the ◀ and ◀◀ buttons to remove custom statistics from being calculate.

System Monitor Operation Guide

System monitors are used for tracking the real-time changes in trading systems that are active in the currently opened time charts.

Every system monitor receives orders from all current trading systems in NeoTicker®.

Use System Monitor when:

- You prefer monitor trading systems and enter orders manually.
- Your broker is not supported by NeoTicker®.

Otherwise, you can configure Order Interface to place orders directly to a broker.

Enabling Monitoring in a Trading System

You will need to enable a trading system for it to be tracked by system monitor.

1 Edit the trading system.

2 Press the **System** tab.

3 Enable the option **Track in System Monitor** under **Visual**.

Creating System Monitor Window

To create a system monitor window, either:

- Press the system monitor button ▤, or
- Choose **Window>New>System Monitor** from the main window.

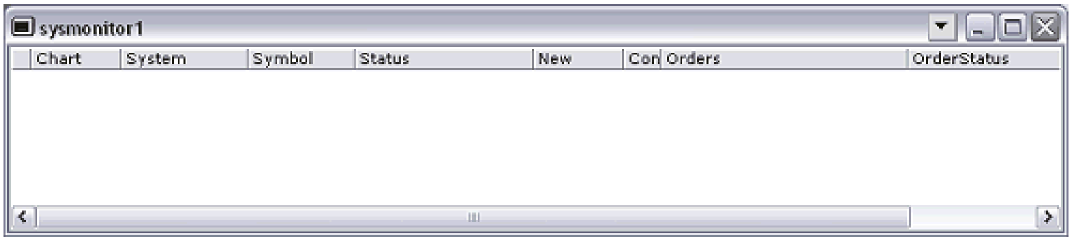
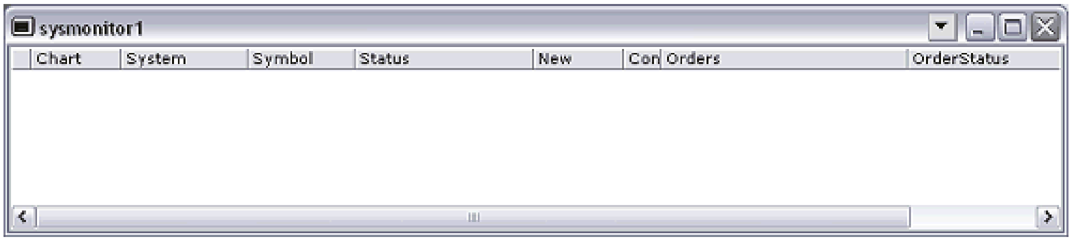


	Chart	System	Symbol	Status	New	Con	Orders	OrderStatus

You cannot type or modify the content of the system monitor window. This window is used only for monitoring trading system running in real-time.

Basic Operations



To set or change any setting in the system monitor window, you can use the pop p menu. To access the pop p menu, right click on the system monitor window.

To change the sorting order of the orders displayed in system monitor, click and drag the column headings to the location you desire. The order of the visible columns determines the order of which the system orders are listed and organized by.

To show or hide a particular column in the system monitor, choose the field name in the pop up menu to toggle its visibility.

All columns can be resized individually and the row height can be adjusted according to personal preference.

Columns

The following are the columns of system monitor window.

Column	Meaning
Chart	The chart in which the system object resides
System	The indicator name of which the system object resides
Symbol	The symbol of the data that the system is applied to
Status	The current open position status and equity level of the trading system
New	How new is this system order. N - The order is new to the system monitor 1-9 - Single digit numeric values indicate the number of minutes since the order was first listed in the system monitor. > - The order has been listed in the system monitor for 10 or more minutes.

Cond Condition of the order

O - Open

P - Pending

F - Filled

C - Canceled

ID Order ID created by the trading system.

Orders Description of the order (i.e. buy or sell how many of what). If the trading system has comments, the comments are also displayed in this field

OrderStatus The status of the order, whether it is open, filled at a specific time and price, pending to be filled at some specific price, or canceled.

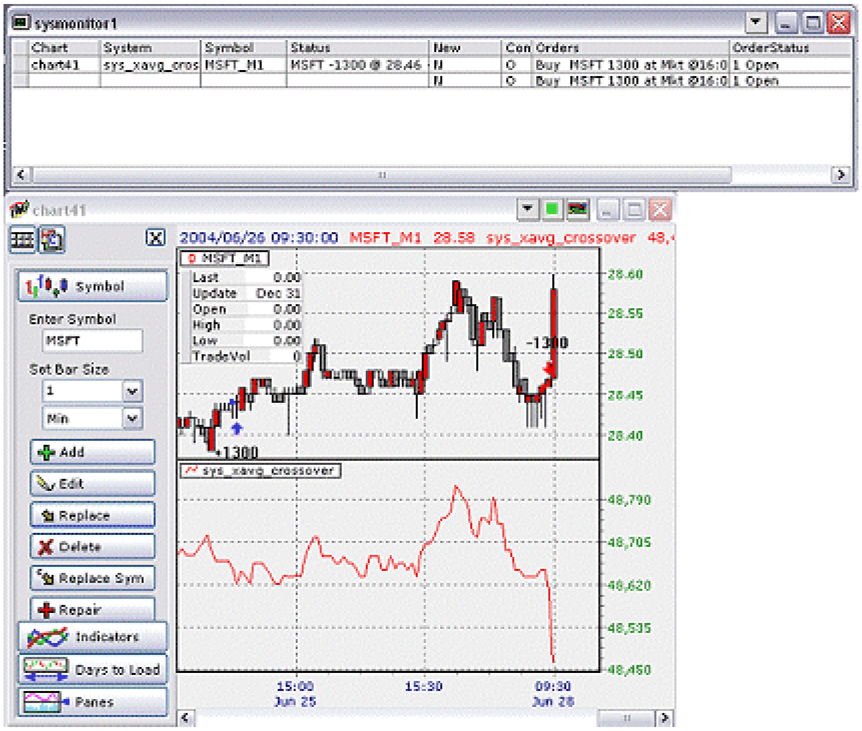
Using System Monitor Window

This example shows how the system monitor is being used.

1 Create a new time chart and a system monitor window.

2 Add the data series MSFT to the time chart. Then apply the indicator **XAverage Crossover System** to a new pane on the chart.

You will see something like the following.



sysmonitor1

Chart	System	Symbol	Status	New	Con	Orders	OrderStatus
chart41	sys_xavg_cros	MSFT_M1	MSFT -1300 @ 28.46	N	O	Buy MSFT 1300 at Mkt @16:0	1 Open
				N	O	Buy MSFT 1300 at Mkt @16:0	1 Open

chart41

2004/06/26 09:30:00 MSFT_M1 28.58 sys_xavg_crossover 48,

Enter Symbol: MSFT

Set Bar Size: 1, Min

Last	0.00
Update	Dec 31
Open	0.00
High	0.00
Low	0.00
TradeVol	0

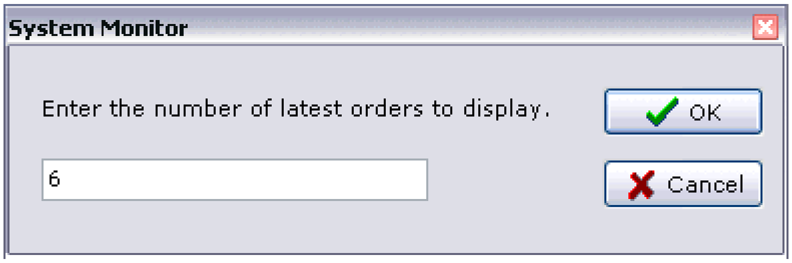
The indicator XAverage Crossover System is an example trading system provided with NeoTicker®. In NeoTicker®, trading systems are implemented as indicators. This trading system shows the equity curve of the trading system mark to each bar as its indicator value.

The system monitor window is showing the current state of the trading system.

Customizing System Monitor Window

Suppose instead of tracking just the open orders, you are interested in seeing the last six orders in the system. You can right click on the system monitor to open the pop up menu, and choose the item **Latest Orders**.

A dialog will be opened to let you enter the number of orders to display. For example, enter 6 and press the **OK** button.



System Monitor

Enter the number of latest orders to display.

6

The system monitor screen will look like this:

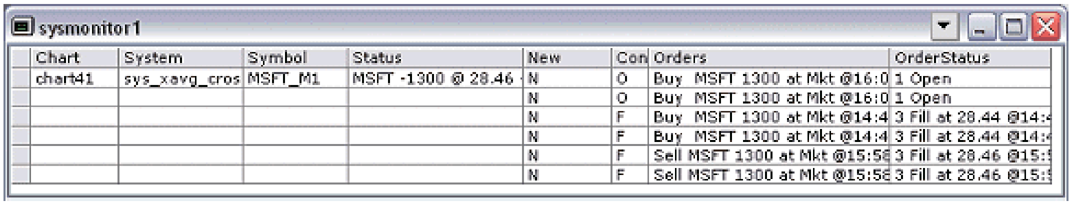


Chart	System	Symbol	Status	New	Con	Orders	OrderStatus
chart41	sys_xavg_cros	MSFT_M1	MSFT -1300 @ 28.46	N	O	Buy MSFT 1300 at Mkt @16:0	1 Open
				N	O	Buy MSFT 1300 at Mkt @16:0	1 Open
				N	F	Buy MSFT 1300 at Mkt @14:4	3 Fill at 28.44 @14:
				N	F	Buy MSFT 1300 at Mkt @14:4	3 Fill at 28.44 @14:
				N	F	Sell MSFT 1300 at Mkt @15:5	3 Fill at 28.46 @15:
				N	F	Sell MSFT 1300 at Mkt @15:5	3 Fill at 28.46 @15:

You can now tell what the system did in the last few orders.

Which System Orders to Display

Choose from the pop up menu **Open Orders Only** or **Latest Orders**.

Item	Meaning
Open Orders Only	Displays only the open orders of the listed systems
Latest Orders	When you choose this option, a dialog is opened for entering the number of latest orders to display. Under this option, the system monitor lists the latest N orders specified in each system being tracked

Visual Settings

Use the pop up menu to change visual settings for system monitor window.

Item	Meaning
Font	the user can customize the font setting through this option
Background Color	the background color can be changed

Tracking Down the Window that Issues the Order

Since system monitor tracks all active trading systems and reports their system status, you can have one single system monitor across all active groups instead of having one system monitor per group.

When you double click on an order, the system monitor takes you to the chart that generates that order and make that chart the active function window.

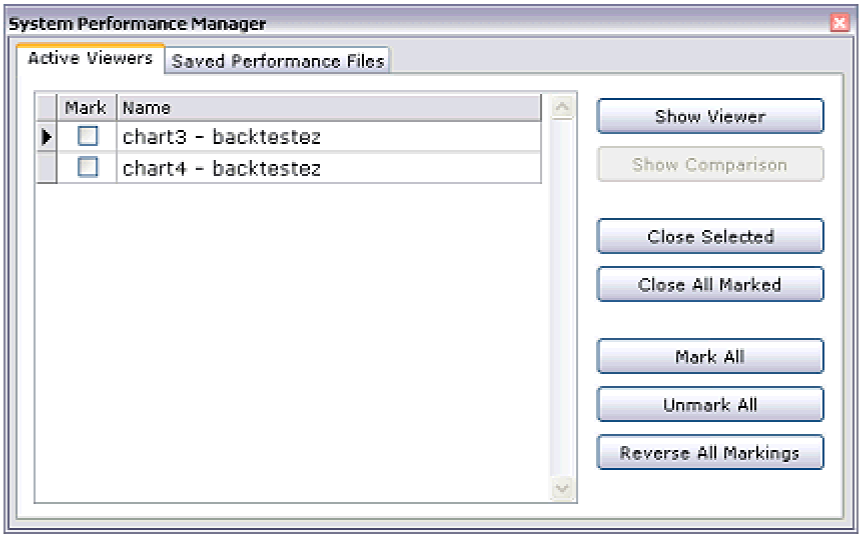
System Performance Manager

System Performance Manager enables you to review and organize System Performance Viewers and system performance files that you have saved from various sources, like indicators with embedded trading system within Time Chart, or the trading records saved within Trade Simulator. With System Performance Manager, you can easily recall and compare different results of either the same trading system with different parameter settings, or different trading systems applied to the same data.

Accessing the System Performance Manager

To open the System Performance Manager, choose from the main menu **Manager>System Performance**.

System Performance Manager will show up.



Mark	Name
□	chart3 - backtestez
□	chart4 - backtestez

Active Viewers

Under this tab, it shows a list of all the opened System Performance Viewers. When you open or close a System Performance Viewer, this list will be updated to reflect the latest changes.

Saved Performance Files

Under this tab, it shows a list of all the saved system performance files. When you save a new system performance file, or delete a system performance files, this list will be updated to reflect the latest changes.

Command Buttons

The right side of the window is the command buttons area. You can use these buttons to apply various operations to the selected items on the left hand side.

Viewing System Performance Viewer/Performance File

To open a System Performance Viewer or a system performance file, you can simply double-click on the item on the left hand side. Or, you can select the item from the list first, then press the **Show Viewer** or the **Open New Viewer** button.

When you open a performance file, a new System Performance Viewer window will be created and the system performance data from the file will be loaded for viewing.

While loading data from the system performance file, the System Performance Viewer window will indicate that it is not ready by showing the message **Loading** in the upper left corner of the window. When the data is completely loaded, the System Performance Viewer will display the proper name and the loading message will disappear. For more details on using the System Performance Viewer, please refer to ***System Performance Viewer*** (on page 951).

Saving a System Performance File from within Time Chart

When you want to save the performance data of an embedded trading system within an indicator in the Time Chart, you can select the indicator by click on its legend or the visual plots of the indicator. Then, right-click to bring up its pop up menu. Choose the item **Trading System>Save Performance File**.

All the trading system related information from orders being placed, to details of the equity fluctuations are saved to a new system performance file will date time stamp attached to its name.

The system performance file you choose to save will be available in the System Performance Manager once its saving is complete.

Deleting System Performance Files

Deleting a Singe System Performance File

To delete a specific system performance file, select the file from the list, then press the **Delete Selected File** button.

Deleting Multiple System Performance File

To delete multiple system performance files, you have to mark the files you want to delete first.

To mark a file, check the box to the left of the file name in the list.

There are buttons in the command button area that are designed to assist you marking the files.

The **Mark All** button will check all the files in the list for you.

The **Unmark All** button will uncheck all the files in the list for you.

The **Reverse All Markings** button will flip the check mark for all the files in the list.

Once you have completed your markings, you can press the **Delete All Marked Files** button to remove the files.

System Performance Viewer

System Performance Viewer allows you to review and analyze the performance of a trading system in details. Summary reports, charts, and other analysis are provided such that you can figure out new ways in improving your trading system faster and easier.

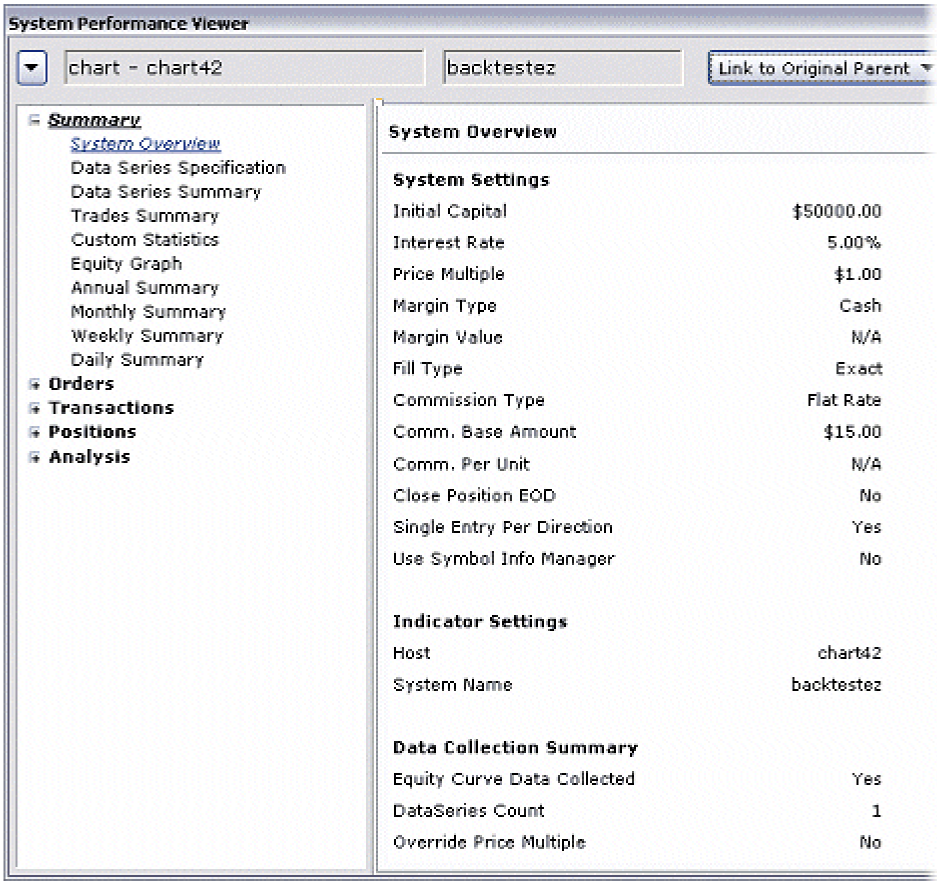
Opening System Performance Viewer from Time Chart

1 Select the indicator that has an embedded trading system (e.g. Backtest EZ)

2 Right-click to bring up the indicator pop up menu. Choose **Trading System>Open Performance Viewer** to open a new viewer window with the performance data of the selected indicator.

3 Alternatively, you can also choose from the pop up menu **Trading System>Save and Open Viewer** to save a copy of the performance data into a system performance file first then open the system performance viewer.

The System Performance Viewer window will show up.



System Overview

System Settings	
Initial Capital	$50000.00
Interest Rate	5.00%
Price Multiple	$1.00
Margin Type	Cash
Margin Value	N/A
Fill Type	Exact
Commission Type	Flat Rate
Comm. Base Amount	$15.00
Comm. Per Unit	N/A
Close Position EOD	No
Single Entry Per Direction	Yes
Use Symbol Info Manager	No

Indicator Settings	
Host	chart42
System Name	backtestez

Data Collection Summary	
Equity Curve Data Collected	Yes
DataSeries Count	1
Override Price Multiple	No

Opening System Performance Viewer from System Performance Manager

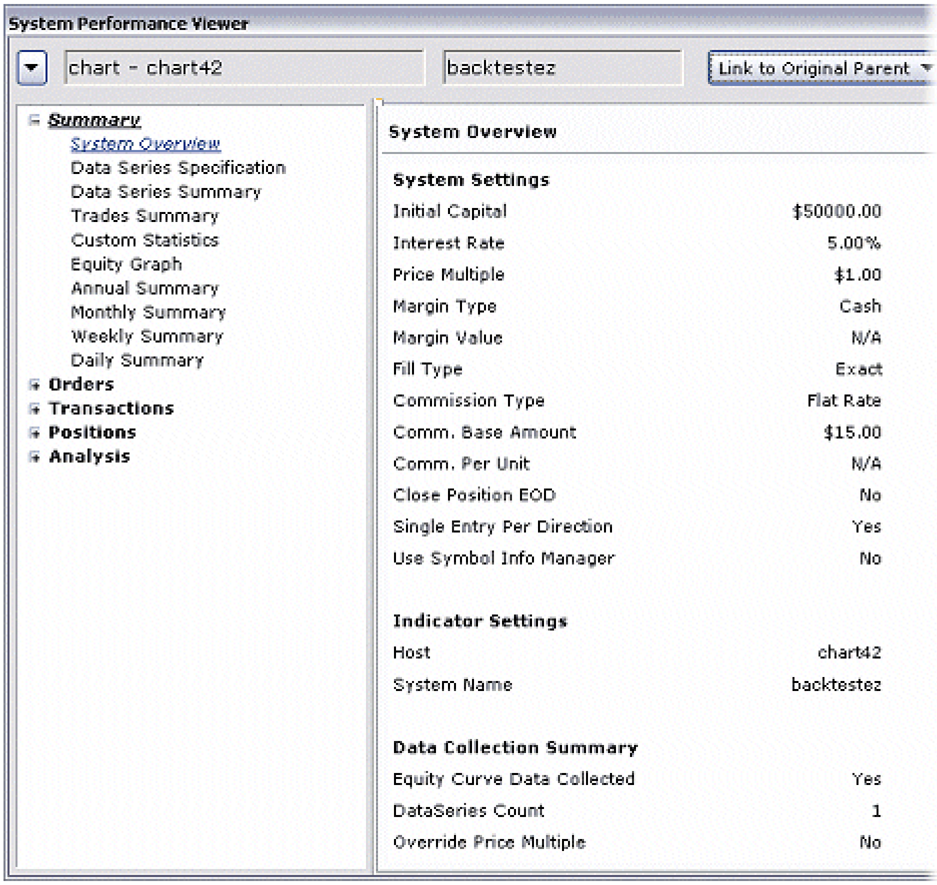
You can save an indicator's trading system performance by choosing from the indicator's pop up menu, **Trading System>Save Performance File** or **Trading System>Save Performance File As**.

Once a system performance file is saved, you can recall it from the System Performance Manager by double-clicking the performance file from within the list inside System Performance Manager.

Refer to the ***System Performance Manager*** (on page 947) for usage details.

Overview

After a system performance viewer window is opened, it will be defaulted to show the first page of its reports.



System Overview

System Settings

Initial Capital	$50000.00
Interest Rate	5.00%
Price Multiple	$1.00
Margin Type	Cash
Margin Value	N/A
Fill Type	Exact
Commission Type	Flat Rate
Comm. Base Amount	$15.00
Comm. Per Unit	N/A
Close Position EOD	No
Single Entry Per Direction	Yes
Use Symbol Info Manager	No

Indicator Settings

Host	chart42
System Name	backtestez

Data Collection Summary

Equity Curve Data Collected	Yes
DataSeries Count	1
Override Price Multiple	No

Top of the viewer has the parent window information listed together with quick access buttons.

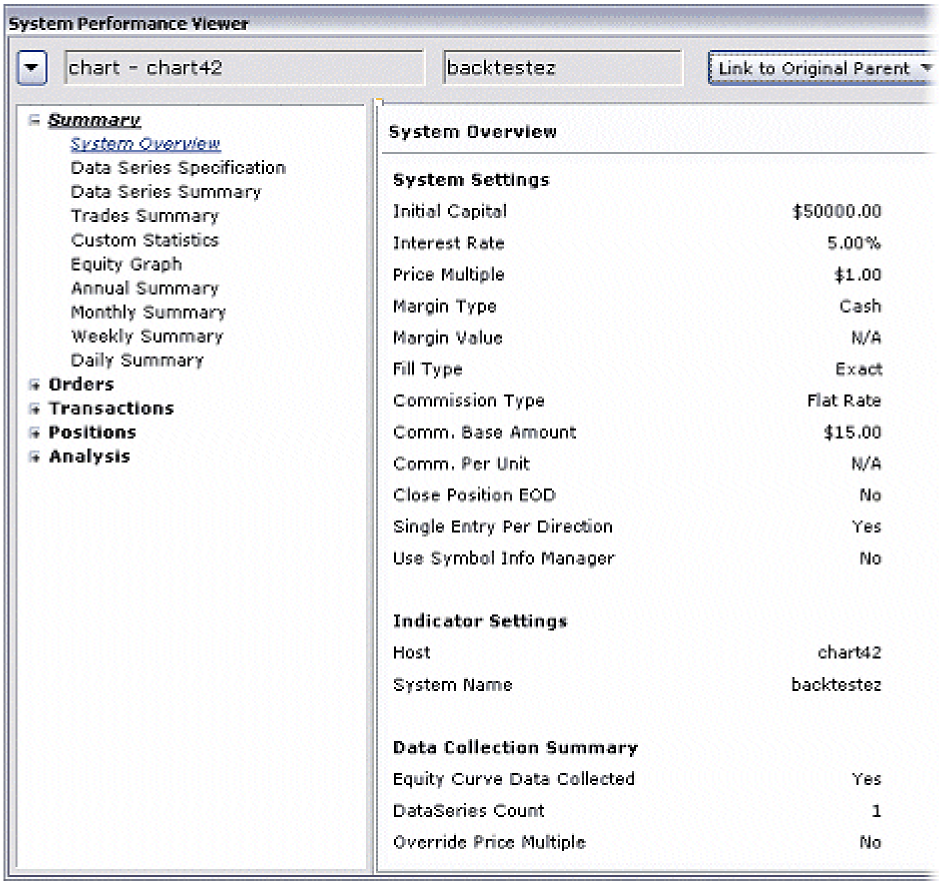
The left side of the viewer is the Report Selection Tree. It provides quick access to all the reports. In the figure, **Summary / System Overview** report is currently active.

The right side of the viewer is the Report Area.

Report Pages

Summary / System Overview

The report **Summary / System Overview** provides a summary of the initial conditions of the trading system together with a summary of the trading system results.

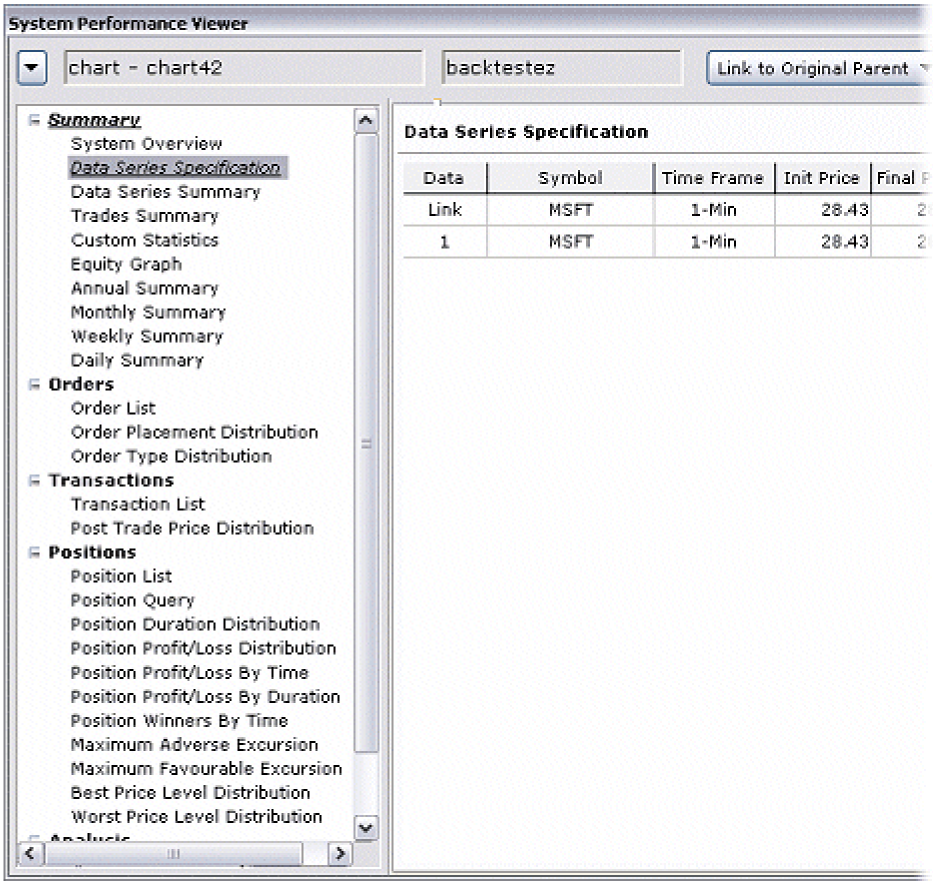


System Overview

System Settings	
Initial Capital	$50000.00
Interest Rate	5.00%
Price Multiple	$1.00
Margin Type	Cash
Margin Value	N/A
Fill Type	Exact
Commission Type	Flat Rate
Comm. Base Amount	$15.00
Comm. Per Unit	N/A
Close Position EOD	No
Single Entry Per Direction	Yes
Use Symbol Info Manager	No
Indicator Settings	
Host	chart42
System Name	backtestez
Data Collection Summary	
Equity Curve Data Collected	Yes
DataSeries Count	1
Override Price Multiple	No

Summary / Data Series Specification

This page reports the summary of all data series that has been traded.



System Performance Viewer
chart - chart42
backtestez
Link to Original Parent
Summary
System Overview
Data Series Specification
Data Series Summary
Trades Summary
Custom Statistics
Equity Graph
Annual Summary
Monthly Summary
Weekly Summary
Daily Summary
Orders
Order List
Order Placement Distribution
Order Type Distribution
Transactions
Transaction List
Post Trade Price Distribution
Positions
Position List
Position Query
Position Duration Distribution
Position Profit/Loss Distribution
Position Profit/Loss By Time
Position Profit/Loss By Duration
Position Winners By Time
Maximum Adverse Excursion
Maximum Favourable Excursion
Best Price Level Distribution
Worst Price Level Distribution

Data Series Specification

Data	Symbol	Time Frame	Init Price	Final P
Link	MSFT	1-Min	28.43	2
1	MSFT	1-Min	28.43	2

For each traded data series, the following information is included:

Data - whether the data series is linked to the system or it is referenced unlinked (e.g. using the `DataSeries` object in a script).

Symbol - symbol of the instrument

Time Frame - 1-tick, 1-minute, etc.

Init Price - Price of the data series when trading start

Final Price - Price of the data series when trading end

Price Multiple - E.g. 1 for stock, other values for contracts

Min Tick Size - E.g. 0.01 for US stocks, other values for other instruments

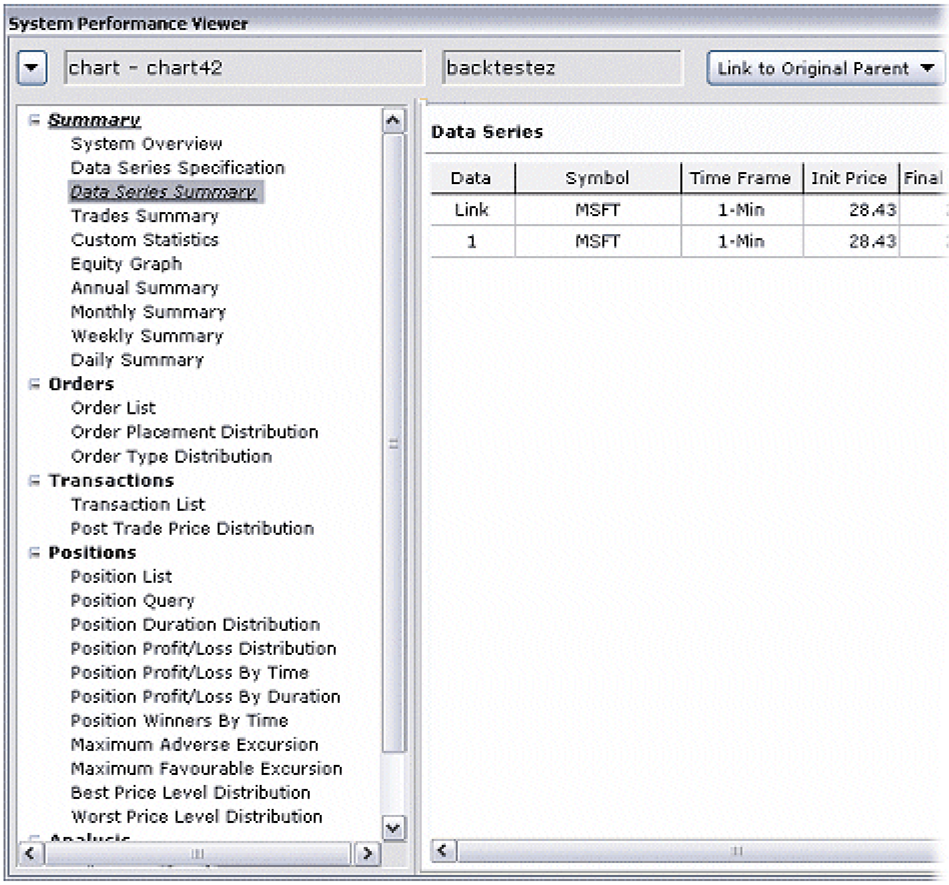
Margin Type - Margin type information

Margin Value - Margin value information

Summary / Data Series Summary

The report **Summary / Data Series** breaks down system performance by symbols, i.e. all linked and non-linked data series in the chart.

This report is primarily for multiple symbol trading systems. If your trading system does not trade multiple symbols, you can ignore this report.


System Performance Viewer
chart - chart42
backtestez
Link to Original Parent
Summary
System Overview
Data Series Specification
Data Series Summary
Trades Summary
Custom Statistics
Equity Graph
Annual Summary
Monthly Summary
Weekly Summary
Daily Summary
Orders
Order List
Order Placement Distribution
Order Type Distribution
Transactions
Transaction List
Post Trade Price Distribution
Positions
Position List
Position Query
Position Duration Distribution
Position Profit/Loss Distribution
Position Profit/Loss By Time
Position Profit/Loss By Duration
Position Winners By Time
Maximum Adverse Excursion
Maximum Favourable Excursion
Best Price Level Distribution
Worst Price Level Distribution
Analysis

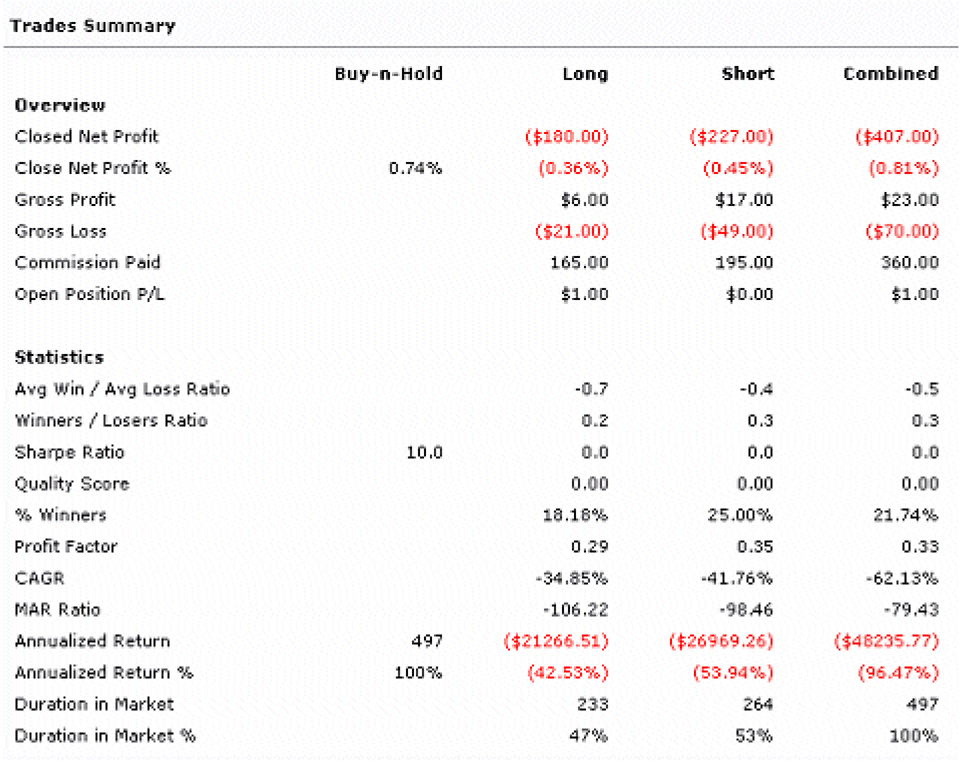
Data Series

Data	Symbol	Time Frame	Init Price	Final
Link	MSFT	1-Min	28.43	
1	MSFT	1-Min	28.43	

Summary / Trades Summary

The report **Summary / Trades Summary** is the most important summary report. It provides key statistics of the trading system. The trading system is analyzed as a whole, and broken down into long only and short only such that you can compare different aspects of the system.

See ***Statistics Reference*** (on page 988).

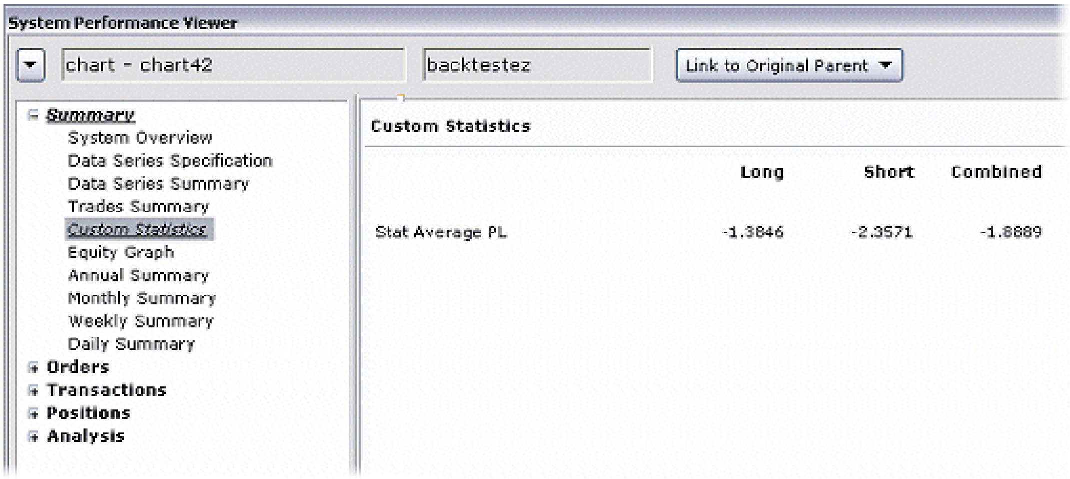


Trades Summary

	Buy-n-Hold	Long	Short	Combined
Overview				
Closed Net Profit		($180.00)	($227.00)	($407.00)
Close Net Profit %	0.74%	(0.36%)	(0.45%)	(0.81%)
Gross Profit		$6.00	$17.00	$23.00
Gross Loss		($21.00)	($49.00)	($70.00)
Commission Paid		165.00	195.00	360.00
Open Position P/L		$1.00	$0.00	$1.00
Statistics				
Avg Win / Avg Loss Ratio		-0.7	-0.4	-0.5
Winners / Losers Ratio		0.2	0.3	0.3
Sharpe Ratio	10.0	0.0	0.0	0.0
Quality Score		0.00	0.00	0.00
% Winners		18.18%	25.00%	21.74%
Profit Factor		0.29	0.35	0.33
CAGR		-34.85%	-41.76%	-62.13%
MAR Ratio		-106.22	-98.46	-79.43
Annualized Return	497	($21266.51)	($26969.26)	($48235.77)
Annualized Return %	100%	(42.53%)	(53.94%)	(96.47%)
Duration in Market		233	264	497
Duration in Market %		47%	53%	100%

Summary / Custom Statistics

This page shows the statistics that are calculated by custom statistics.



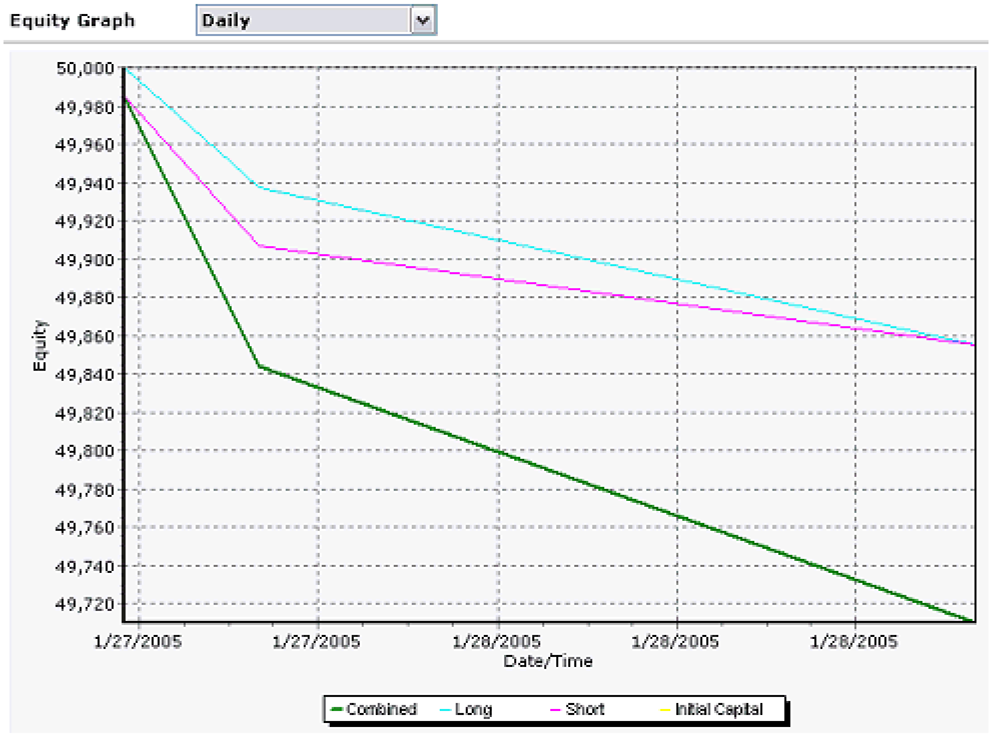
Custom Statistics

	Long	Short	Combined
Stat Average PL	-1.3846	-2.3571	-1.8889

Custom statistics are user defined statistics. For more information how to create custom statistics, see ***Custom Statistics Programming Topics*** (on page 1515).

Summary / Equity Graph

The report **Summary / Equity Graph** provides a visual representation of the equity curve over the tested period of time. You can visually see the relative performance of the complete system, taking the long positions only, taking the short positions only, or holding cash all along (Free Cash). You can also display the margin used.



Equity Graph: Daily

Lines

Press the button to open a pop up menu to turn on/off individual lines in Equity Graph.

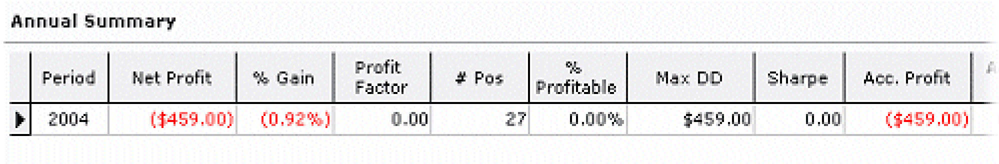
Resolution

You can adjust the resolution of equity graph. By default, equity graph shows daily equity change, you can adjust the resolution to **As Is** (all equity changes shown) to **Yearly**, depending on the trading system you are working on.

Regression

Displays linear regression lines for combined/long/short equity.

Summary / Annual, Monthly, Weekly, Daily Summary



Annual Summary

	Period	Net Profit	% Gain	Profit Factor	# Pos	% Profitable	Max DD	Sharpe	Acc. Profit
▶	2004	($459.00)	(0.92%)	0.00	27	0.00%	$459.00	0.00	($459.00)

Annual Summary

The report **Summary / Annual Summary** provides a table of yearly performance. Both statistics for the specific year and running totals are collected in this table.

Monthly Summary

The report **Summary / Monthly Summary** provides a table of monthly performance. Both statistics for the specific month and running totals are collected in this table.

Weekly Summary

The report **Summary / Weekly Summary** provides a table of monthly performance. Both statistics for the specific week and running totals are collected in this table.

Daily Summary

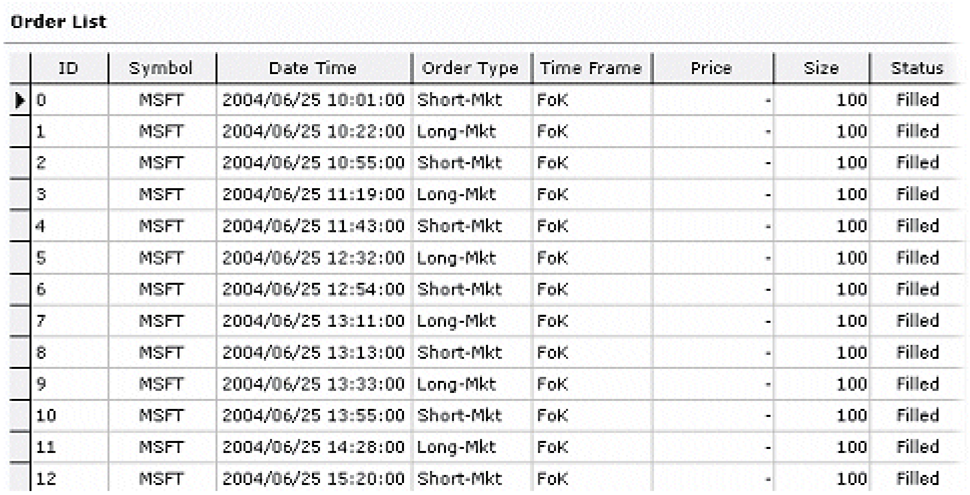
The report **Summary / Daily Summary** provides a table of monthly performance. Both statistics for the specific day and running totals are collected in this table.

Applications

A summary breaks down the performance by time. For example, annual and monthly summary breaks down the performance by year and month.

With a trading system (even an intraday system), you can test the system over a long period of time, then break down the performance by time. This lets you see if the trading system performance is consistent in a basis over time.

Orders / Order List

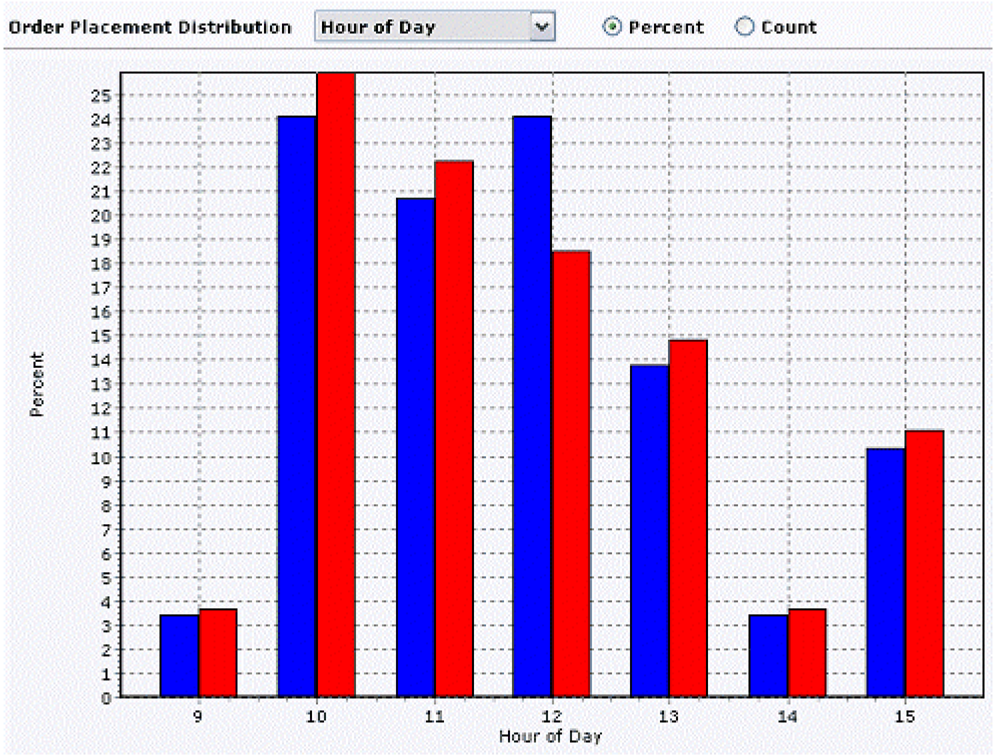


Order List

ID	Symbol	Date Time	Order Type	Time Frame	Price	Size	Status
0	MSFT	2004/06/25 10:01:00	Short-Mkt	FoK	-	100	Filled
1	MSFT	2004/06/25 10:22:00	Long-Mkt	FoK	-	100	Filled
2	MSFT	2004/06/25 10:55:00	Short-Mkt	FoK	-	100	Filled
3	MSFT	2004/06/25 11:19:00	Long-Mkt	FoK	-	100	Filled
4	MSFT	2004/06/25 11:43:00	Short-Mkt	FoK	-	100	Filled
5	MSFT	2004/06/25 12:32:00	Long-Mkt	FoK	-	100	Filled
6	MSFT	2004/06/25 12:54:00	Short-Mkt	FoK	-	100	Filled
7	MSFT	2004/06/25 13:11:00	Long-Mkt	FoK	-	100	Filled
8	MSFT	2004/06/25 13:13:00	Short-Mkt	FoK	-	100	Filled
9	MSFT	2004/06/25 13:33:00	Long-Mkt	FoK	-	100	Filled
10	MSFT	2004/06/25 13:55:00	Short-Mkt	FoK	-	100	Filled
11	MSFT	2004/06/25 14:28:00	Long-Mkt	FoK	-	100	Filled
12	MSFT	2004/06/25 15:20:00	Short-Mkt	FoK	-	100	Filled

The report **Orders / Order List** is a complete listing of all the orders placed into the trading system in chronological order. This report list all orders, including orders that are filled, cancelled, etc.

Orders / Order Placement Distribution



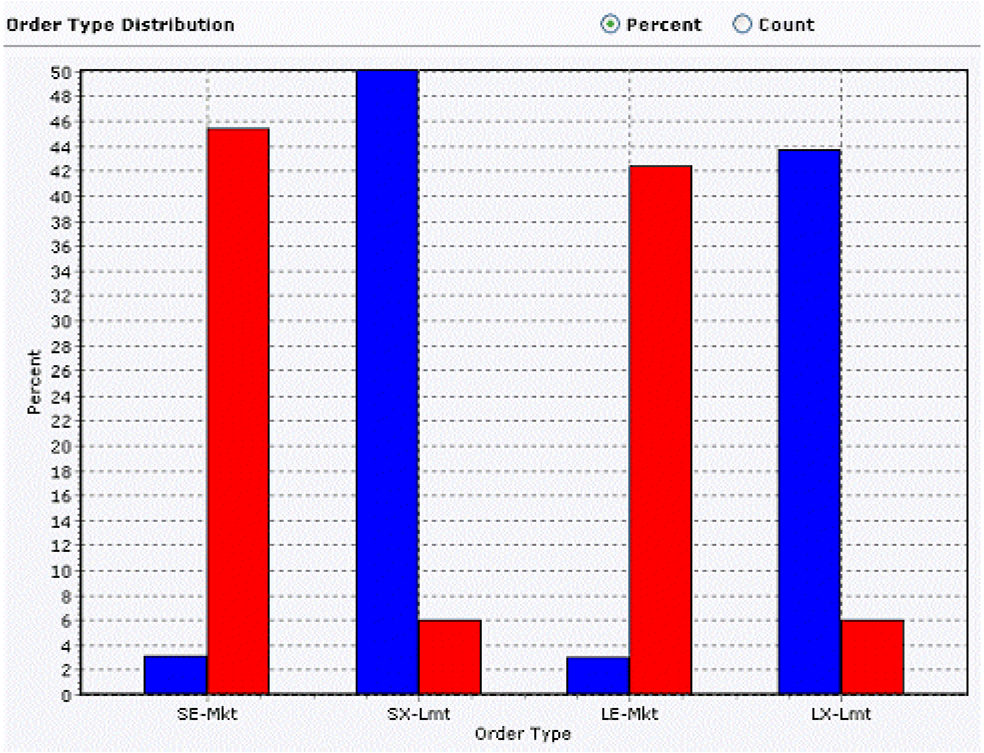
The report **Order / Order Placement Distribution** shows you in either percentage or absolute number of orders that the trading system has placed within a particular time cycle like hour of day, or day of week.

If you choose **Percent**, the distribution is by percentage. If you choose **Count**, the distribution is by the number of orders.

Applications

Coupled with reports such as Position Winner by Time, Order Placement Distribution is an effective tool to weed out money losing trades.

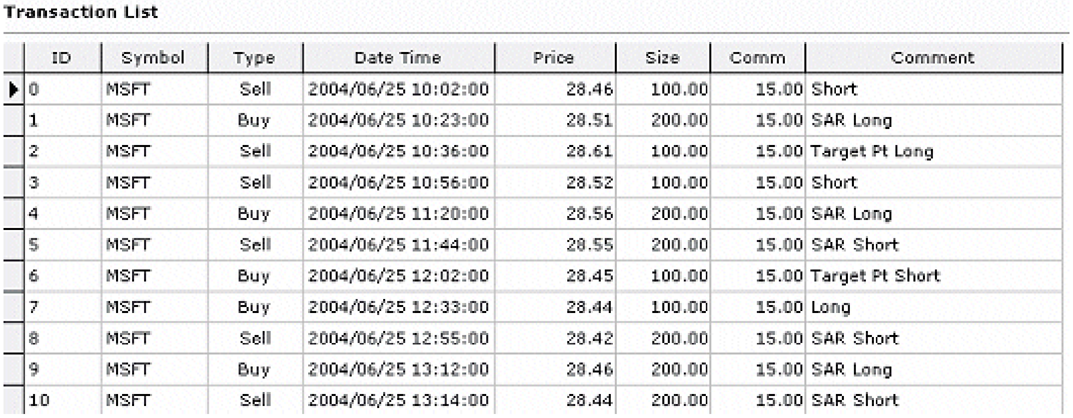
Orders / Order Type Distribution



The report **Order / Order Type Distribution** shows you in either percentage or absolute number of orders that a particular order type is used. It is useful for spotting design errors in a trading system that has an unexpected number of orders of a particular type.

Transactions / Transaction List

The report **Transactions / Transaction List** is a complete listing of all the transactions taken by the trading system sorted in chronological order.



Transaction List

ID	Symbol	Type	Date Time	Price	Size	Comm	Comment
0	MSFT	Sell	2004/06/25 10:02:00	28.46	100.00	15.00	Short
1	MSFT	Buy	2004/06/25 10:23:00	28.51	200.00	15.00	SAR Long
2	MSFT	Sell	2004/06/25 10:36:00	28.61	100.00	15.00	Target Pt Long
3	MSFT	Sell	2004/06/25 10:56:00	28.52	100.00	15.00	Short
4	MSFT	Buy	2004/06/25 11:20:00	28.56	200.00	15.00	SAR Long
5	MSFT	Sell	2004/06/25 11:44:00	28.55	200.00	15.00	SAR Short
6	MSFT	Buy	2004/06/25 12:02:00	28.45	100.00	15.00	Target Pt Short
7	MSFT	Buy	2004/06/25 12:33:00	28.44	100.00	15.00	Long
8	MSFT	Sell	2004/06/25 12:55:00	28.42	200.00	15.00	SAR Short
9	MSFT	Buy	2004/06/25 13:12:00	28.46	200.00	15.00	SAR Long
10	MSFT	Sell	2004/06/25 13:14:00	28.44	200.00	15.00	SAR Short

Transactions / Post Trade Price Distribution

The **Transactions / Post Trade Price Distribution** reports the price distribution after the trades are made. The distribution gives you the general direction where the price is going for trades made by your trading system.

Enabling Post Trade Price Distribution

You need to enable this feature because it is CPU intensive. To enable:

1. Before opening the system performance viewer, edit the trading system indicator in time chart or pattern scanner.
2. Press the **System** tab.
3. Enable the **Trade Analysis** option.
4. Adjust the **Bars to Track** value.
5. Re-apply the indicator.
6. Open system performance viewer.

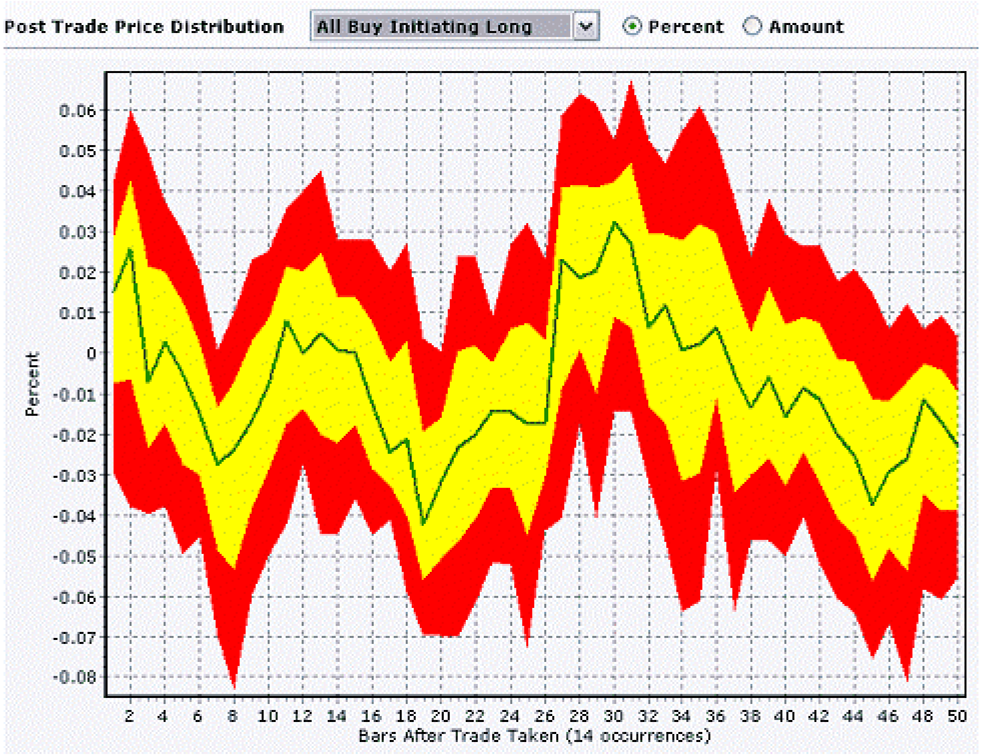
This option uses a lot of CPU time and is not recommended for trading systems being used in real-time. It is recommended you only use this feature for historical analysis.

Bars To Track

When you enable Post Trade Price Distribution by enabling the **Trade Analysis** option, there is **Bars to Track** value you can adjust.

This parameter controls at each bar that a trade happens, the sample space of the distribution. The sampling space will be the price from the current bar, to **Bars to Track** number of bars into the future.

How to Read



Consider the Post Trade Distribution Chart above.

Note the **All Buy Initiated Long** is selected. The chart thus reports the distribution for the average of all buy trades that are initiating a long position, i.e. it does not include buy trades that cover short positions.

At the horizontal axis, the chart reports that there are 14 occurrences. Thus the chart is a summary of 14 buy trades.

The chart considers the trades happen at 0 bar. The horizontal axis represents how many bars away from the initial trade.

The vertical axis reports the deviation of the price from the trade's entry price. The deviation is reported in percentage, or amount.

The green line is the closing price deviation. The red region is range of high price deviation and low price deviation. The yellow region is 50% between close and high, or between close and low.

Back to the chart above.

Look at the chart near bar 19, the green line is at -0.04%. This tells us on average, 19 bars after a trade, the price would drop by -0.04%. Since these are buy trades, the trades are losing money 19 bars after they are filled.

Near bar 30, the green line is at 0.03%. This tells us on average, 30 bars after a trade, the price would be up by 0.03%. Since these are buy trades, the trades are making money 30 bars after they are filled.

Now consider the red region. Near bar 7, the top of red region touches 0. This tells us on average, if you manage to sell at the best price after 7 bars, you will break even.

Overall, the chart tell us that on average, there is little chance this system makes money on the long side. The green line is going down in general after bar 30, the longer the system holds on the trade, the more money it loses.

The Post Trade Distribution Chart indicates that the long side of the trading system needs some work.

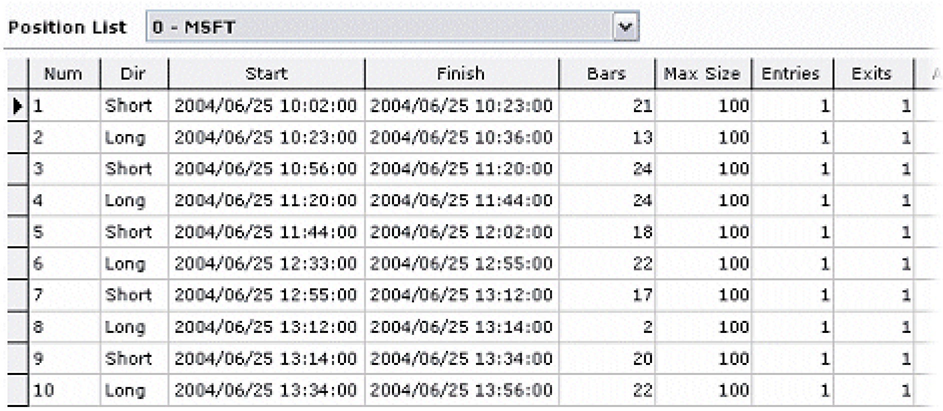
Transactions Tracked

Basic types of transactions are tracked:

- **All Buy** type is the distribution of all the trades that buy.
- **All Sell** type is the distribution of all the trades that sell.
- **All Buy Initiating Long type** is the distribution of all the trades that establish a long position.
- **All Sell Initiating Short type** is the distribution of all the trades that establish a short position.

Additional types are controlled by the comments of the order. Trades that are taken with non-empty comments are organized according to the comments and can be filtered for price distribution analysis similar to the 4 basic types.

Positions / Position List



Position List: 0 - MSFT

Num	Dir	Start	Finish	Bars	Max Size	Entries	Exits
1	Short	2004/06/25 10:02:00	2004/06/25 10:23:00	21	100	1	1
2	Long	2004/06/25 10:23:00	2004/06/25 10:36:00	13	100	1	1
3	Short	2004/06/25 10:56:00	2004/06/25 11:20:00	24	100	1	1
4	Long	2004/06/25 11:20:00	2004/06/25 11:44:00	24	100	1	1
5	Short	2004/06/25 11:44:00	2004/06/25 12:02:00	18	100	1	1
6	Long	2004/06/25 12:33:00	2004/06/25 12:55:00	22	100	1	1
7	Short	2004/06/25 12:55:00	2004/06/25 13:12:00	17	100	1	1
8	Long	2004/06/25 13:12:00	2004/06/25 13:14:00	2	100	1	1
9	Short	2004/06/25 13:14:00	2004/06/25 13:34:00	20	100	1	1
10	Long	2004/06/25 13:34:00	2004/06/25 13:56:00	22	100	1	1

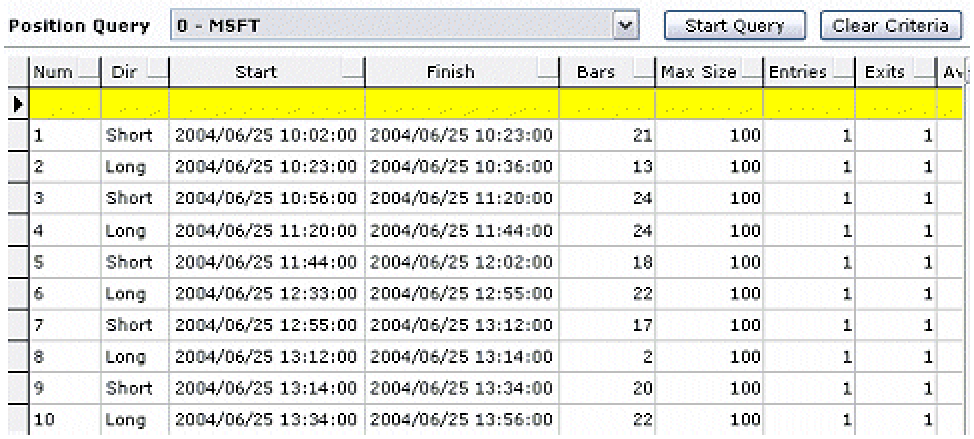
The **Positions / Position List** is the complete listing of all the positions being taken by the trading system. The positions are organized by the symbols they are taken for. To view the positions of a particular symbol, simply choose the symbol from the list at the top of the report area.

The position list table has smart jump capability. When you double-click on a position line, the linked window will scroll to the point where the trades took place. It will even zoom in to cover the whole period of time from the first entry to the last exit of the position.

If you double-click on the **Start** or **Finish** column on any position entry, then the linked window will only scroll to the date time indicated in the cell and will not change the zoom level.

Positions / Position Query

The **Positions / Position Query** report is a special reporting tool that allows you to filter the positions by your custom criteria.



Position Query: 0 - MSFT

Num	Dir	Start	Finish	Bars	Max Size	Entries	Exits	Av
1	Short	2004/06/25 10:02:00	2004/06/25 10:23:00	21	100	1	1	
2	Long	2004/06/25 10:23:00	2004/06/25 10:36:00	13	100	1	1	
3	Short	2004/06/25 10:56:00	2004/06/25 11:20:00	24	100	1	1	
4	Long	2004/06/25 11:20:00	2004/06/25 11:44:00	24	100	1	1	
5	Short	2004/06/25 11:44:00	2004/06/25 12:02:00	18	100	1	1	
6	Long	2004/06/25 12:33:00	2004/06/25 12:55:00	22	100	1	1	
7	Short	2004/06/25 12:55:00	2004/06/25 13:12:00	17	100	1	1	
8	Long	2004/06/25 13:12:00	2004/06/25 13:14:00	2	100	1	1	
9	Short	2004/06/25 13:14:00	2004/06/25 13:34:00	20	100	1	1	
10	Long	2004/06/25 13:34:00	2004/06/25 13:56:00	22	100	1	1	

To make a query:

1. Select the symbol at **Position Query**.
2. Type your criteria into the query entry area (yellow background color).
3. Press the **Start Query** button, the viewer will locate all the positions that match the specified criteria and display them in the table.

To specify a condition in the query entry area based on a particular field or column, you have the following choices,

- Leave it blank - the field will be ignored
- Enter a value - the exact value must match before a position will be displayed.
- Enter a comparison rule - you can enter one of the comparison operators (<, >, <=, >=, <>, =) followed by a value to compare against, then only those positions satisfying the criteria will be displayed.

All the conditions must be matched for a position to be included in the query result.

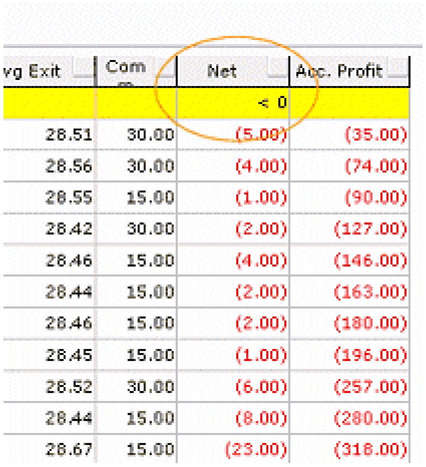
If you want to start over in specifying the criteria, you can press the **Clear Criteria** button to clear all the conditions you have entered into the query entry area.

The position query table also has smart jump capability.

Operator Less Than <

Use this operator to locate position with values less than the one specified.

E.g. < 0 in the **Net** column means you want to see only the losing positions in the query result.



vg Exit	Com m	Net	Acc. Profit
		< 0	
28.51	30.00	(5.00)	(35.00)
28.56	30.00	(4.00)	(74.00)
28.55	15.00	(1.00)	(90.00)
28.42	30.00	(2.00)	(127.00)
28.46	15.00	(4.00)	(146.00)
28.44	15.00	(2.00)	(163.00)
28.46	15.00	(2.00)	(180.00)
28.45	15.00	(1.00)	(196.00)
28.52	30.00	(6.00)	(257.00)
28.44	15.00	(8.00)	(280.00)
28.67	15.00	(23.00)	(318.00)

Operator Greater Than >

Use this operator to locate position with values greater than the one specified.

E.g. > 0 in the Net column means you want only winning positions in the query result.

Operator Less Than Or Equal To <=

Use this operator to locate position with values less than or equal to the one specified.

E.g. <= 10/1/2002 in the Finish column means you want only positions completed on or before Oct 1, 2002 in the query result.

Operator Greater Than Or Equal To >=

Use this operator to locate position with values greater than or equal to the one specified.

E.g. >= 10/1/2002 in the Start column means you want only positions established on or after Oct 1, 2002 in the query result.

Operator Not Equal To <>

Use this operator to locate position than does not match the value specified.

E.g. <> Long in the Dir column means you do not want the Long positions in your query result.

Operator Equal To =

Use this operator to locate positions than match the value you have provided. This is the default operator. If you do not specific an operator and simply enter a value to the query entry area. It is assumed that you want to do an exact match.

E.g. = Long in the Dir column means you only want the Long positions returned in your query result.

Operator Range ...

Use this operator to locate positions that has a field value in the specified range.

E.g. 1 ... 10

Operator Regular Expression ~

Use this operator for regular expression searching. Regular expression provides a powerful way to specify non-exact search phrases. It is especially handy for text searching, i.e. searching the entry/exit comments. For example, you can use the following expression:

~Short

in the Entry Comment column to search for any position that produces a comment containing the word "Short", e.g. SAR Short, Short, My Short, etc.

Regular expression searching is case insensitive.

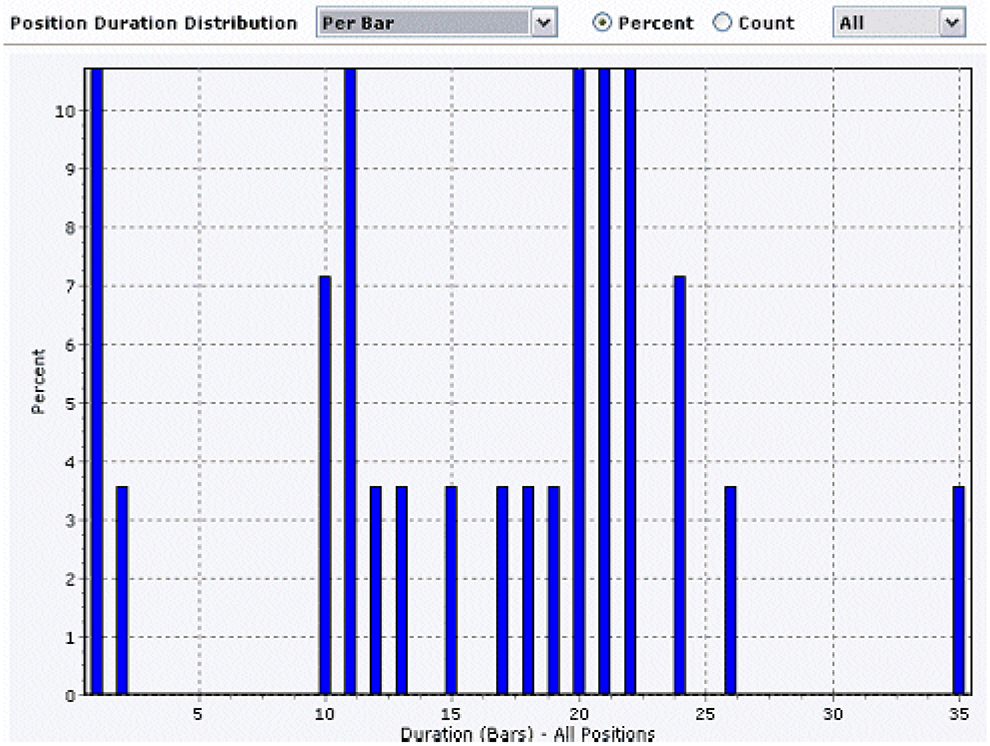
The following table lists the wild card characters of regular expression.

Constant	Explanation	Example
*	Match zero or more characters	~My*Short will match the following phrases "My Short", "My Large Short", "XYZ My Short"
^	Match the beginning of line	~^Short will match the following phrases "Short", "Shorting", but not "My Short"
$	Match the end of line	~Short$ will match the following phrases "Short", "My Short" but not "Shorting"
\	Allow the use of wild cat character in the search	~\$ will match any phrase containing the dollar sign ($).
-	Used in square bracket	~Short[a-z] will match phrases "Shorta", "Shortb", "Shortc".
^	Negate next character	~Short^[0-4] will match phrases "Short5", "Short6", "Shorta" but not "Short0", "Short1".
[	Opening bracket for matching a range of characters.	~Short[012] will match phrases "Short0", "Short1", "Short2"
]	Closing bracket for matching a range of characters	See above.
.	Match any single character	~Short. will match phrases "Short0", "Shorta", "Shortb" but not "Short"

Positions / Position Duration Distribution

The **Positions / Position Duration Distribution** chart gives you an overview of how long each position stays in the market. If you find the distribution of per bar basis is too condensed, you can change the duration interval to per 10 bars, per 100 bars, etc. to have a better-summarized picture.

There are options to let you changing the representation from percent to absolute count of occurrence at each duration slot. You can also choose to view the distribution for long or short positions only by selecting the item from the list at the top of the report area.

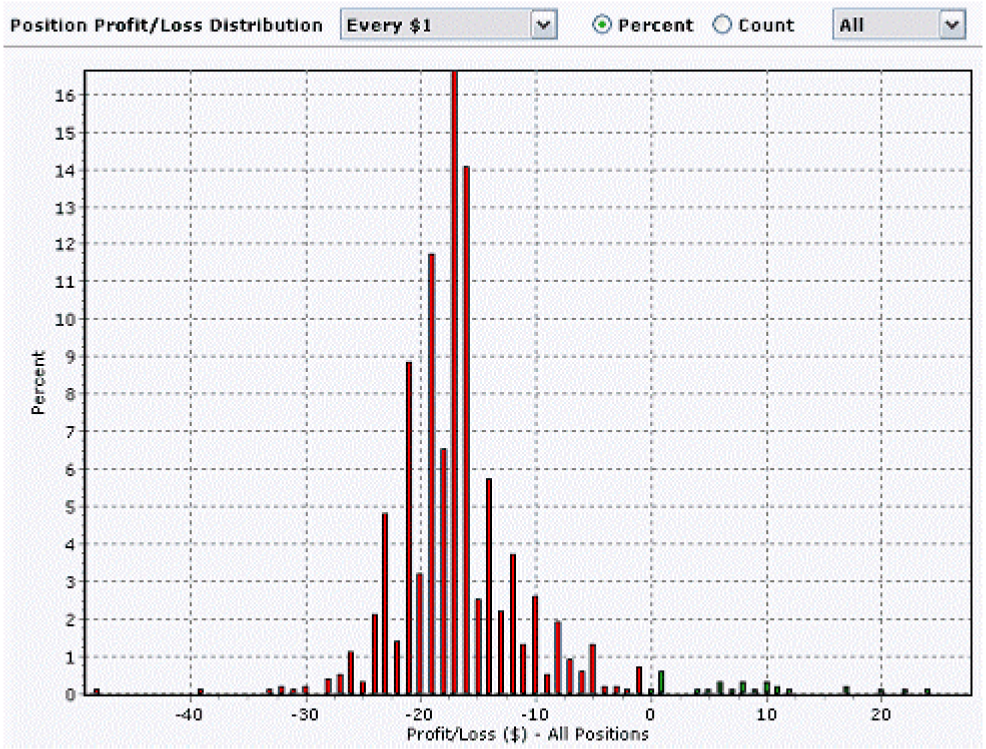

Position Duration Distribution
Per Bar
● Percent
○ Count
All
Percent
0
1
2
3
4
5
6
7
8
9
10
5
10
15
20
25
30
35
Duration (Bars) - All Positions

The chart above is to **Per Bars** in percentage. It shows that majority of trade stays for 10 to 25 bars in the trading system.

Positions / Position Profit/Loss Distribution

The **Positions / Position Profit/Loss Distribution** chart allows you to examine the percentage or number of positions having similar profit/loss values. You can choose to partition the profit/loss values on a per $1 basis, per $10 basis, etc. You can also choose to view the distribution of long or short only positions.

This chart is useful for identifying if absolute amount stop loss or target is useful for the trading system.



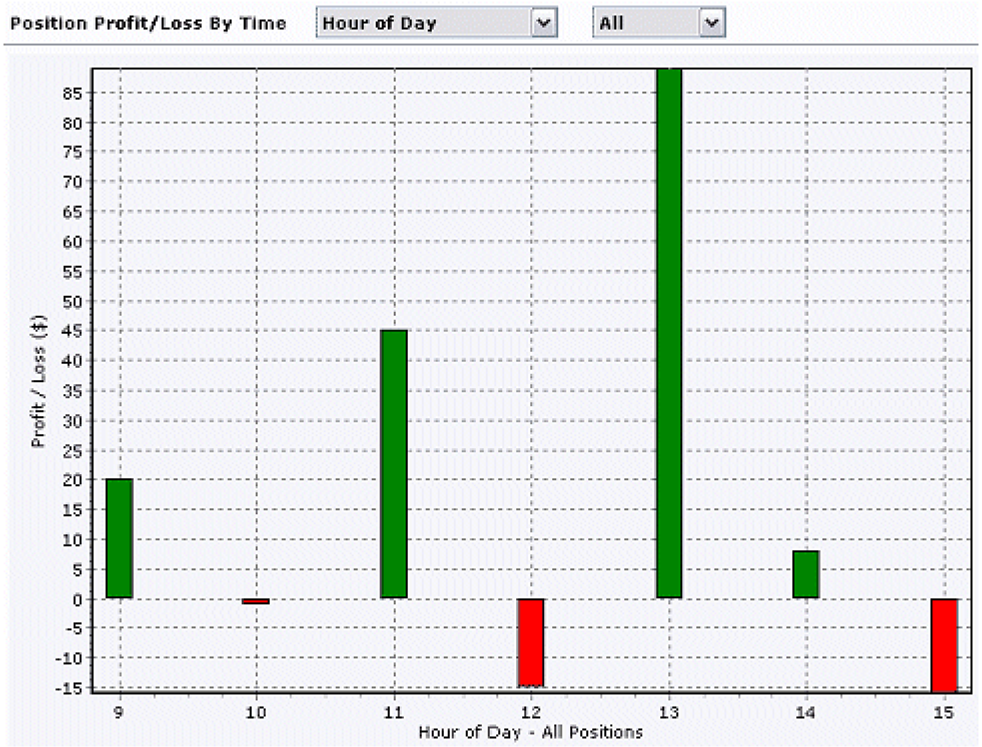
Position Profit/Loss Distribution Every $1 ● Percent ○ Count All

The chart above shows that majority of the trades in the trading system lost around $16 per trade.

Positions / Position Profit/Loss By Time

The **Positions / Position Profit/Loss By Time** chart allows you to examine the profit/loss of the trading system based on a particular time cycle like hour of day, day of week, month of year, etc. The statistics is based on entry time. You can also choose to display distribution on long or short positions only.

This chart is useful for locating bad trading time period so that time filters can be added to the trading system to improve its performance. For example, if the chart clearly shows that on an hour of day basis, the trading system loses significant amount of money say at the market open hour, than you can add a time filter to block the system from taking positions at that time to avoid the bad signals.

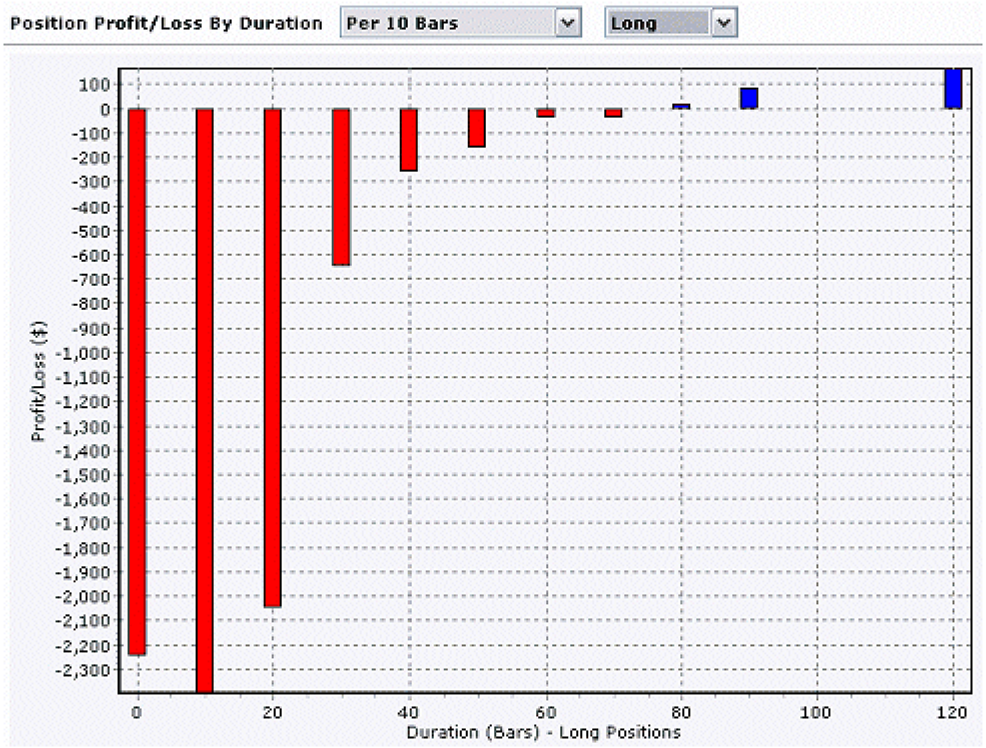


In the chart above, it shows that trading system tend to make money if the entry time is around 1:00pm.

Positions / Position Profit/Loss By Duration

The **Positions / Position Profit/Loss By Duration** chart allows you to examine the profit/loss of the trading system based on the duration of the position, in bars. The bars can be single, 10-bars, etc. You can also choose to display distribution on long or short positions only.

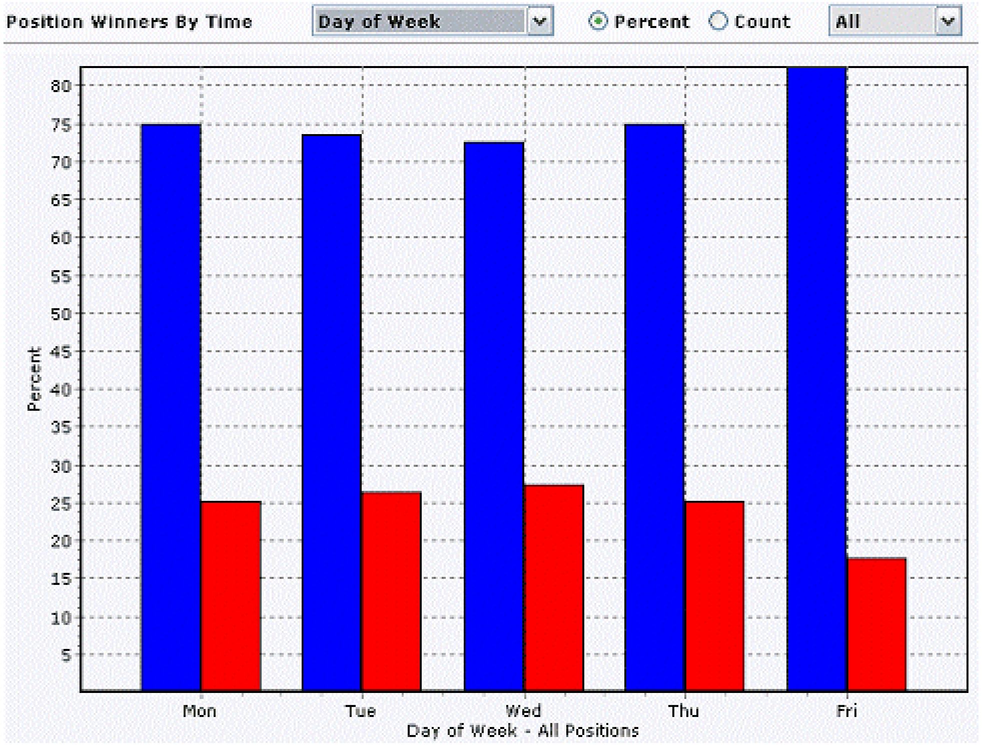
This chart is useful for improving entry/exit point. For example, the chart below shows that positions that last 10 bars or less lose money most. This suggests you can improve the system performance by adding confirmations signals before entering a position.


Position Profit/Loss By Duration Per 10 Bars Long

Positions / Position Winners by Time

The **Position / Position Winners By Time** chart provides the comparison between the winning and losing positions based on a particular time cycle.

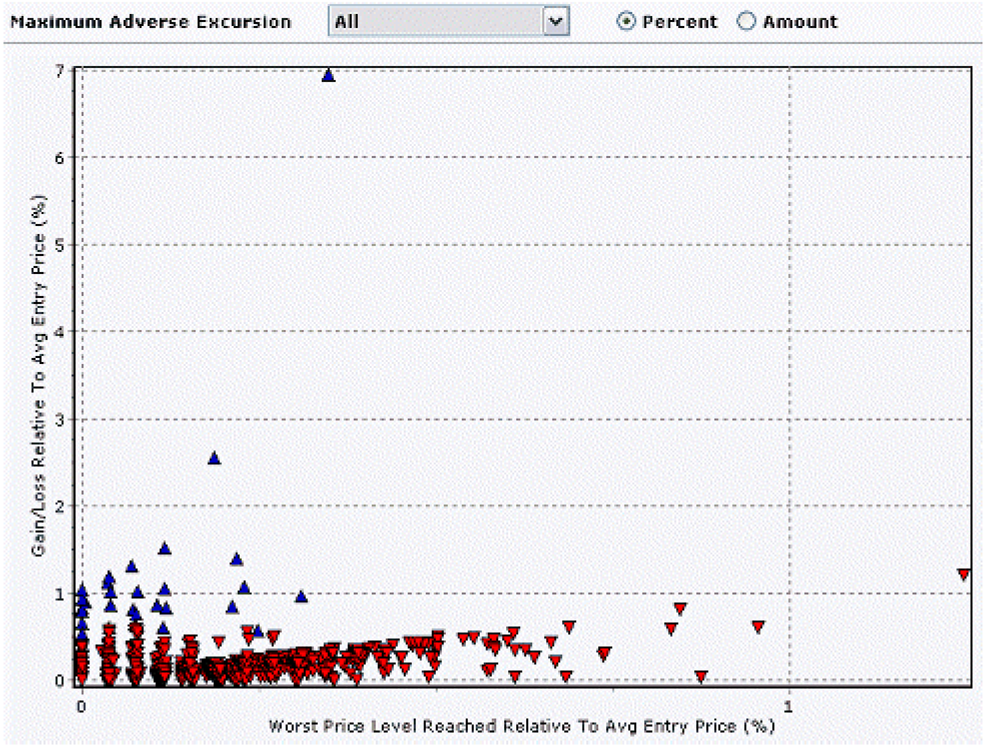
This chart is useful for identifying weaknesses of the trading system at particular time period so that improvements can be made. For example, if a trading system has a lower than average percentage winners at the end of each month (using the Day of Month option) than you can either filter out those signals or add special money management methods during the end of month to protect those trades.



In the chart above, the system performs the best on Friday.

Positions / Maximum Adverse Excursion

The **Position / Maximum Adverse Excursion** chart shows you the relative distribution of the completed positions based on the worst price level they have reached vs. their final profit/loss result. You can choose to display the chart in percentage or in absolute amount from the entry price.


Maximum Adverse Excursion
All
● Percent
○ Amount
Gain/Loss Relative To Avg Entry Price (%)
7
6
5
4
3
2
1
0
0
1
Worst Price Level Reached Relative To Avg Entry Price (%)

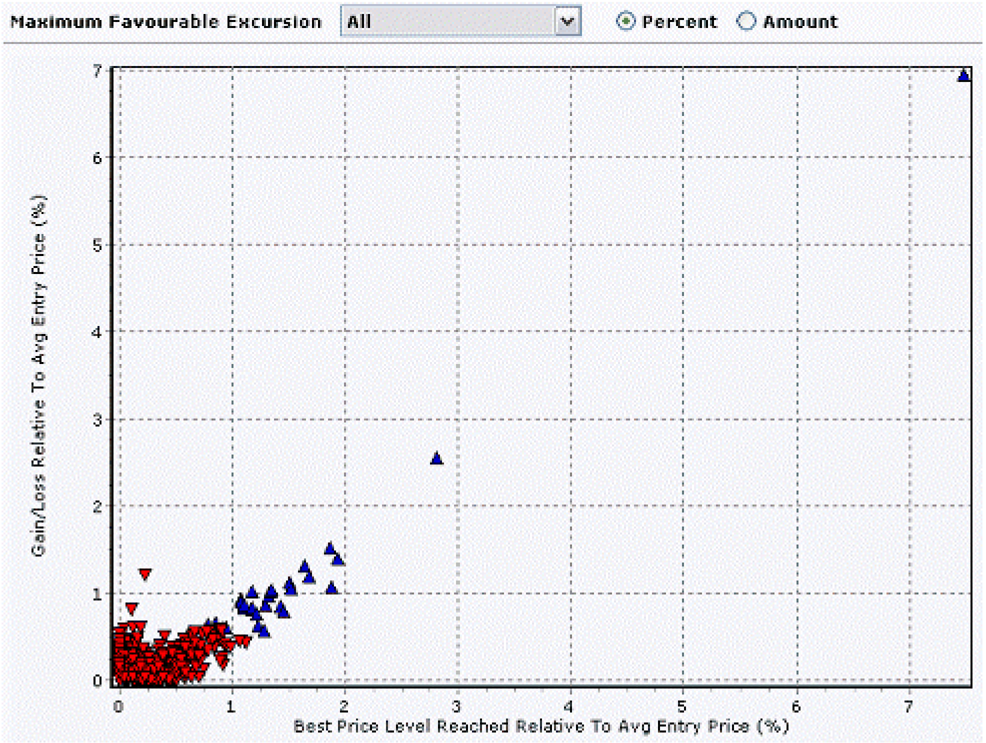
The blue markers are profitable positions while the red markers are the ones that are not profitable.

The MAE chart can help you to determine if applying stop loss to your trading system will help improve the performance results. Since you can choose from either percent or absolute amount style for the chart. You can use the one with better consistency to add the appropriate money management rules to your system.

In Backtest EZ, you can add stop loss either based on a fixed percentage from the entry price or, a fixed amount of Points from the entry price. Using MAE, you will be able to find the best type of stop loss to be used with your trading system.

Positions / Maximum Favourable Excursion

The Positions / Maximum Favourable Excursion chart shows you the relative distribution of the completed positions based on the best price level they have reached vs. their final profit/loss result. You can choose to display the chart in percentage or in absolute amount from the entry price.



Maximum Favourable Excursion All ● Percent ○ Amount

Gain/Loss Relative To Avg Entry Price (%)
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
Best Price Level Reached Relative To Avg Entry Price (%)

The blue markers are profitable positions while the red markers are the ones that are not profitable.

The MFE chart can help you to determine if applying profit target to your trading system will help improve the performance results. Since you can choose from either percent or absolute amount style for the chart. You can use the one with better consistency to add the appropriate money management rules to your system.

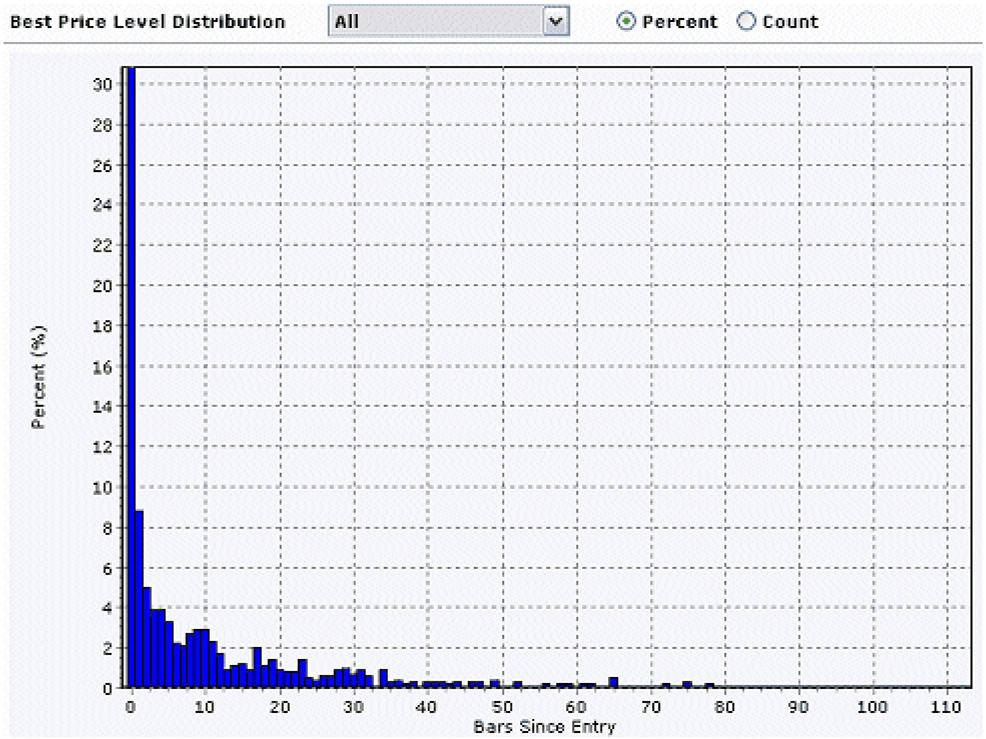
In Backtest EZ, you can add profit target either based on a fixed percentage from the entry price or, a fixed amount of Points from the entry price. Using MFE, you will be able to find the best type of profit target to be used with your trading system.

Positions / Best and Worst Price Level Distribution

Best Price Level Distribution displays the statistics about how many bars it takes for the positions to reach their best price.

Worst Price Level Distribution displays the statistics about how many bars it takes for the positions to reach their worse price.

Consider the following Best Price Level Distribution:



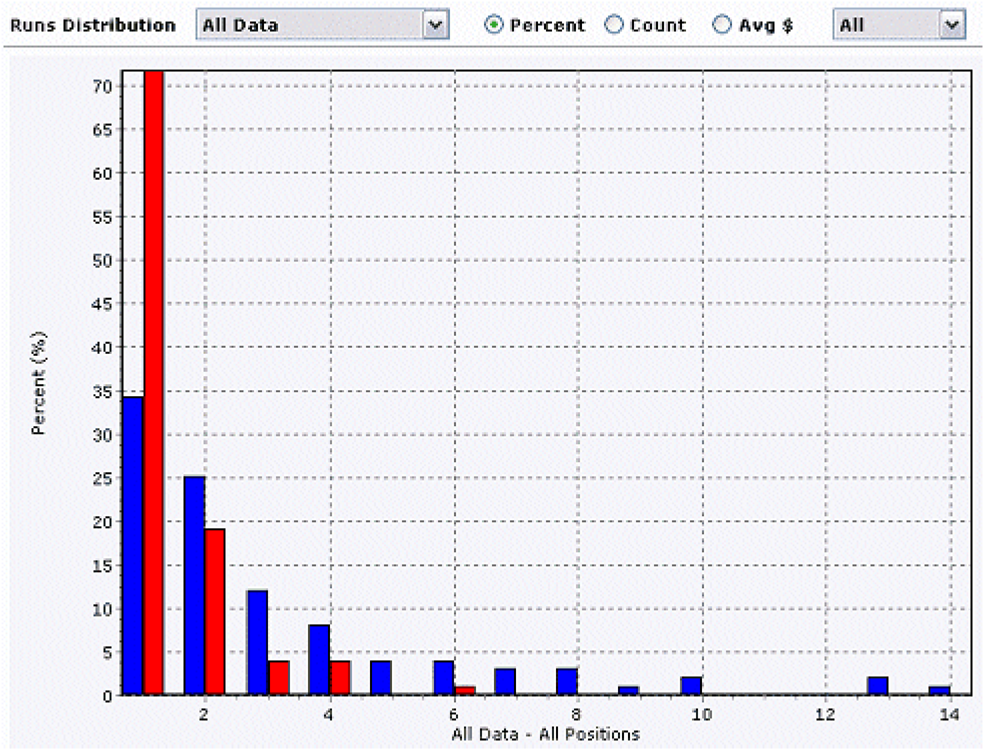
From the chart, we can see that many positions (about 30%) reach their best price level at around 0 bars since entry. This indicates that the system is not doing a good job at entering positions. For these positions, the price never goes above entry price.

You can use Best and Worst Price Level Distributions to help you fine tune when to exit trading systems.

Analysis / Runs Distribution

Runs Distribution collects statistics of runs of positions that are all winning or all losing. For example, if three consecutive positions are all winning, it has a winning run of 3. If four consecutive positions are all losing, it has a losing run of 4.

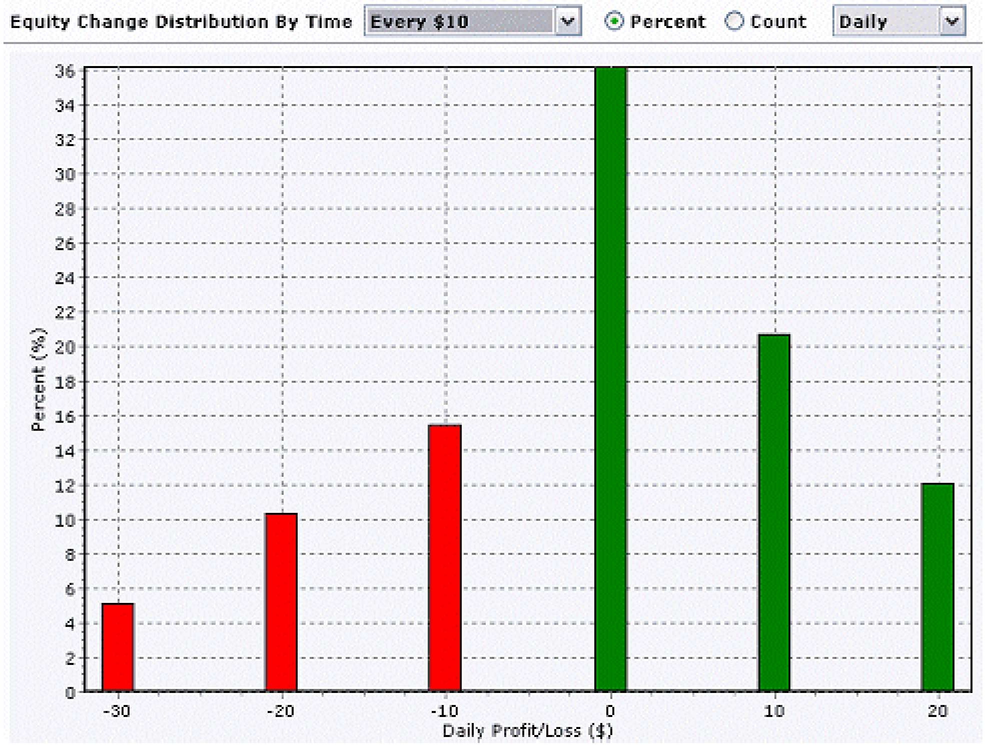
Consider the following figure.



The blue bars are the statistics for winning runs. The red bars are statistics of losing runs. For example, the figure tells us that 25% of the winning runs are winning runs of 2.

Analysis / Equity Change Distribution by Time

Equity Change Distribution by Time analysis partitions equity change by time. This is best illustrated by example. Consider the following figure.

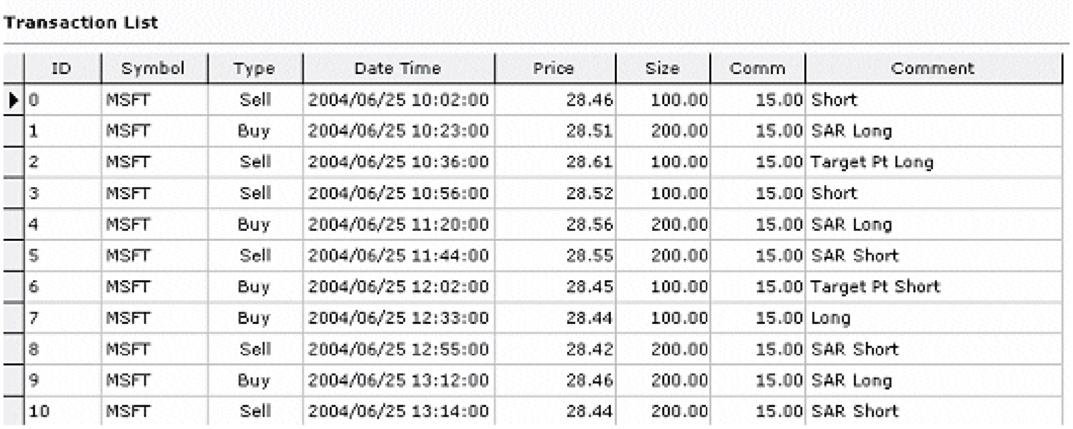

Equity Change Distribution By Time Every $10 ● Percent ○ Count Daily

The figure is daily Equity Change Distribution by using $10 slot size. The red bar at -10 indicates that about 15% of the time, the system has a daily loss of $10 or less. The green bar at 0 indicates that 36% of the time, the system has a daily win of $0 to $9. The green bar at 10 indicates that about 21% of the time, the system has a daily win of $10 to $19.

Locating Orders, Transactions, Positions

You can locate the time when an order, trade, or position happens.

1 Open the report of the item. For example, the transaction list:



Transaction List

ID	Symbol	Type	Date Time	Price	Size	Comm	Comment
0	MSFT	Sell	2004/06/25 10:02:00	28.46	100.00	15.00	Short
1	MSFT	Buy	2004/06/25 10:23:00	28.51	200.00	15.00	SAR Long
2	MSFT	Sell	2004/06/25 10:36:00	28.61	100.00	15.00	Target Pt Long
3	MSFT	Sell	2004/06/25 10:56:00	28.52	100.00	15.00	Short
4	MSFT	Buy	2004/06/25 11:20:00	28.56	200.00	15.00	SAR Long
5	MSFT	Sell	2004/06/25 11:44:00	28.55	200.00	15.00	SAR Short
6	MSFT	Buy	2004/06/25 12:02:00	28.45	100.00	15.00	Target Pt Short
7	MSFT	Buy	2004/06/25 12:33:00	28.44	100.00	15.00	Long
8	MSFT	Sell	2004/06/25 12:55:00	28.42	200.00	15.00	SAR Short
9	MSFT	Buy	2004/06/25 13:12:00	28.46	200.00	15.00	SAR Long
10	MSFT	Sell	2004/06/25 13:14:00	28.44	200.00	15.00	SAR Short

2 Double click on an item. For example, item 22.

3 By default, the chart that creates the trading performance will pop to the top and scroll to the position when the item happens.

The chart must be present in the current active group.

Changing the Link To Designation

The **Link To** option at the top of the viewer window has 2 choices. You can choose either the **Original Parent** or the **Active Window**. To change the setting, simply left click on the **Link To** box and a pop up menu with the choices will show up. Make your selection by clicking on the type of window you wish the viewer to link with.

When this setting is **Original Parent**, double-clicking will instruct the viewer to talk to the original parent window to scroll to the date time position specified by the position list entry.

When the setting is **Active Window**, the target window will be changed from the original parent window to the last active function window. Setting to **Active Window** can be used to look at what happens at other charts when a particular item happens.

Hiding and Showing Report Selection Tree

To hide the Report Selection Tree Area, you can press the **Hide Selection** button at the top of the viewer. The Report Selection Tree Area will be hidden while the button caption will change to **Show Selection**.

To show the Report Selection Tree Area again, you can press the **Show Selection** button. The Report Selection Tree Area will show up again while the button caption will change back to **Hide Selection**.

Sending a Table from Report Page to Excel

If the current report page is a table (e.g. orders, transactions, positions), you can press the **To Excel** button to send the table to Microsoft Excel.

You must have Microsoft Excel installed on your computer for this feature to work.

Printing a Report Page

To print the active report page, press the **Print Page** button.

You can use choose from the main window **Program>Page Setup** to change the default page setup options.

To change the target printer or printer specific options, you can choose from the main window **Program>Printer Setup**.

System Performance Viewer and Real-life Broker Connection

System Performance Viewer can be used to analyze trades placed to a real-life broker.

The prerequisite are:

- The broker is connected to NeoTicker® through an ActiveX interface in Order Interface Manager
- The broker can return order filling information to the ActiveX interface

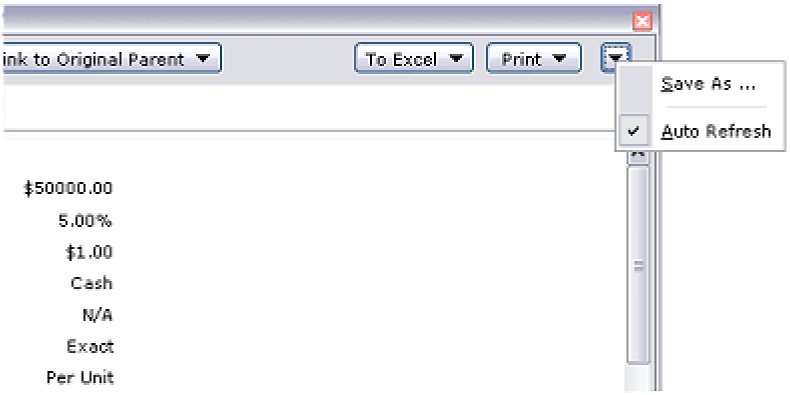
Connection to Broker

Connection to broker is handled by Order Interface Manager. See ***Connecting to a Real-life Broker*** (on page 736) in the Order Interface Manager usage guide for more information.

Auto Refresh

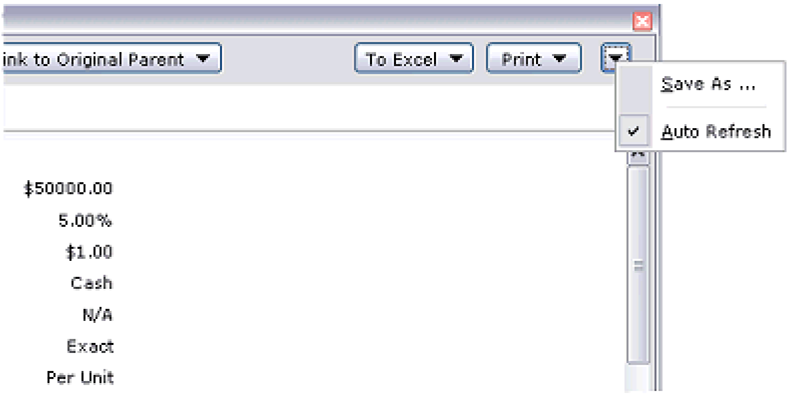
By default, System Performance Viewer will automatically refresh its statistics when the underlying trading system changes. For example, if you adjust a trading system parameter, once after the change takes effect, the related System Performance Viewer will also be updated.

You can turn off this behavior by pressing the menu button at the upper right corner of System Performance Viewer and turn off **Auto Refresh**.



Saving

You can save a System Performance Viewer under another name by click on the menu button on the upper right corner of System Performance Viewer and choose **Save As**. You can adjust trading system parameters, save the performance to different files, and compare the result later on.



Statistics Reference

Most statistics are reported for long, short and combined positions.

Account Size Required

Account size ($ amount) required to trade the system.

Annualized Return

Yearly profit/loss ($ amount).

Annualized Return %

Convert the profit/loss to annual return in percentage of initial capital.

Average Duration In Bars

Average amount of time (in bars) a position stayed in the market.

Average Loser

Average ($ amount) of losing positions.

Average P/L

Average ($ amount) profit/loss of all positions.

Average Winner

Average ($ amount) of winning positions.

Avg Win / Avg Loss Ratio

Average winner to average loser ratio.

Best Winner

Amount of profit ($ amount) of the best winning position.

CAGR

Compound Annual Growth Rate.

Closed Net Profit

Net profit assuming open position are closed at latest price.

Closed Net Profit %

Closed net profit in percentage of initial capital.

Commission

Total amount of paid commission.

Duration in Market

Amount of time (in bars) the system stayed in the market.

Duration in Market %

Duration in market as a percentage of total number of bars.

Gross Loss

Gross loss (loss before commission).

GrossProfit

Gross profit (profit before commission).

Loser Average Duration

For losing positions, the average duration in market in bars.

# of Losers

Number of losing positions.

MARRatio

MAR Ratio.

Max Consec Losers per Symbol

Maximum number of consecutive losing positions.

Max Consec Winners

Maximum number of consecutive winning positions.

Max DD in Single Position

Maximum drawn down ($ amount) in a single position.

Max DD in Equity

Maximum drawn down ($ amount) in equity (initial capital)

Max DD % in Equity

Maximum drawn down as a percentage of equity (initial capital).

Max Position Size

Maximum number of contracts/shares hold in a single position.

# of Non Scratch

Number of non scratch positions. Scratch positions are positions that has a win/loss that is lower than commission.

Open Position P/L

Profit loss ($ amount) of open positions.

# of Positions Taken

Number of positions the system has taken.

Profit Factor

Average winner to average loser ratio.

Quality Score

To be written.

Sharpe Ratio

Sharpe Ratio.

Winner Average Duration

For winning positions, the average duration in market in bars.

% Winners

Percentage of winners of all positions.

# of Winners

Number of winning positions.

Winners / Losers Ratio

Winning positions to losing positions ratio.

Worst Loser

Amount of loss of the worst losing position.

Tick Filter Manager

Tick filters described here are for time charts. There is a separate basic level tick filter that automatically filter many bad ticks at the data feed level. Use the tick filters described here only if you deem the basic tick filter is not sufficient for your need.

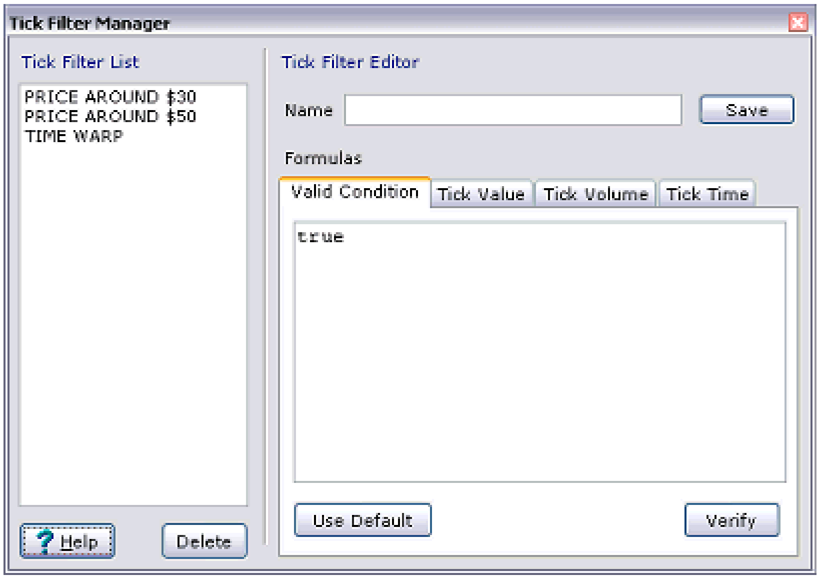
Real-time tick-by-tick update to your charts can contain errors and misprints from either the data vendor or the exchanges. Without tick filtering, one bad tick could mess up your indicator calculation and render your trading signals useless. By using a tick filter that is designed to filter errors from your data, you can better control your data display and indicator calculations.

Tick Filter Manager enables you to define or customize your own version of tick filters with formulas to filter real-time incoming tick data.

Filters can also be designed to transform tick data into customized data series.

Access the Tick Filter Manager

To open the Tick Filter Manager, choose from the main menu, **Manager>Tick Filter.**



PRICE AROUND $30
PRICE AROUND $50
TIME WARP

```
true
```

Left side of the window is a list of all the defined tick filters.

Right side of the window is the tick filter editor. It allows you to set special formula for each ticks received.

Under the **Valid Condition** tab, the formula specifies the valid condition. If it is evaluated to true, the tick will be used in construction of the data series and subsequently used in indicator calculations like indicators. If it is evaluated to false, the tick is discarded (in both chart and tick filter).

Under the **Tick Value** tab is the formula for transforming tick value. If **Enable Transformation** is true, the tick value is modified using the formula.

Under the **Tick Volume** tab is the formula for transforming tick volume. If **Enable Transformation** is true, the tick volume is modified using the formula.

Under the **Tick Time** tab is the formula for transforming tick time. If **Enable Transformation** is true, the tick time is modified using the formula.

Add a New Tick Filter

To add a new tick filter, simply type in your formulas into the editor area, give a name to the filter, and then press **Save** button to save the filter. The new filter will show up in the list area at once.

To add a new tick filter based on an existing one, select an existing tick filter from the list first. The formulas of this tick filter will be shown in the editor area. Make your modification now and assign a new name to it. Press the **Save** button to save this new filter.

Edit an Existing Tick Filter

To edit an existing tick filter, select the tick filter from the list. Its formulas will be shown in the editor area. Make your changes and press the **Save** button. The tick filter will have its formulas updated.

Remove an Existing Tick Filter

To remove an existing tick filter, select the tick filter from the list on left side. Press the **Delete** button. The list will be updated to reflect the removal of the tick filter you have chosen.

Using a Tick Filter

If you want to filter a data series within a time chart:

1. Choose the data series.
2. Open the data setup window.
3. In the data setup window, under the **Misc** tab, you will find the tick filter drop-down list. Choose from one of the defined tick filters. Then press the **Apply** button. This data series will then be filtered tick by tick in real-time update.

To turn off the tick filter is simple. Simply choose **None** instead of a tick filter name. Then press **Apply** button. The data series will then update without tick filtering.

About Historical Bars

If you just specified a tick filter for a chart, note that only the new ticks coming in from real-time are filtered. Historical data is not filtered automatically (as the data may be in minute format and ticks are not readily available). To force a refilter, press the **Replay** button under **Replay by Tick** in **Misc** tab in chart manager.

Replay by tick is a time consuming operation as the historical ticks have to be downloaded from the data feed.

Tick Filter Formula Reference

There are three separate buffers stored in a tick filter. One for tick, one for bid and one for ask. Thus you can use the bid and ask buffers to help determine if a trade tick is good or not.

Tick Filter formula has the following context driven functions available. Each function has a single parameter that is used for referring to number of ticks ago. If the parameter is left out, then it is assumed that you are referring to the current tick, which is zero (0) tick ago.

Function	Meaning
`CLOSE`	the tick price function. Short form is `c`. The current tick price value is `CLOSE(0)` or simply `CLOSE`.
`VOLUME`	the tick volume function. Short form is `V`.
`TIME`	the time function. Short form is `T`.
`DATE`	the date function. Short form is `D`.
`COUNT`	Number of ticks in the buffer, e.g. `close(count - 1)` will reference the first tick in the buffer.
`BIDPRICE`	the bid price function. The current bid price is `BIDPRICE(0)` or simply `BIDPRICE`.
`BIDSIZE`	the bid size function.
`BIDTIME`	the bid time function
`BIDDATE`	the bid date function
`BIDCOUNT`	Number of bids in the buffer, e.g. `BIDPRICE(BIDCOUNT - 1)` will reference the first bid in the buffer.
`ASKPRICE`	the ask price function. The current ask price is `ASKPRICE(0)` or simply `ASKPRICE`.
`ASKSIZE`	the ask size function.
`ASKTIME`	the ask time function
`ASKDATE`	the ask date function
`ASKCOUNT`	Number of asks in the buffer, e.g. `ASKPRICE(ASKCOUNT - 1)` will reference the first ask in the buffer.

In addition, all the standard general formula functions are available.

If you need examples on how to use the context driven functions, take a look at the pre-defined tick filters **Price around $30** and **Price around $50**.

Special Application: Time Wrap

Since the tick filter engine is capable of transforming all the values from a tick into something else. You can use tick filters for the purpose of data transformation.

One example provided is called Time Warp. Price and Volume are not modified, but the time is transformed in the following way.

- Before 11:00 am. All data stay at the same time
- 11:00 am to 1:00 pm. Time runs 2x the normal speed, so 1:00 pm becomes 12:00 pm.
- After 1:00 pm – 4:00 pm. Time is slowed down to run from 12:00 pm to 4:00 pm.

Time wrap is part of the standard installation.

What Time Wrap Does

Time wrap is kind of like tick chart. For many instruments, 11:00am to 1:00pm is a period of slow trading (lunch hour). By compressing the 2 hours into 1 hour, the bars within the time and similar tick density to the rest of trading day. The result is more reliable indicators, especially for support and resistance type indicators.

Special Application: Large Trades Only Chart

Another possible usage of tick transformation is selective tick usage based on the tick volume.

For example, you can simply filter out ticks that have a volume less than 500 by using the formula `V >= 500` as the condition (or any volume you choose).

A chart with large trades only will reflect the action larger players in the market. Larger players typically define market trend. So a chart of large players is cleaner for trend determining indicators.

Tick Precise Indicators

The following indicators are called Tick Precise indicators:

- TP Tick N
- TP Tick Vol N
- TP Bid Ask Trade N
- TP Bid Ask Trade Vol N

Tick Precise Indicators summarize of tick level actions over the last N ticks. For these indicators, a positive value indicates bullishness and a negative value indicates bearishness.

Requirements

Tick Precise Indicators work on tick level, the data source must be a symbol that is receiving real-time streaming ticks.

Because Tick Precise indicators perform tick level calculations, to properly recalculate historical value, you must use Tick Replay feature in Chart Manager.

TP Tick N

Consider the last N ticks. If a tick is an up tick, it has a value of 1. If a tick is a down tick, it has a value of -1. TP Tick N is the sum of the tick values (the 1 and -1's) over the last N ticks.

Note that if you set N to 16, TP Tick N is equivalent to Tick16.

TP Tick Vol N

Consider the last N ticks. If a tick is an up tick, it has a value of the tick's volume. If a tick is a down tick, it has a value of the negative of the tick's volume. TP Tick Vol N is the sum of the tick volume values (the positive and negative volumes) over the last N ticks.

TP Bid Ask Trade N

Consider the last N ticks. If a tick is traded at or above ask, it has a value of 1. If a tick is traded at or below bid, it has a value of -1. TP Bid Ask Trade N is the sum of the bid ask trade values (the 1 and -1's) over the last N ticks.

TP Bid Ask Filtered Trade N

Identical to TP Bid Ask Trade N, with the addition field **Min Volume**. A tick does not count unless it has at least the minimum volume specified.

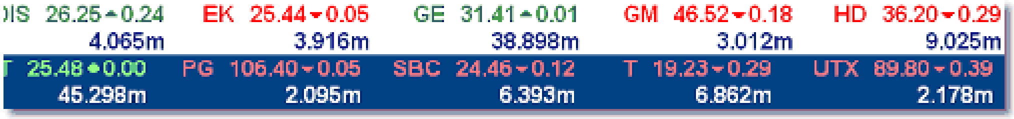
TP Bid Ask Trade Vol N

Consider the last N ticks. If a tick is traded at or above ask, it has a value of the tick's volume. If a tick is traded at or below bid, it has a value of the negative of the tick's volume. TP Bid Ask Trade Vol N is the sum of the bid ask volume values (the positive and negative volumes) over the last N ticks.

Ticker Operation Guide

Ticker is used to display intraday quotes in a continuously scrolling fashion that is commonly seen on financial television channels such as CNBC. Ticker takes considerably less screen space than a full scale quote window. So you can read your quotes without using up a lot of screen space.

Ticker can be displayed in a free floating window, or at the bottom of your screen. In the latter case, multiple tickers can be stacked.



)IS 26.25▲0.24 4.065m EK 25.44▼0.05 3.916m GE 31.41▲0.01 38.898m GM 46.52▼0.18 3.012m HD 36.20▼0.29 9.025m

T 25.48●0.00 45.298m PG 106.40▼0.05 2.095m SBC 24.46▼0.12 6.393m T 19.23▼0.29 6.862m UTX 89.80▼0.39 2.178m

Creating a Ticker

To create a ticker, either

- Press the ticker button , or
- Choose **Window>New>Ticker** from the main window.

Adding Symbols to the Ticker Window

When created, the ticker window is empty. One way to add symbols and quickly set up the ticker window is to use **Quick Settings** from the pop up menu.

For example, to set up a blue ticker for stocks in the Nasdaq 100, choose **Quick Settings>Big Blue**, then select one of current Nasdaq 100 symbol list.

Closing a Ticker Window

You can remove the window border from a ticker window to make it border less. In a borderless window, you don't have the standard window buttons to help you close the window.

To close a border less ticker window, you can either:

- Choose **Window>Close** from the main window, or
- You can choose **Show Window Border** from the ticker window's pop up menu. After the window border is shown, you can use the standard window close button to close the window.

Snapping

You can snap the ticker to the screen top or bottom by choosing **Snap To>Top** or **Snap To>Bottom** from the pop up menu.

Editing the Symbols

You can edit the symbols the ticker window displays.

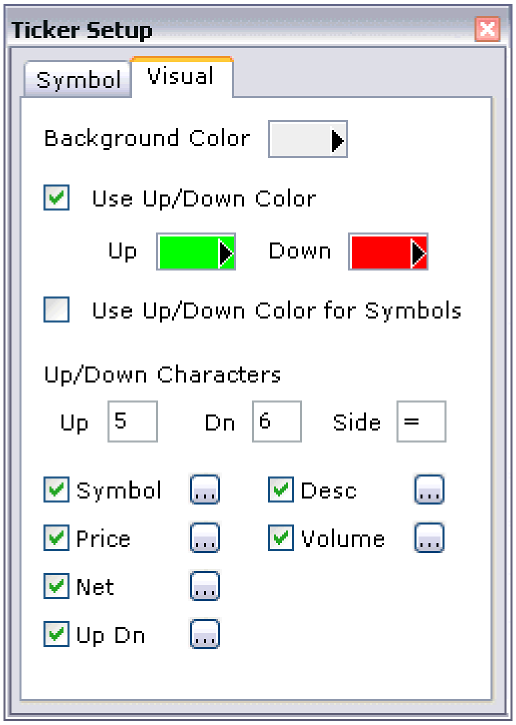
1. From the pop up menu, choose **Setup** to open the ticker setup window.
2. Under the **Symbol** tab, you can add, remove and sort the symbols.

Modifying the Look of the Ticker

You can customize the way the ticker looks.

1. From the pop up menu, choose **Setup** to open the ticker setup window

2 Settings are under the **Visual** tab.



Ticker Setup

Symbol | Visual

Background Color

☑ Use Up/Down Color

Up 🟩 Down 🟥

□ Use Up/Down Color for Symbols

Up/Down Characters

Up 5 Dn 6 Side =

☑ Symbol ☑ Desc

☑ Price ☑ Volume

☑ Net

☑ Up Dn

Creating Your Own Symbol List

Ticker uses the symbol list files from the `SymbolList` folder in the NeoTicker® installation. Refer to ***Tutorial: Creating a Symbol List*** (on page 155) for more information about symbol list.

Creating Your Own Quick Settings

Quick settings are simply templates with automatic symbol list application. After you are satisfy with the look of a ticker window, you can choose from the main window **Window>Save As Template** to create a template for the ticker.

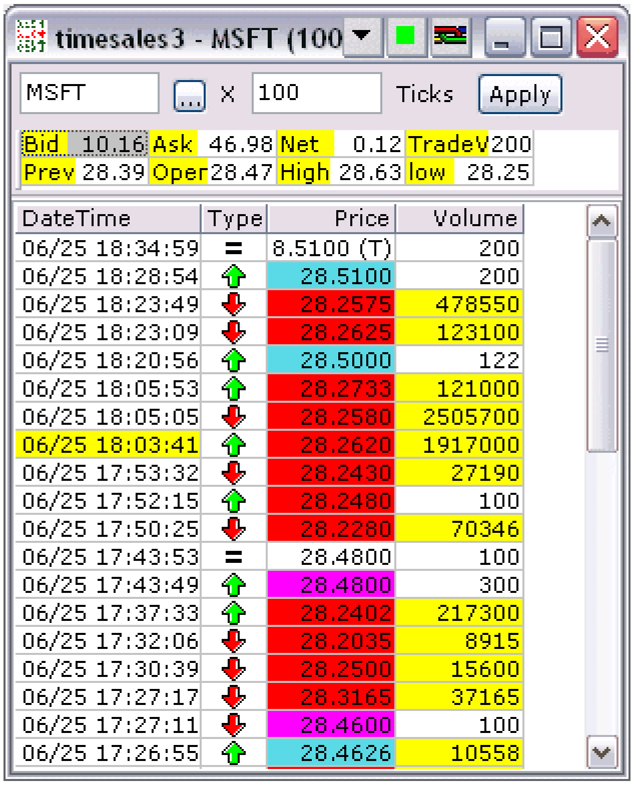
In addition to normal template actions, quick settings perform the follow actions:

- Removing the window border
- Snapping the window to the bottom

Time and Sales Window Operation Guide

NeoTicker®'s time and sales window lets you list the latest trades for a security. It offers you advance features such as customized coloring and simple charting to let you make the most out of time and sales records.

Basic Operations



MSFT ... X 100 Ticks

Bid	10.16	Ask	46.98	Net	0.12	TradeV	200
Prev	28.39	Oper	28.47	High	28.63	low	28.25

DateTime	Type	Price	Volume
06/25 18:34:59	=	8.5100 (T)	200
06/25 18:28:54	↑	28.5100	200
06/25 18:23:49	↓	28.2575	478550
06/25 18:23:09	↓	28.2625	123100
06/25 18:20:56	↑	28.5000	122
06/25 18:05:53	↑	28.2733	121000
06/25 18:05:05	↓	28.2580	2505700
06/25 18:03:41	↑	28.2620	1917000
06/25 17:53:32	↓	28.2430	27190
06/25 17:52:15	↑	28.2480	100
06/25 17:50:25	↓	28.2280	70346
06/25 17:43:53	=	28.4800	100
06/25 17:43:49	↑	28.4800	300
06/25 17:37:33	↑	28.2402	217300
06/25 17:32:06	↓	28.2035	8915
06/25 17:30:39	↓	28.2500	15600
06/25 17:27:17	↓	28.3165	37165
06/25 17:27:11	↓	28.4600	100
06/25 17:26:55	↑	28.4626	10558

Creating

1 Choose **Window>New>Time Sales**, or use the time and sales window tool button .

Tracking a Symbol

1 Enter a symbol into the symbol field in the upper left corner of the time and sales window.

2 Fill in the number of records you want to track in the field besides the **X** mark.

3 Press the **Apply** button.

Reading

An up tick in the time and sales window is indicated by a green up-arrow and is in green.

A down tick is indicated by a red down-arrow.

A side tick is indicated by an black equal sign.

It is possible that a tick is in bad sequence and the up/down tick information is unreliable. In this case, the tick will be indicated by a yellow cross sign.

Number of Ticks

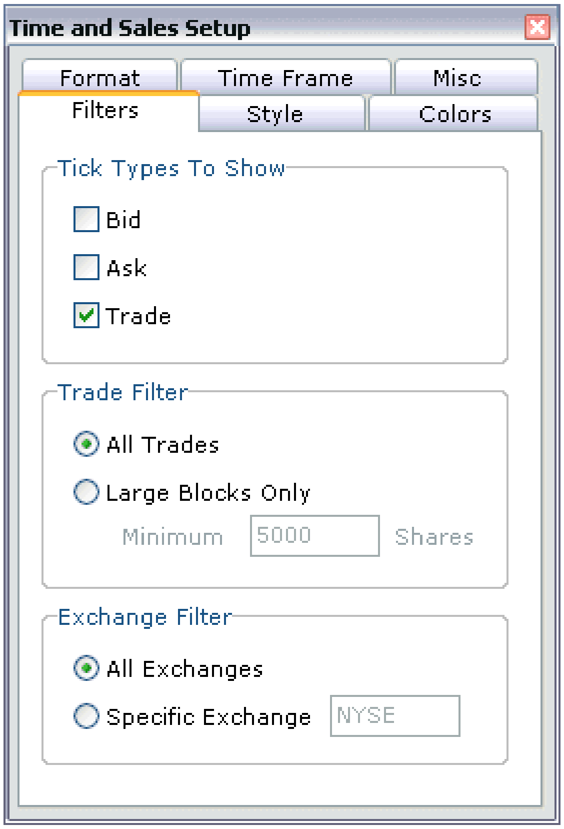
Number of ticks (**# of Ticks**) setting in the time and sales window controls how many ticks the window stores. Increase this number when you need to display more ticks. The number must be greater than or equal to 1

Pop up Menu

Right click anywhere in the time and sales window except in the summary panel to open the pop up menu.

Tick Filtering

You can open the time and sales filter setup by choosing **Filters** from the pop up menu. Filters are a set of controls that determines what information are displayed in the time and sales window on a tick-by-tick basis.



Time and Sales Setup

Tick Types To Show

□ Bid

□ Ask

☑ Trade

Trade Filter

● All Trades

○ Large Blocks Only

Minimum 5000 Shares

Exchange Filter

● All Exchanges

○ Specific Exchange NYSE

Tick Types To Show

Controls what type of ticks are to be displayed. You can turn on bid and ask support here.

Trade Filters

Provide a filter on the trades, e.g. you can set to display only trades with volume of 5000 or higher.

Exchange

If you choose Specific Exchange, only ticks from the specified exchange will be shown.

Colors

You can open the time and sales color setup by choosing **Colors** from the pop up menu. The colors provide the coloring rules time and sales record. For example, if today's high color is set to blue, then the time and sales record that makes a day high will have a blue color.

1 Right click on the time and sales window to open the pop up menu, and choose **Colors.**

2 Adjust the color. Time and sales window will change color immediately.

Marking Large Blocks

Time and sales window marks large block trades in a special color. To change the settings:

1 In time and sales window, right click to open pop up menu, choose **Style** to open the set up dialog for changing style.

2 Adjust the **Large Blocks** parameter.

Large blocks are marked by yellow in the volume field. To adjust the large block color:

1 In time and sales window, right click to open pop up menu, choose **Color** to open the set up dialog for changing style.

2 Adjust the color for **Large Blocks**.

Changing Tick Style (UD and UDS Ticks)

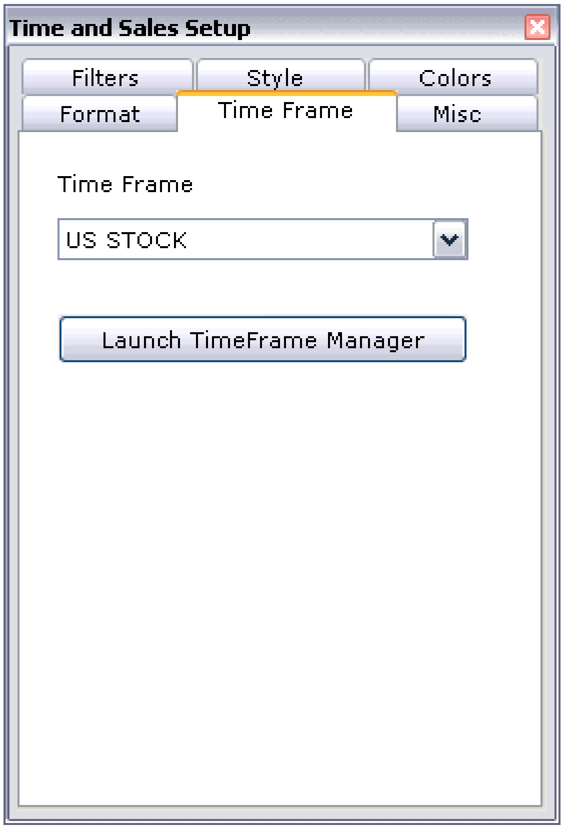
Time and sales window let you choose between Up/Down style ticks, or Up/Down/Side style ticks:

1 In time and sales window, right click to open pop up menu, choose **Style** to open the set up dialog for changing style.

2 Choose **Up / Down** or **Up / Down / Side**. Time and sales will update immediately.

Time Frame

You can open the time and sales time frame setup by choosing **TimeFrame** from the pop up menu.



Time and Sales Setup

Time Frame: US STOCK

Time and sales window supports indicator calculation and display in the summary panel (dynamic grid). Indicator calculations require the knowledge of trading hours, holidays, etc. These information are collectively known as time frame. Time frames are created and managed by the time frame manager. By default, the US STOCK time frame is used. If you are tracking the time and sales record of index futures, you will need to use the US INDEX time frame for the indicators to be calculated correctly.

Price Format

You can adjust how price are formatted, e.g. number of decimal places, fractions, etc.

1. Right click on time and sales window, and choose **Format**.
2. The setup window is open, the **Format** tab is selected. Under this tab, you can adjust the variables that affect formatting.

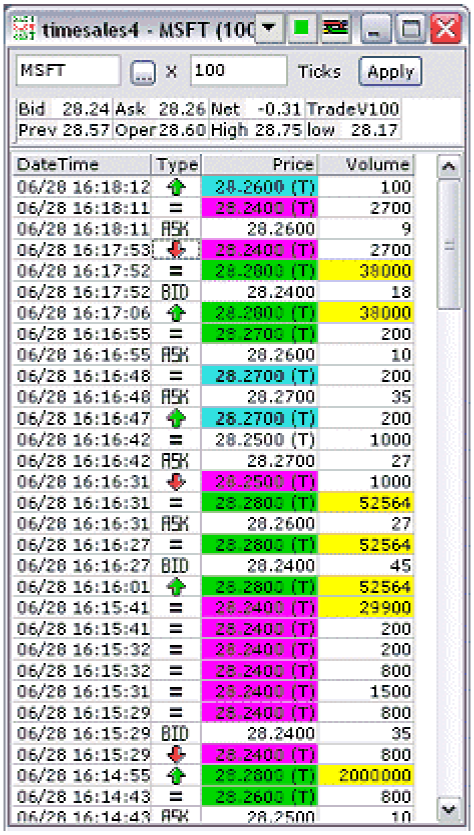
Showing Bid and Ask Records

By default, bid and ask records are not shown in time and sales windows. You can turn on the bid and ask support to display bid and ask records.

1. Right click on the time and sales window to open the pop up menu, and choose **Filters.**
2. A setup window is opened. Check the **Bid** and **Ask** options under the **Filters** tab to enable bid and ask display.

Below is an example of time and sales window with bid and ask records displayed.

Unlike normal trades, the bid and ask volume are shown in lots.



MSFT ... x 100 Ticks Apply

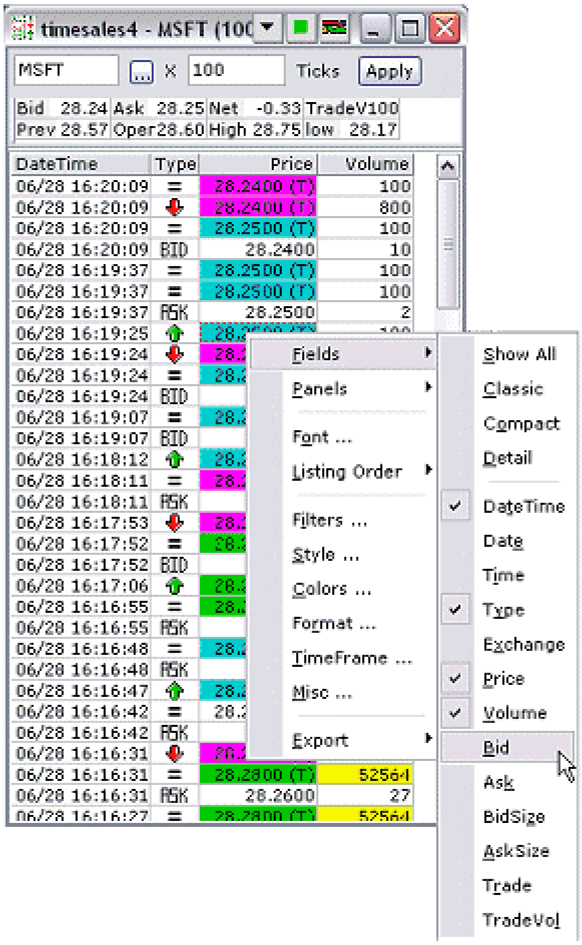
Bid	28.24	Ask	28.26	Net	-0.31	TradeV	100
Prev	28.57	Oper	28.60	High	28.75	low	28.17

DateTime	Type	Price	Volume
06/28 16:18:12	↑	28.2600 (T)	100
06/28 16:18:11	=	28.2400 (T)	2700
06/28 16:18:11	ASK	28.2600	9
06/28 16:17:53	↓	28.2400 (T)	2700
06/28 16:17:52	=	28.2800 (T)	39000
06/28 16:17:52	BID	28.2400	18
06/28 16:17:06	↑	28.2800 (T)	39000
06/28 16:16:55	=	28.2700 (T)	200
06/28 16:16:55	ASK	28.2600	10
06/28 16:16:48	=	28.2700 (T)	200
06/28 16:16:48	ASK	28.2700	35
06/28 16:16:47	↑	28.2700 (T)	200
06/28 16:16:42	=	28.2500 (T)	1000
06/28 16:16:42	ASK	28.2700	27
06/28 16:16:31	↓	28.2500 (T)	1000
06/28 16:16:31	=	28.2800 (T)	52564
06/28 16:16:31	ASK	28.2600	27
06/28 16:16:27	=	28.2800 (T)	52564
06/28 16:16:27	BID	28.2400	45
06/28 16:16:01	↑	28.2800 (T)	52564
06/28 16:15:41	=	28.2400 (T)	29900
06/28 16:15:41	=	28.2400 (T)	200
06/28 16:15:32	=	28.2400 (T)	200
06/28 16:15:32	=	28.2400 (T)	800
06/28 16:15:31	=	28.2400 (T)	1500
06/28 16:15:29	=	28.2400 (T)	800
06/28 16:15:29	BID	28.2400	35
06/28 16:15:29	↓	28.2400 (T)	800
06/28 16:14:55	↑	28.2800 (T)	2000000
06/28 16:14:43	=	28.2600 (T)	800
06/28 16:14:43	ASK	28.2500	10

Displaying and Hiding Fields

You can have different fields displayed in time and sales window for your particular trading need. Example of fields are date time, volume. To include/remove fields:

1 Right click on the time and sales record to open the pop up menu, and choose the fields from the **Fields** entry.



MSFT ... X 100 Ticks Apply

Bid 28.24	Ask 28.25	Net -0.33	TradeV100
Prev 28.57	Oper28.60	High 28.75	low 28.17

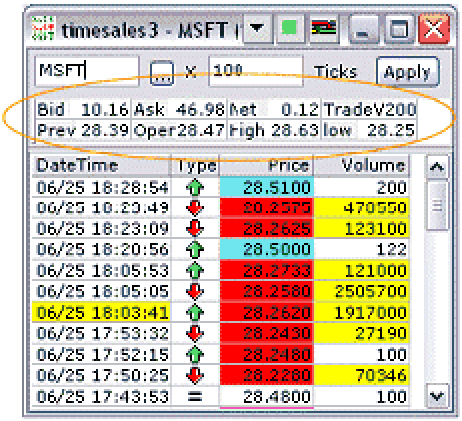
DateTime	Type	Price	Volume
06/28 16:20:09	=	28.2400 (T)	100
06/28 16:20:09	↓	28.2400 (T)	800
06/28 16:20:09	=	28.2500 (T)	100
06/28 16:20:09	BID	28.2400	10
06/28 16:19:37	=	28.2500 (T)	100
06/28 16:19:37	=	28.2500 (T)	100
06/28 16:19:37	ASK	28.2500	2
06/28 16:19:25	↑	28.2	100
06/28 16:19:24	↓	28.2	
06/28 16:19:24	=	28.2	
06/28 16:19:24	BID		
06/28 16:19:07	=	28.2	
06/28 16:19:07	BID		
06/28 16:18:12	↑	28.2	
06/28 16:18:11	=	28.2	
06/28 16:18:11	ASK		
06/28 16:17:53	↓	28.2	
06/28 16:17:52	=	28.2	
06/28 16:17:52	BID		
06/28 16:17:06	↑	28.2	
06/28 16:16:55	=	28.2	
06/28 16:16:55	ASK		
06/28 16:16:48	=	28.2	
06/28 16:16:48	ASK		
06/28 16:16:47	↑	28.2	
06/28 16:16:42	=	28.2	
06/28 16:16:42	ASK		
06/28 16:16:31	↓	28.2	
06/28 16:16:31	=	28.2800 (T)	52564
06/28 16:16:31	ASK	28.2600	27
06/28 16:16:27	=	28.2800 (T)	52564

Exporting Time and Sales Data to Microsoft Excel

You can export the time and sales data directly to Microsoft Excel.

1 Right click to open the pop up menu, and chose **Export>Excel**.

Summary Panel Setup



Bid	10.16	Ask	46.98	Net	0.12	TradeV	200
Prev	28.39	Oper	28.47	High	28.63	low	28.25

DateTime	Type	Price	Volume
06/25 18:28:54	⇧	28.5100	200
06/25 10:23:49	⇩	20.2575	470550
06/25 18:23:09	⇩	28.2625	123100
06/25 18:20:56	⇧	28.5000	122
06/25 18:05:53	⇧	28.2733	121000
06/25 18:05:05	⇩	28.2580	2505700
06/25 18:03:41	⇧	28.2620	1917000
06/25 17:53:32	⇩	28.2430	27190
06/25 17:52:15	⇧	28.2480	100
06/25 17:50:25	⇩	28.2280	70346
06/25 17:43:53	=	28.4800	100

The summary panel in a time and sales window is a dynamic grid. For more information, refer to ***Dynamic Grid Operation Guide*** (on page 545).

Time Chart Operation Guide

In this section, we will show you time chart operations. NeoTicker® is designed with different users and trading styles in mind. In general, you will find more than one way of doing something. You can simply pick the way that best fit your style.

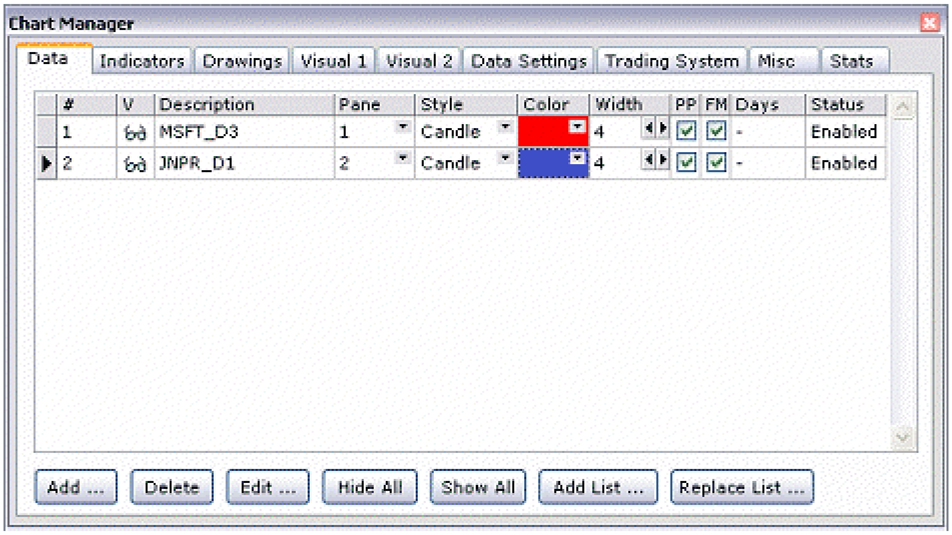
Basic Operations

This section covers the basic chart operations. Most users will adapt to certain usage style of NeoTicker® and do not need all the features here. However, you do want to at least quickly browse through the sections here so you know what is available.

Chart Manager

Chart Manager is the central place to manage chart objects and chart parameters. You can open Chart Manager window by:

- Pop up menu by choosing **Chart Manager.**
- Chart manager button on the user panel and tool bar.
- Quick Jump button on the chart's caption bar.



#	V	Description	Pane	Style	Color	Width	PP	FM	Days	Status
1	👓	MSFT_D3	1	Candle	🟥	4	☑	☑	-	Enabled
2	👓	JNPR_D1	2	Candle	🟦	4	☑	☑	-	Enabled

Quick Reference

Here is a brief explanation of the tabs in Chart Manager:

Tab	Function
Data	Lets you manage data series
Indicators	Lets you manage indicators
Drawings	Lets you manage drawing tools
Visual 1	Lets you change the visual appearance of the chart, part 1
Visual 2	Lets you change the visual appearance of the chart, part 2
Data Setting	Lets you control how the chart interprets and loads data
Trading System	Controls trading system markings
Misc	Miscellaneous options
Stats	Chart statistics for trouble shooting purpose

Data Tab

Here is an explanation of the columns under **Data** tab.

Item	Function
V	Toggles visibility of the data series
Description	Describes the data series
Pane	The pane the data series resides
Style	How the data series is drawn
Color	Color of the data series
Width	Width of the data series
PP	Part of pane scale setting
FM	Float marker setting for the data series

Days	Days of data of the data series. If the data series uses time chart's default days to load, a - sign is shown.
Status	Whether the data series is on
Add	Add data series
Delete	Delete data series
Edit	Edit data series
All>Hide	Make all data series invisible
All>Show	Make all data series visible
All>Delete	Delete all data series
Add List	Add data series with a symbol list
Replace List	Replace current data series with a symbol list

Indicators Tab

Here is an explanation of the columns under **Indicators** tab.

Item	Function
V	Toggles visibility of the indicator
Description	Describes the indicator
Pane	The pane the indicator resides
Plot	Plot number
Vis	Plot visibility
Style	How the indicator is drawn
Color	Color of the indicator
Width	Width of the indicator
PP	Part of pane scale setting
FM	Float marker setting
Status	Whether the indicator is on

Add	Add indicator
Delete	Delete indicator
Edit	Edit indicator
Propagate	Apply the selected indicator to all data series
All>Hide	Make all indicators invisible
All>Show	Make all indicators visible
All>Delete	Delete all indicators
Trading System	If an indicator is a trading system, this button open a pop up menu that provides the equivalent functions of the Trading System submenu in the chart.
Show All Plots/Show First Plot	Information for all plots of the indicators are visible / Only information for first plot of the indicators is visible

You can choose to show all plots/first plot only of the indicator by pressing the **Show All Plots / Show First Plot** button.

Drawings Tab

Here is an explanation of the columns:

Item	Function
V	Toggles the visibility of the drawing tool. Only drawing tools with proper attached symbols are visible
Description	Describes the drawing tool
Pane	The pane the drawing tool resides
Style	The drawing style
Color	Color
Width	Width
Label	Whether labels are shown
Symbol	Symbol the drawing tool is attached to

Copy	Copy drawing tool, except snapping properties
Copy w/ Snap	Copy drawing tool with snapping properties
Delete	Delete drawing tool
Edit	Edit drawing tool
Hide All	Make all drawing tools invisible
Show All	Make all drawing tool visible, subjected to attached symbols
Delete All	Delete all symbols

Visual 1 Tab

Under this tab are various visual settings. In general, settings under this tab has an immediate effect on the chart and you do not need to explicitly apply the changes.

Item	Function
Show Grid	Grid lines in the chart
Show Legend	Whether data series and indicator legends are visible
Day Break	Vertical line at the end of trading day. Useful for intraday charts only
Wide Price Axis	Extra space for price axis labels
Time Scroll Bar	Whether the horizontal scroll bar is visible
Floating Marker	Whether the float markers for the chart is on
Window Border	Whether chart's window border is visible
Last Time/Sales	Controls what is being displayed in the report area of the chart (top area)
Data Alignment	Where data is placed when there is insufficient number of bars to fill the chart. See ***Time Chart Configuration>Data Alignment.*** **(see "**Data Alignment**" on page 1073)**
Large/Medium/Small	Let you quickly set font size for the chart
Change	Select an item above the **Change** button, then press the **Change** button to modify font setting for the item

Coloring Scheme	For changing the color of specific items
Default	Reset to default coloring scheme
Inverse	Quickly inverse the current color

Visual 2 Tab

Under this tab are various visual settings. In general, settings under this tab require you to press the **Apply** button to apply any changes.

Item	Function
Auto Scroll Chart	See ***Auto Chart Scrolling*** (on page 1071).
Redraw Frequency	See ***Redraw Frequency*** (on page 1080).
Default Zoom Level	See ***Default Zoom Policy*** (on page 1073).
Right Side Spacing	See ***Controlling the Right Side Spacing*** (see "Right Side Spacing" on page 1081).

Data Settings Tab

Under this tab are settings that affect how data is handled in the chart. You need to press the **Apply** button to apply any changes.

Item	Function
Days To Load	See ***Days of Data to Load*** (on page 1029).
Time Axis Style	See ***Even Bar Space between Bars*** (on page 1075).
Time Frame Auto Switched Based On Primary Symbol, Predefined Time Frame, Customized, Trading Time, Trading Days, Holiday List	Time frame settings. You can choose the setting to follow primary symbol, use a pre-defined time frame, or use a customized time frame. See ***Trading Time and Holiday List Settings*** (on page 1086).
Last Day	See ***Historical Charts*** (on page 1032).
Conserve Memory	See ***Conserve Memory*** (on page 1071).

Trading System Tab

Under this tab are settings that affect trading system markings if the chart contains one or more trading system. See ***Trading System Markings*** (on page 1185).

Misc Tab

Item	Function
Replay by Tick	See ***Replay by Tick*** (on page 1123).
Complete Reload	See ***Completely Reloading all Data*** (see "Replay by Tick" on page 1123).
Data View Window>Show	For resetting Data View window position to upper left corner of the chart. See ***Data View Window*** (on page 1030).

Stat Tab

This tab is used for trouble shooting chart related performance problems. As chart does not have explicit limit on number of data series, indicators and drawing tools, it is easy to overload the chart with too many items and cause performance problems.

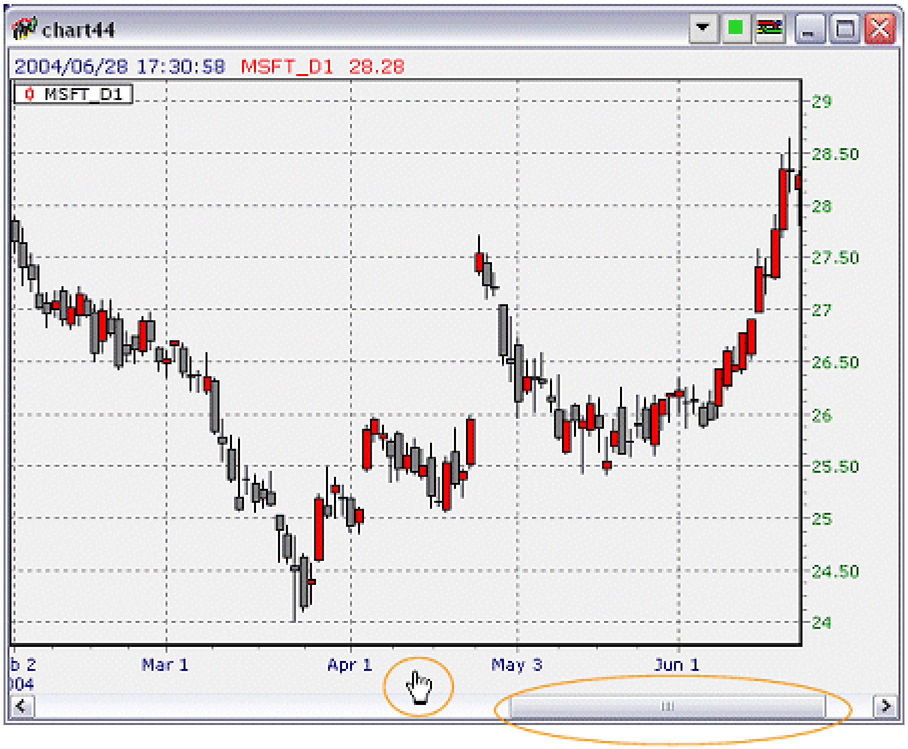
General rule of thumb:

- Celeron 1.4GHz with onboard graphics card can handle about 1,000,000 visible bars on screen.
- Pentium 4 2.8GHz with 64M AGP graphics card can handle about 5,000,000 visible bars on screen.

Chart Scrolling

There are several ways to scroll a chart:

- Use the scroll bar
- Use the mouse wheel
- Use the arrow keys on the keyboard
- Dragging the the time axis area



You can hide the scroll bar for more chart space. See ***Hiding and Showing the Scroll Bar*** (on page 1078).

When you use arrow keys to scroll, the chart is scrolled one bar at a time. If you have more than one data series in the chart, the first data series is used as the reference. If you hold the CTRL key while pressing the arrow key, the chart is scrolled 10 bars at a time.

When the chart is realtime updating, the chart is also obeying rules for right side chart spacing (see ***Controlling the Right Side Spacing*** (see "Right Side Spacing" on page 1081)). Right side spacing rules can be inconsistent to the scrolling actions you make and can override the scrolling actions.

Creating Time Chart Window

To create a new time chart, either:

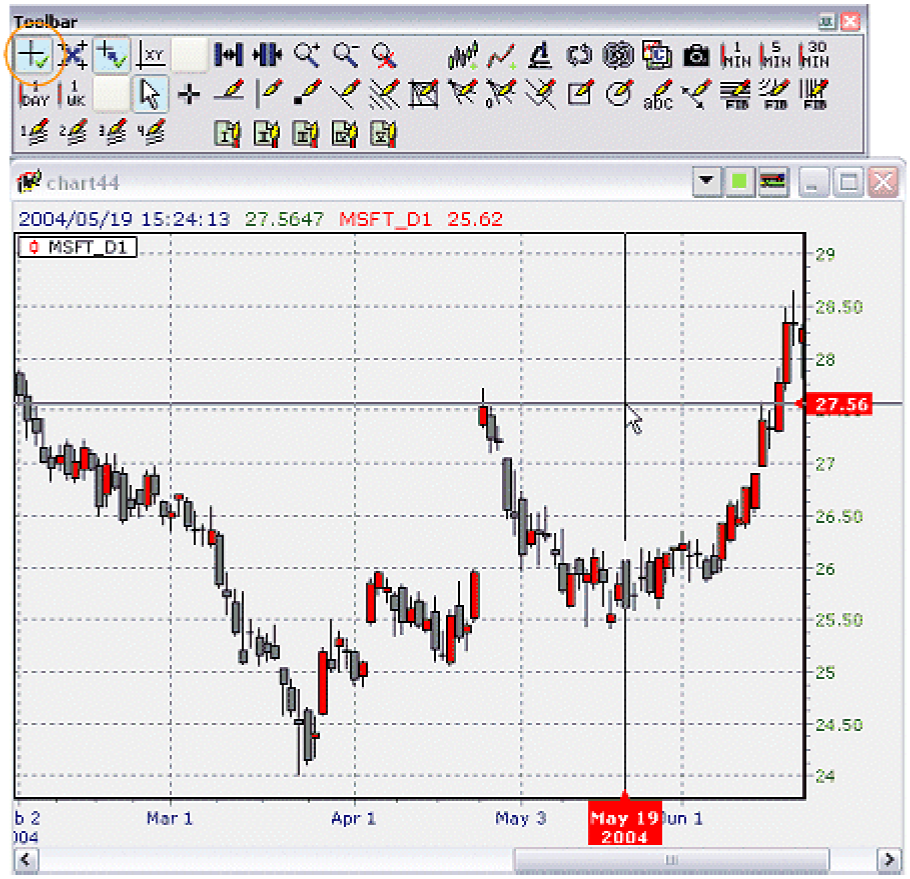
- Press the time chart button , or
- Choose **Window>New>Time Chart** from the main window.

A new time chart window is blank with one pane in it.

Crosshair and Global Crosshair

Crosshair

You can toggle the crosshair display on/off by pressing the crosshair button in the tool bar window. The crosshair moves with the cursor and extends vertically and horizontally to help you read the chart.



2004/05/19 15:24:13 27.5647 MSFT_D1 25.62

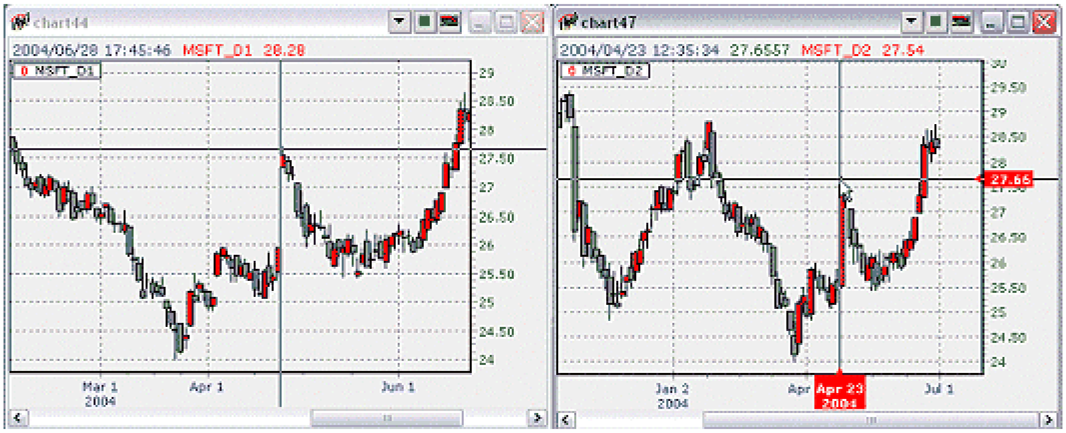
Global Crosshair

Another type of crosshair is global crosshair. Global crosshair works on multiple chart windows. When you position your mouse on one chart, the time and price is used to draw crosshair on other charts. Global crosshair is useful for pinpointing time/price across different charts for comparison purposes.

Let's open several chart windows to see how global crosshair work. Add symbols that similar in time and price to these charts. For example, you can use 1-day, 2-day and 3-day MSFT.

Turn on global crosshair by pressing the global crosshair button on the tool bar.

Now move the cursor, and you will see all the charts receive a global crosshair update. Note that you do not have to explicitly turn on normal crosshair display on all the charts for them to draw global crosshair. Also, if you have multiple panes in the chart, each pane will draw its own price horizontal line in the crosshair.



Individual charts do not have to receive global crosshair updates. To turn off global crosshair for a single chart, press the enable/disable button for global crosshair.

To set global crosshair options, choose **Manager>Global Crosshair** from the main window. You can make global crosshair to draw time and price crosshair (vertical and horizontal lines) or time only crosshair (vertical line only).

Ctrl-Space Bar Short Cut

This feature only works with a broadcast data feed with IDS. It does not work with Internet data feed because a symbol universe is not available.

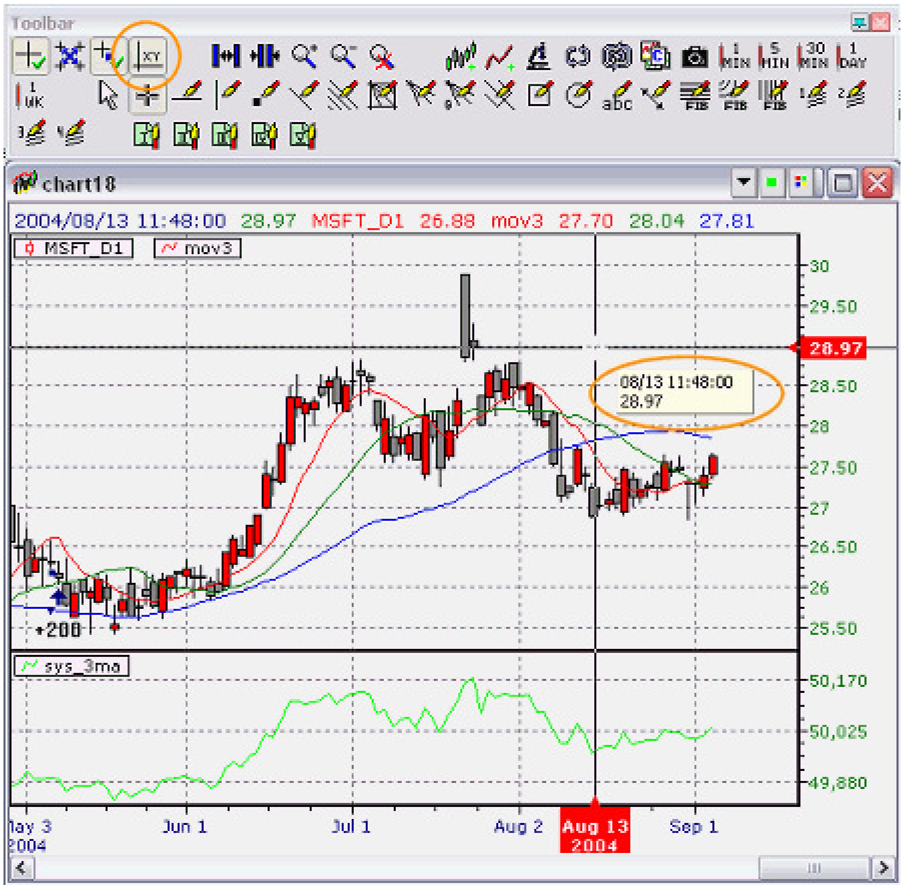
If you press ctrl-space bar when nothing is selected, you will bring up the select symbol window.

This window lets you quickly replace the symbol of the first data series. All instances of the symbol are replaced.

This feature only works if the first data series uses the real-time server. It does not work with end of day data.

Cursor Values

If you turn on cursor values, the chart will display the cursor values at the lower right of the cursor. This feature lets you view the exact value the cursor is pointing at. To turn on the feature, press the cursor value button on the tool bar.



Keyboard Short Cut

Without turning on button on tool bar, you can use cursor value by moving the mouse while holding the SHIFT key.

If you hold the CTRL key while moving the mouse, an extended version of cursor value is shown.

Data Series and Indicators

Time chart has two types of series: data series and indicators.

A data series has an underlying security. The security can be a stock, index, future, etc.

An indicator is the result of calculation you performed on data series and indicators. Moving Average is an example of an indicator.

Terminology:

Series — In this manual the term Series refers to either a data series or an indicator. When distinction is needed, the terms data series and indicators are used.

Time Chart — We use the terms chart and time chart interchangeably in this series of manuals.

Data Series, Indicator and Drawing Tool Selection

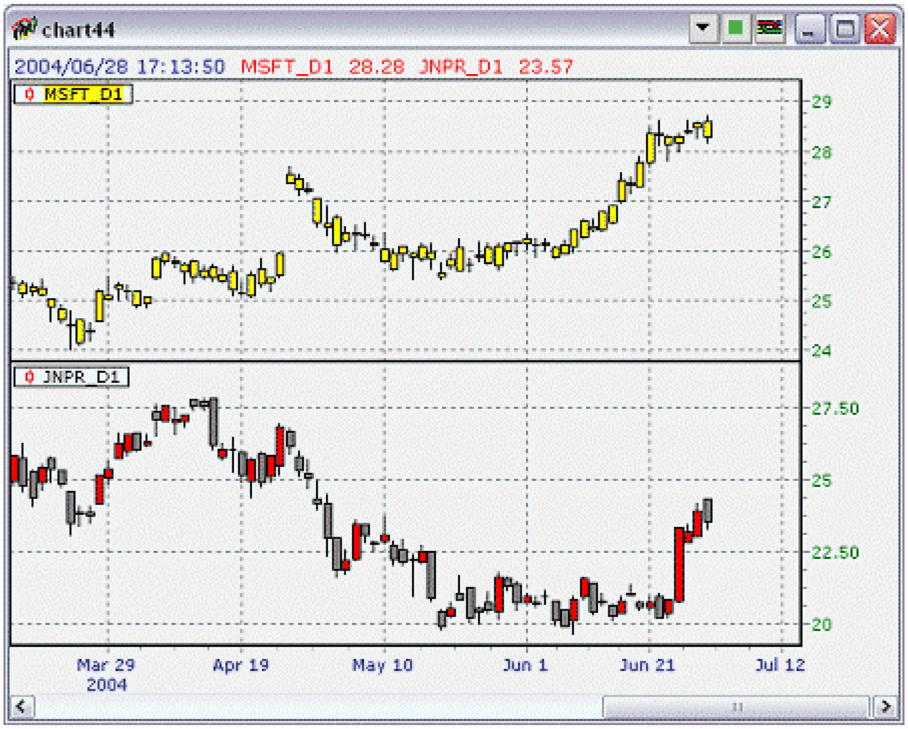
Some operation depends on the current selection (e.g. you select a data series and press DELETE key to delete it).

There are three ways to select a data series, an indicator or a drawing object:

- Select the data series, indicator or drawing tool directly by clicking on the item in the time chart window
- Select the data series or indicator by clicking on its legend (not applicable to drawing tools)
- Select the item by clicking on its name in Chart Manager

Direct selection depends on the cursor you use. You can select data series, indicators and drawing tools using the pointer cursor. If you use the crosshair cursor, you can only directly select drawing tools. You have to use legend or chart manager to select data series and indicators.

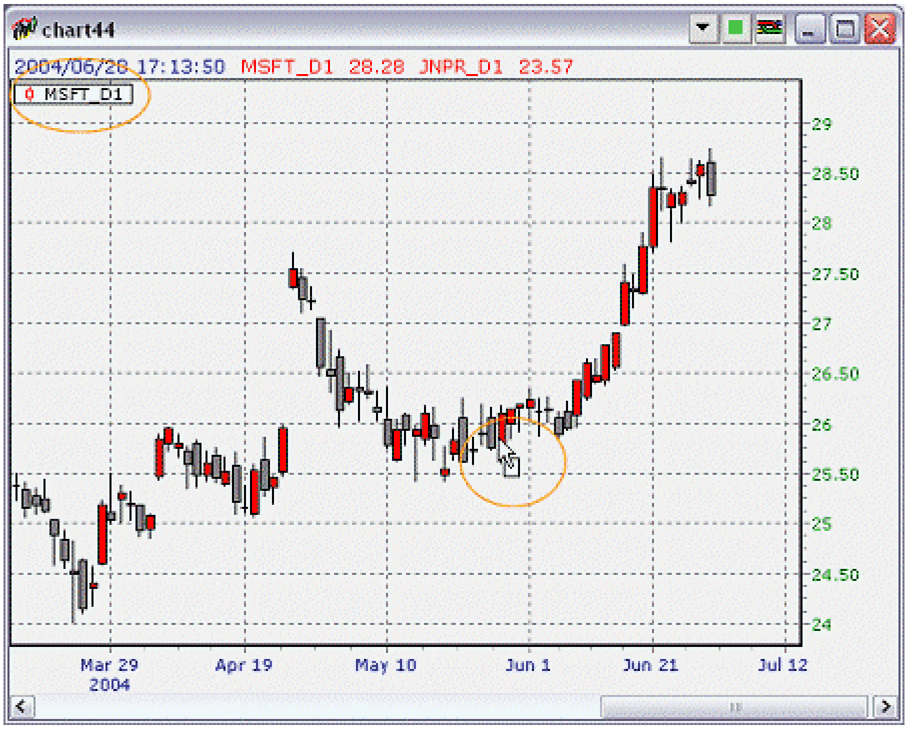
When an item is selected, it is highlighted in yellow. In the following figure, the data series MSFT is selected while the data series JNPR is not.



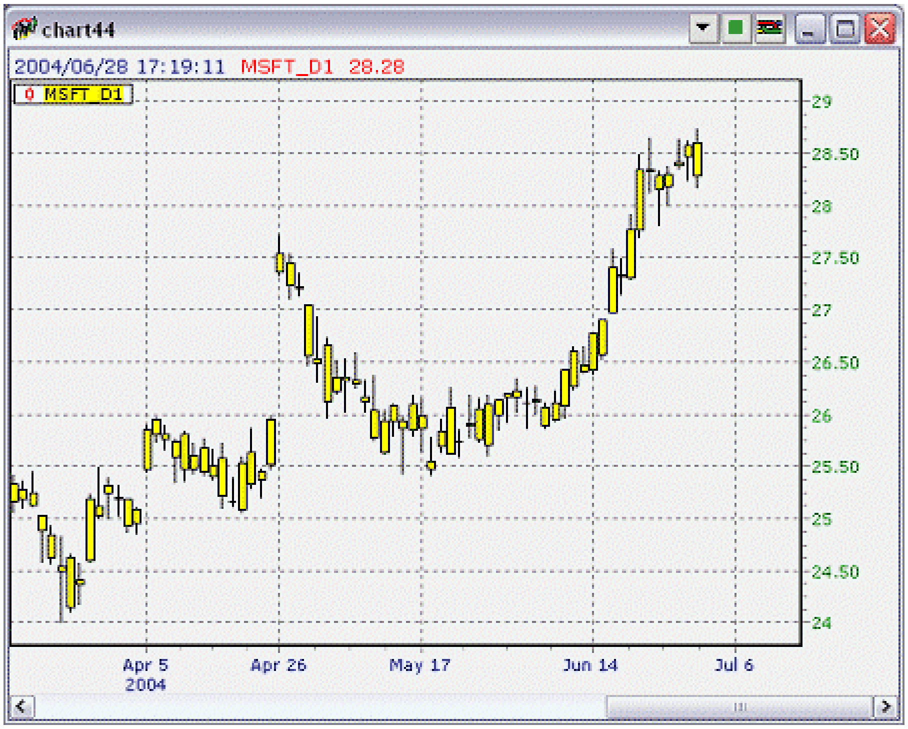
Selecting Data Series and Indicator

To select an object, move the cursor near the object and click on it. You can tell if you can select an object by looking at the cursor.

For data series and indicators, when you move the cursor near the object and the cursor changes to a drag cursor, you can click and select the data series or indicator. Often it is not easy to exactly pinpoint an object to select it. For data series or indicator, you can also click on the its legend to select it.

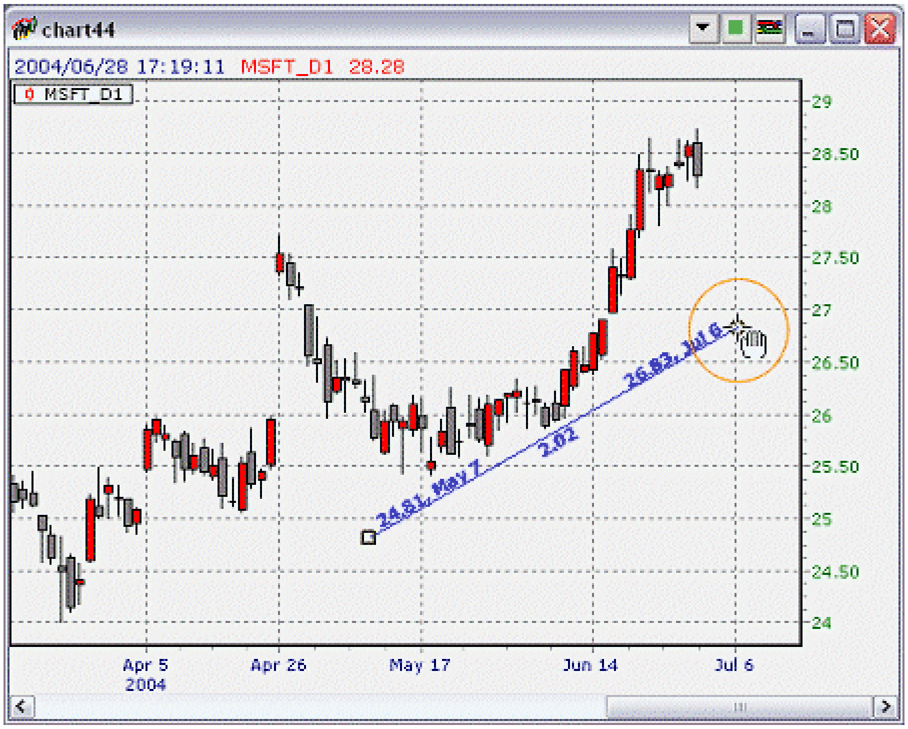


A selected object is highlighted. The default highlight color is yellow. So when a data series is selected, it looks like:



Selecting Drawing Tool

For drawing tools, if the cursor changes to a hand cursor, you can click and select the drawing tool.



Cursor affects object selection. When you are using the pointer cursor, you can directly click on any object to select it.

When you are using the crosshair cursor, you can only directly click on and select drawing tools. If you need to select a data series or indicator, you can click on its legend. This is often desirable when you are working on many drawing tools because the crosshair cursor reduces the chance of accidentally selecting a data series or an indicator.

Un-select an Object

To un-select an object, click on a blank area in the chart. Alternatively, you can use the ESC key on your keyboard to un-select.

Days of Data to Load

Time chart uses a static days to load per chart to ensure maximum calculation consistency.

Each chart has a default days of data to load. To change the setting, either:

- Use the pop up menu, choose **Days ot Data to Load**,
- Use chart manager, **Data Settings** tab, or
- User panel

You can override the days to load for individual data series, see ***Control Days to Load for Individual Data Series*** (on page 1107).

Calendar Days vs. Trading Days

Days to load either means calendar days or trading days. To control which definition is used:

1. Open Chart Manager.
2. Press the **Data Settings** tab.
3. Choose **Calendar Days** or **Trading Days** under **Days to Load**.

Current trading day is count as one day, even if it is not completed.

This setting affects individual data series days to load. If you override a data series with its own days to load, the definition will follow the settings for the chart. For example, if a data series has a override 5 days to load and the chart has a trading days setting, 5 trading days of data will be loaded for the data series.

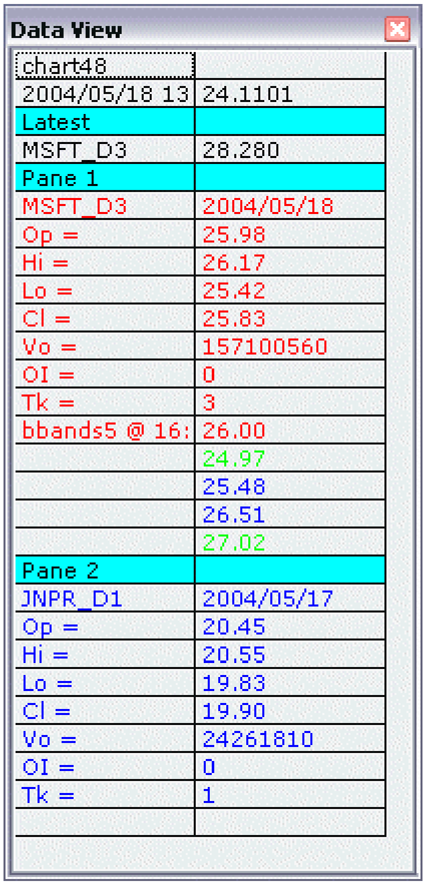
Data View Window

Data view window displays the data series and indicators in the chart in a tabular format. Data view is shared by all chart windows. When your cursor moves from one chart window to another, the data view will change accordingly.

To open data view, either:

- Choose **Data View** from a chart pop up menu;
- Press the data view button on the tool bar; or
- Press the F6 key on the keyboard.

Data view options are set by the pop up menu of data view. Right click on the data view window to open the pop up menu.



Data View

chart48	
2004/05/18 13	24.1101
Latest	
MSFT_D3	28.280
Pane 1	
MSFT_D3	2004/05/18
Op =	25.98
Hi =	26.17
Lo =	25.42
Cl =	25.83
Vo =	157100560
OI =	0
Tk =	3
bbands5 @ 16:	26.00
	24.97
	25.48
	26.51
	27.02
Pane 2	
JNPR_D1	2004/05/17
Op =	20.45
Hi =	20.55
Lo =	19.83
Cl =	19.90
Vo =	24261810
OI =	0
Tk =	1

At the top of the data view window is the time and value that correspond to the cursor position.

The rest of the data view window lists the values of the visible data series and indicators. The values are the data series or indicator's values at the time position of the cursor.

Controlling What to Show

You can choose what information to display by choosing one of the item under **Show**. You can control whether to display chart name, time and price information, pane number, data series values and indicator values.

Label Format

Under the **Label** menu, you can control the alignment and the length of the labels.

Auto Size

Turning on **Auto Size** will make the data view window scaled to accommodate different charts. For charts that have more data, auto sizing will make the data view window bigger. For charts that have few data, auto sizing will make the data view window smaller.

Auto Snap

Turning on **Auto Snap** will make the data view window snapped to the chart window from which the data is coming from. To change the snap position, choose **Snap Position**. A dialog will be opened to let you choose the snapping position.

Grid Settings

You can turn on/off vertical and horizontal grid lines in the pop up menu under **Grid**.

Coloring Scheme and Font

By default, data view window follows the coloring scheme of the chart window. If the chart has a white background, the data view window will have a white background. The text color depends on the color of the data series and indicator.

You can override this coloring scheme by choosing **Enable User Defined Settings**. Once this option is enabled, you can specify custom colors such as using **User Defined Font** (you can change color in the font dialog) and **User Defined Background Color**.

Item	Meaning
User Defined Font	Setting the font and default color of Data View.
User Defined Background Color	Setting the background color of Data View.
User Defined Separator Color	Setting the separator color of Data View. Separator color is used by rows that act as separators (e.g. Pane1, Pane2, etc).
User Defined Grid Color	Setting the grid color.

Labels Use Grid Color	When enabled, the labels (data in first column) will use the color of the plot/data series if available.
Values Use Grid Color	When enabled, the values (data in second column) will use the color of the plot/data series if available.

Restoring Position

If you have lost the Data View window (this can happen after you re-arrange multiple monitors), you can restore Data View window's position to near the chart.

1 Open ***Chart Manager*** (on page 1013) using caption button or pop up menu.

2 In Chart Manager, press **Misc** tab.

3 Under **Data View Window**, press **Show** button.

This operation will remove any snapping on the Data View and move it to the upper left corner of the chart.

Deleting Data Series, Indicator and Drawing Tool

To delete an object, select it and press the DELETE key on your keyboard.

You can also delete objects in user panel and chart manager.

Historical Charts

You can specify a last day for the chart. Once set, the chart becomes a historical chart and data will be cut off on the last day.

Historical charts are useful for studying past performance without the influence of new data. Trading system writers can use this feature to measure historical performance in different periods.

To set up the last day:

1 Open chart manager.

2 Press **Data Settings** tab.

3 Choose **Specific Date** under **Last Day**, and select day using the drop down.

4 Press the **Apply** button.

Choose **Most Recent Data** under **Last Day** to revert the chart to a real-time chart.

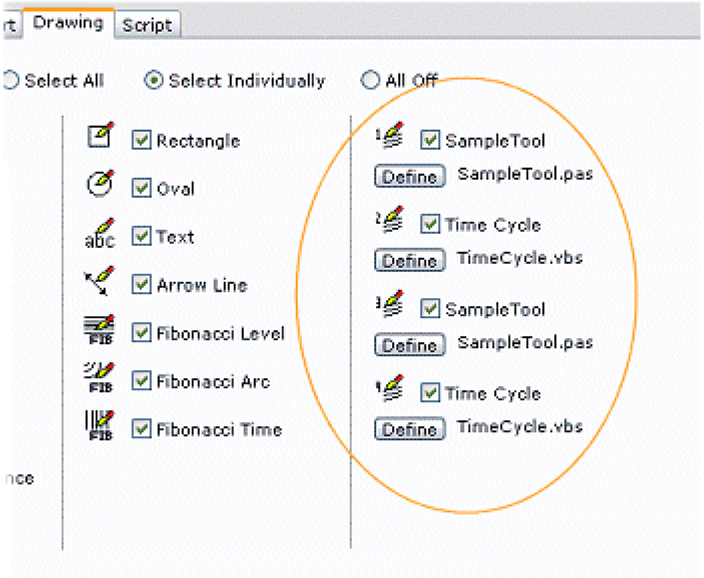
Installing Custom Drawing Tool Scripts

You can install custom drawing tools provided by TickQuest, third party developers, or created by yourself which can carry out specialized drawing capabilities.

To install a custom drawing tool:

1 Right click on the tool bar to open tool bar's pop up menu, choose **Setup** to open the tool bar set up

2 On the right side of the **Drawing** tab, there are 4 items called Custom Drawing Tool buttons



○ Select All ● Select Individually ○ All Off

☑ Rectangle
☑ Oval
☑ Text
☑ Arrow Line
☑ Fibonacci Level
☑ Fibonacci Arc
☑ Fibonacci Time

☑ SampleTool
Define SampleTool.pas
☑ Time Cycle
Define TimeCycle.vbs
☑ SampleTool
Define SampleTool.pas
☑ Time Cycle
Define TimeCycle.vbs

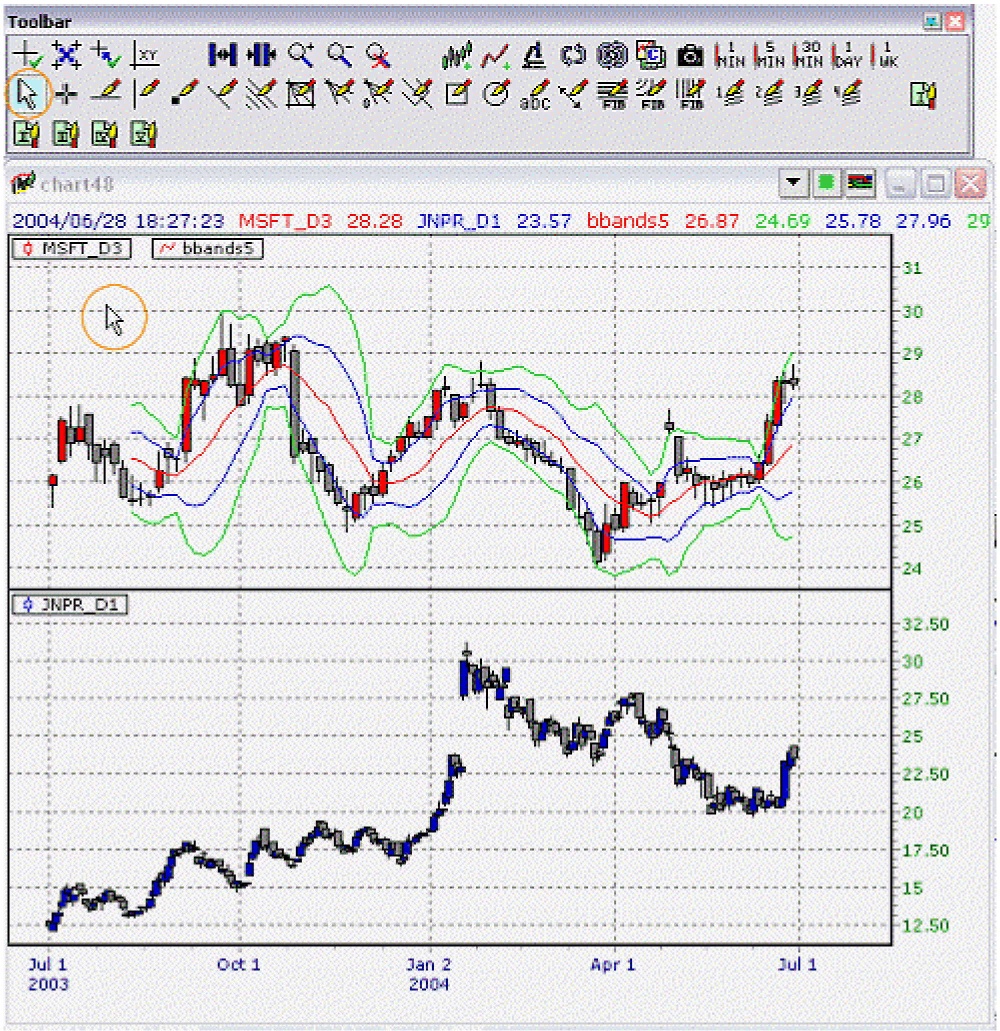
3 By default, each of the custom drawing tools are assigned with a custom drawing script already. To change the assignment, click the **Define** button right underneath the icon you want to make a new assignment.

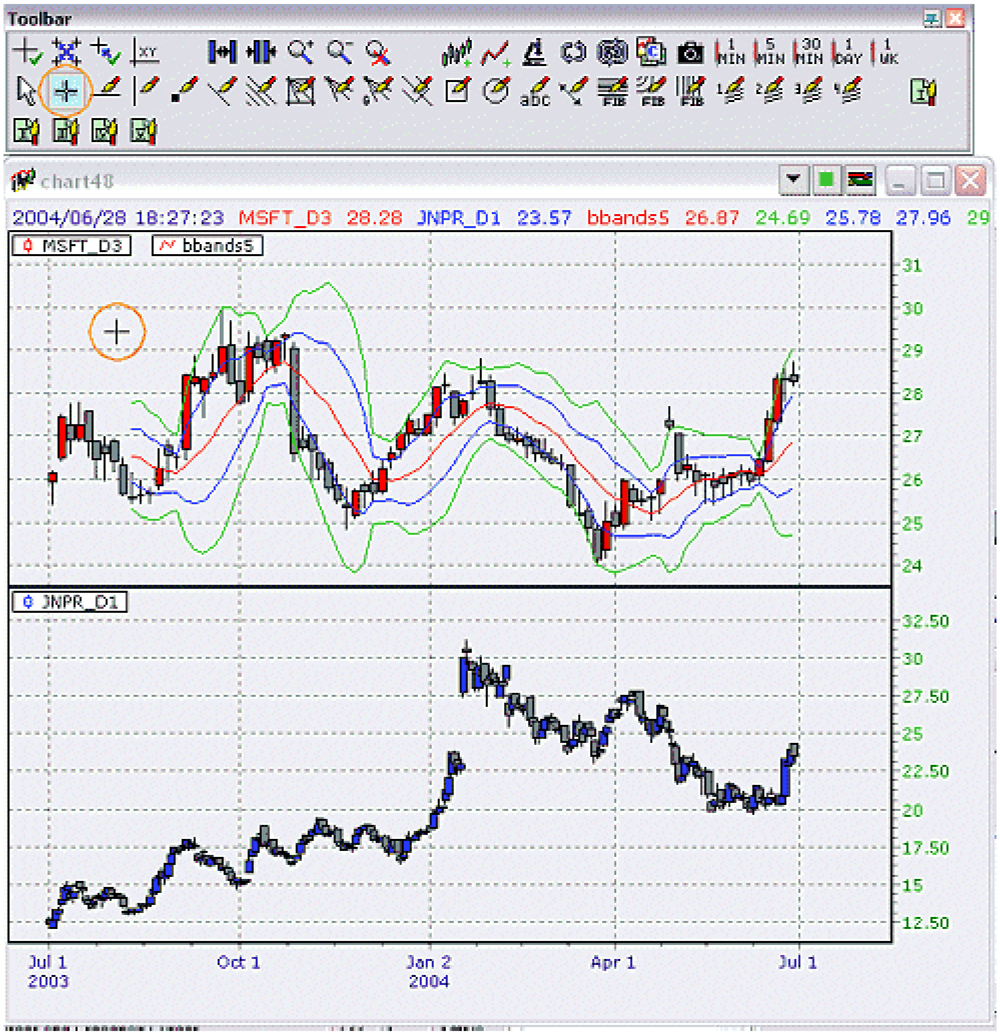
4 Choose from one of the available scripts listed in the `CustomDrawingTool` directory in the NeoTicker® installation and your new setting will be saved as part of your preference.

To learn more about Custom Drawing Tools, please refer to the ***Custom Drawing Tools*** (see "Custom Drawing Tool" on page 1655).

Pointer Cursor and Crosshair Cursor

There are two style of cursors you can use in a chart: a pointer cursor and a crosshair cursor.





You can use the cursor style that you feel most comfortable with. You can press the TAB key on the keyboard to switch between different combination of cursor and crosshair: pointer cursor, pointer cursor with crosshair, crosshair cursor, crosshair cursor with crosshair.

Cursors also affect objects selection. See ***Data Series, Indicators and Drawing Objects Selection*** (see "Data Series, Indicator and Drawing Tool Selection" on page 1025) for details.

Pop up Menu

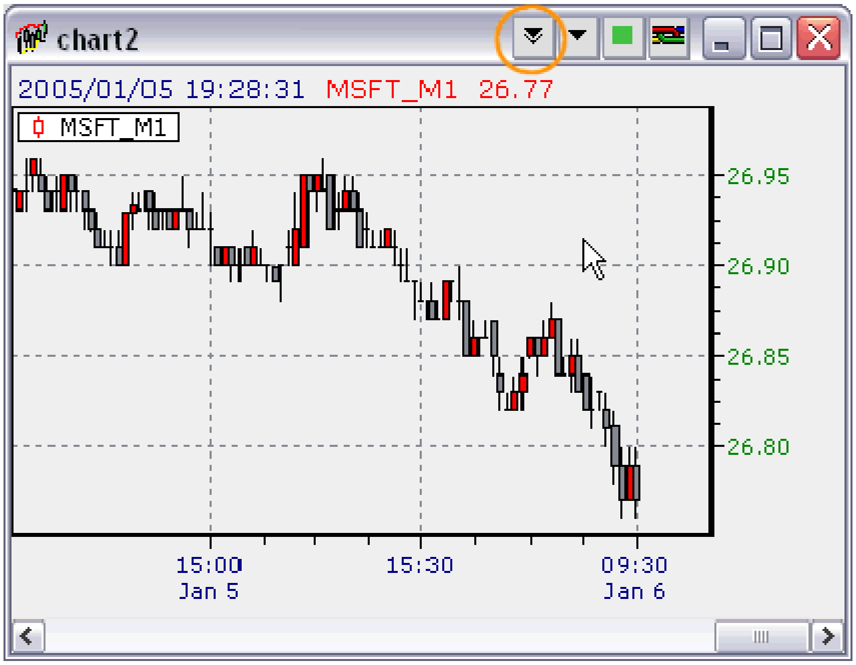
Pop up menu provides access to most chart functions. To bring out the pop up menu, right click the mouse.

The functions of the pop up menu depend on object selection. This avoids making the pop up menu overly large.

See ***Pop up Menu Reference*** (on page 1217) for more information.

Quick Jump Button

Quick Jump button ▼ on a chart's caption bar provides short cuts to frequently used commands.



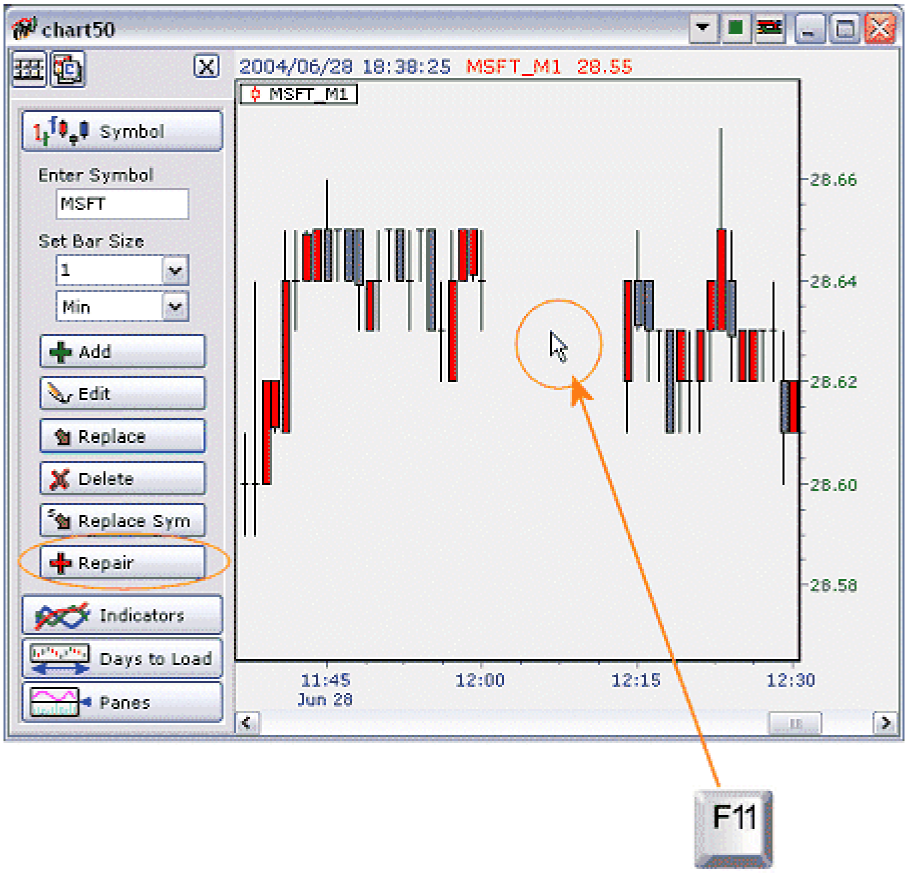
Examples:

- Opening Chart Manager
- Adjusting time frame
- Replaying ticks
- Reloading/refreshing chart

Repairing Data

This feature is useful only if you use Internet data feed that has a historical database (i.e. Quote.com, eSignal, myTrack).

During real-time trading, your data can be affected by many factors (outage of data vendor, your ISP, power, etc.) and resulting in a gaps, holes, and bad ticks in the charts.



You can use the F11 key to instruct NeoTicker® to automatically repair the holes in intraday data (minutes, seconds and ticks). Data repairing is the automation of the following steps:

- deleting the disk cache where the hole appears
- reloading the data from the data vendor
- recalculating all related indicators.

To use F11 to repair bad ticks, the data vendor must have already fixed the bad tick problem in their historical database. The time it takes varies from vendor to vendor. Data vendors typically correct bad ticks when they receive a report from users. So it will take more than a few minutes before bad ticks are corrected.

In addition, you can use the F12 key to auto repair all charts across all groups in NeoTicker®.

Resizing Chart

Normally when you resize a chart window (by dragging the window border with mouse), the number of bars in the chart remain unchanged. This has the effect that the space between bars will change.

If you want to retain the space between bars, hold both left and right mouse buttons when you drag the window border. This will preserve the chart spacing, i.e. when you size up the chart, more bars will be shown.

Space Bar Short Cut

If you press the space bar, you will bring up a window to let you quickly edit the chart. The behavior of the space bar depends on your selection and whether the chart is blank. Here are the rules:

- If a data series, an indicator or a drawing tool is selected, pressing the space bar will bring out the editor for these objects
- If nothing is selected, pressing the space bar will bring out the editor for the first data series
- If the chart has no data series (i.e. a blank chart), pressing the space bar will let you add a data series to the chart

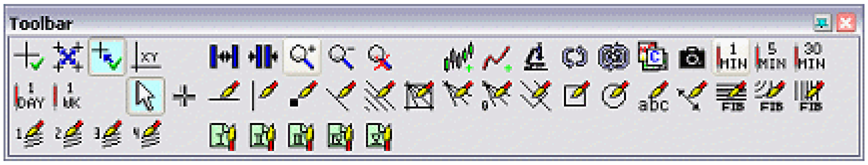
Taking Screenshot

You can take snapshot of the chart by either:

- Choose one of the **Export** option under the pop up menu,
- Press the key CTRL-F11
- Press the camera button 📷 on the tool bar.

Tool Bar

The tool bar buttons provides buttons for the commonly used functions in charts. If you don't see the tool bar, press F4 or use the user panel's tool bar button ▦ to open the tool bar. The tool bar looks like:



Like many windows in NeoTicker®, you can resize and reposition the tool bar window to your liking.

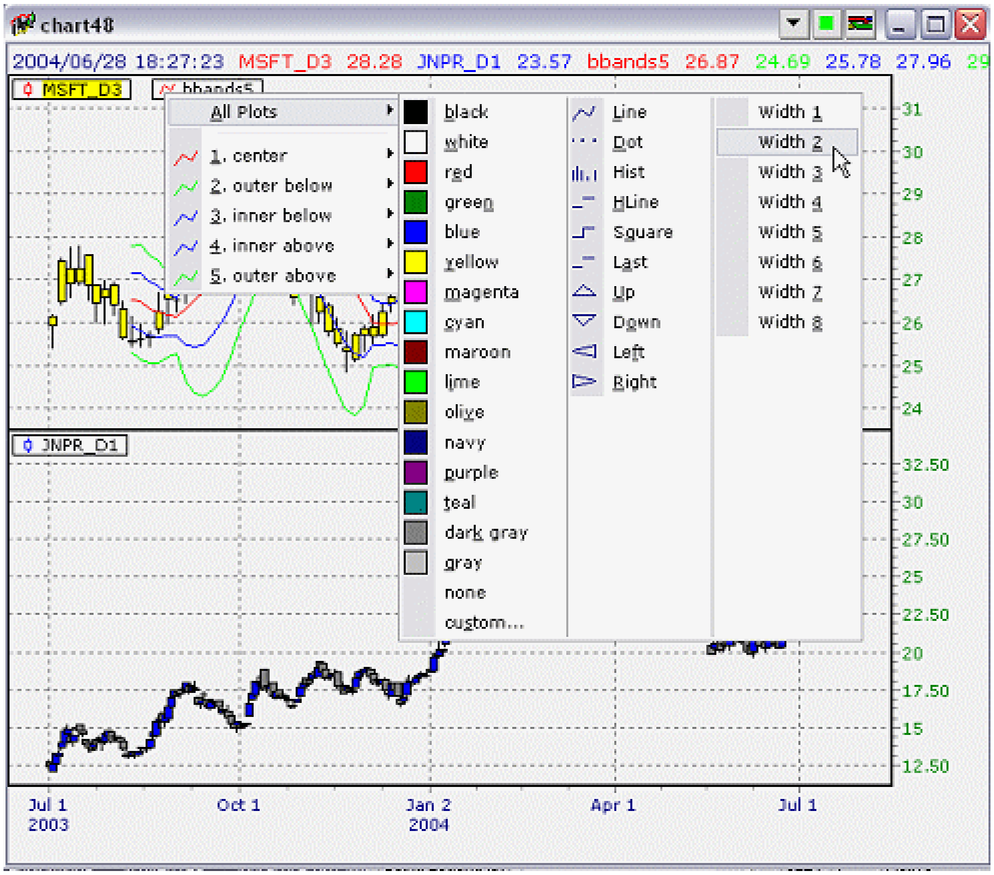
The pin button on the caption controls the behavior of the tool bar after you press a button. If the pin is up 📌, after pressing a button (except the zoom buttons), the tool bar will be closed. If the pin is down 📌, after pressing a button, the tool bar will remain open.

You can configure the visibility of the buttons on the tool bar. NeoTicker® also supports multiple tool bars. To find out more, go to ***Customizing the Tool Bar*** (on page 1209).

Using Legend Icon to Change Visual Appearance

You can set the visual appearance of data series and indicators using their legend icon. Simply click on the legend icon and choose the color, width or style.

Make sure you are clicking the graphics icon, not the description (e.g. MSFT_D1, bbands3) or you will be selecting the item instead.

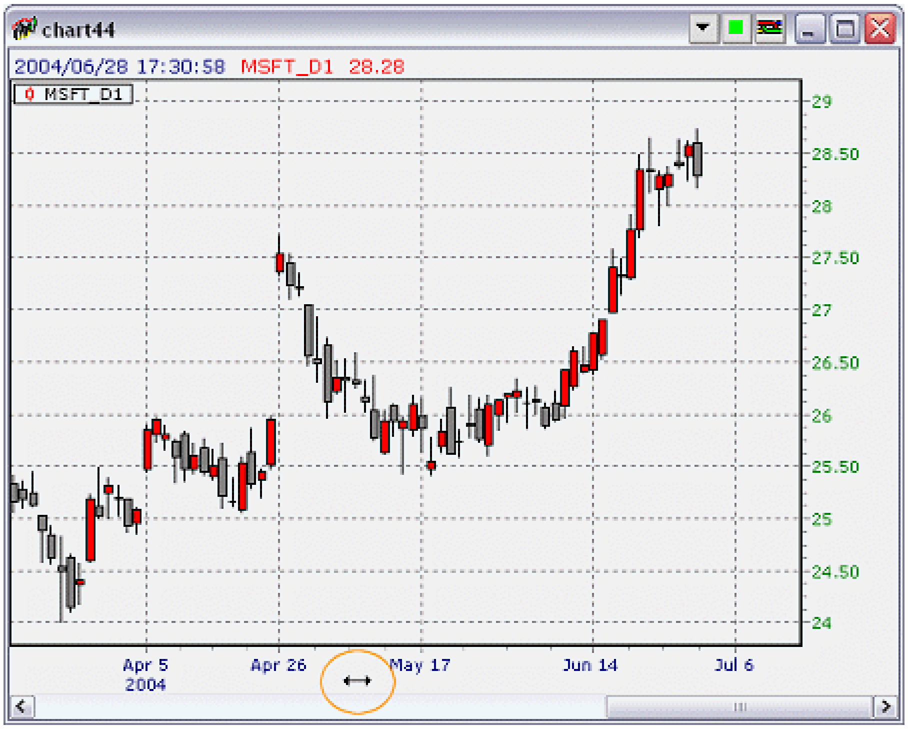


Zooming

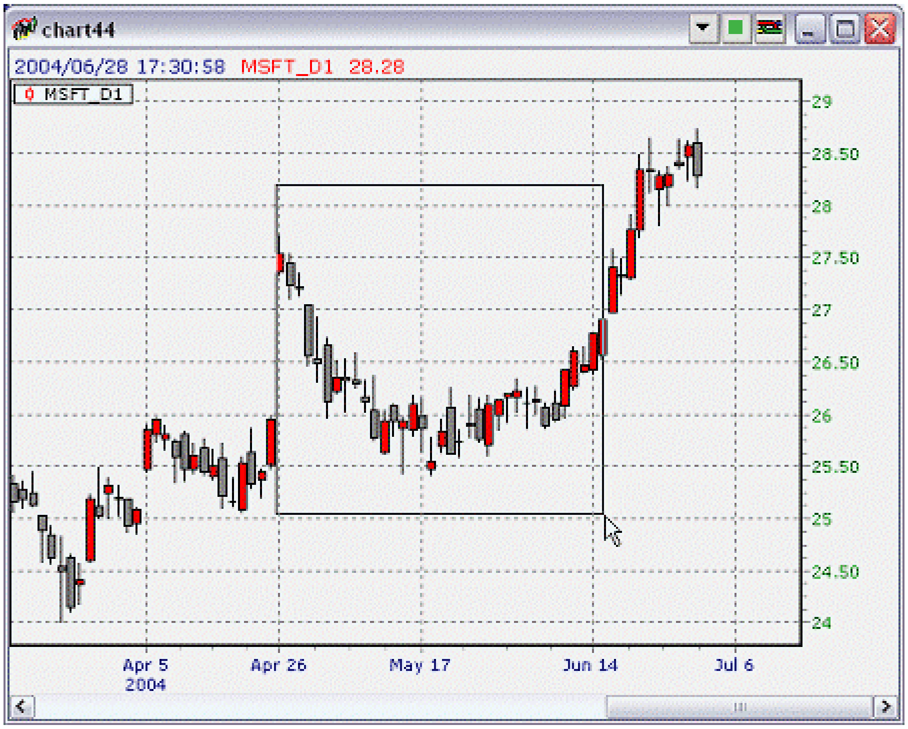
Zoom in and Zoom out

Zoom in to a chart to see more details. Zoom out to a chart to see more bars. You can zoom in/out by:

- Pressing the zoom in and zoom out buttons in the tool bar. Pressing the + and - keys on the keyboard performs the equivalent actions.
- Pressing the more bar space and less bar space buttons in the tool bar. The zooming is at 1/20 of the regular zoom in/out level. The holding the SHIFT key while pressing the + and - keys on the keyboard performs the equivalent actions.
- Hold the SHIFT key and drag the time axis to zoom interactively.



- Click on an empty region of the chart, drag the mouse to lower right. This will zoom precisely in to a region. There is no equivalent zoom out action.



Un-zoom to Default Zoom Level

Time chart maintains a default zoom level. The default zoom level is calculated automatically to give you a good view of the data series. When you un-zoom, the chart returns to its default zoom level. To un-zoom:

- Click on an empty region of the chart, drag an click the mouse to the upper left
- Press the un-zoom button in the tool bar.
- Press Ctrl-Z on the keyboard.

Working with Panes

It makes sense to plot a data series and its moving average together since their values are close. However, this is not always the case. Often times you may want to plot the data series for another symbol or create an indicator that has values very different from the data series (e.g. stock price and volume).

This problem can be solved by using overlay (see ***Overlay*** (on page 1101)).

Another solution NeoTicker® offers is pane. A pane is a separate section in the chart that shares the same time axis with other panes, but it has its own value axis. Panes provide extra control and flexibility over overlay.

It is possible to control panes with user panel. See ***Using the User Panel in Time Charts*** (on page 1055)

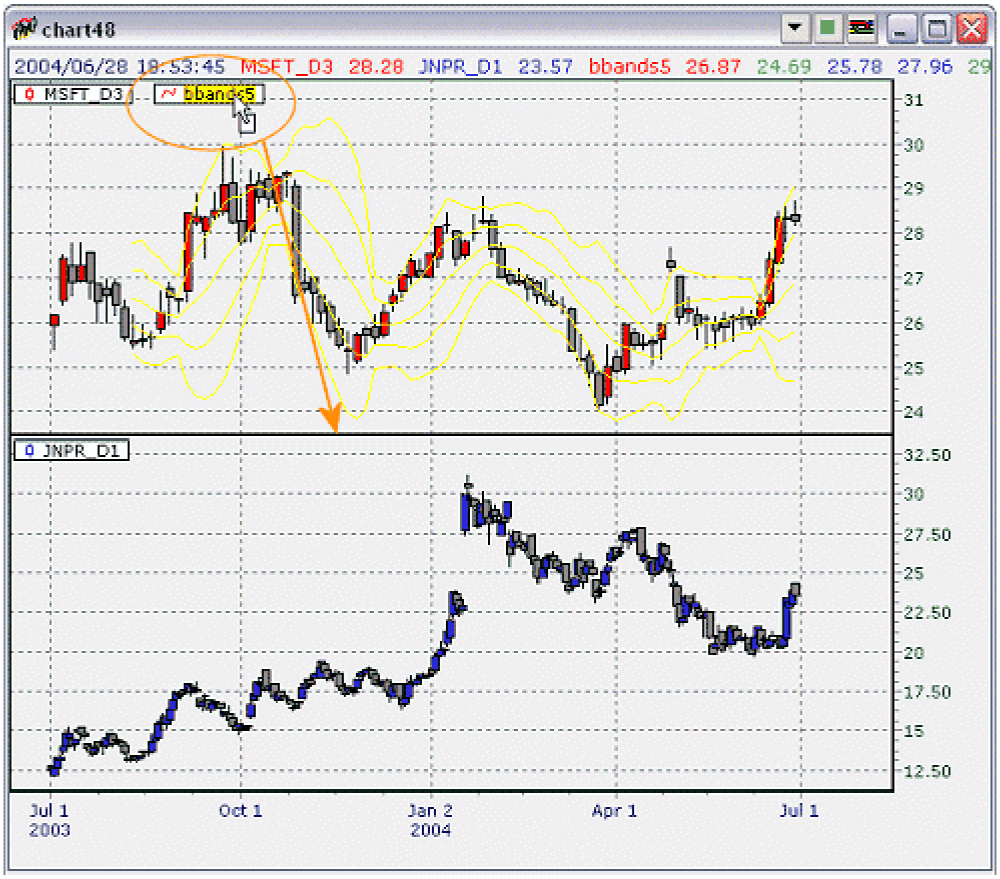
Adding New Pane

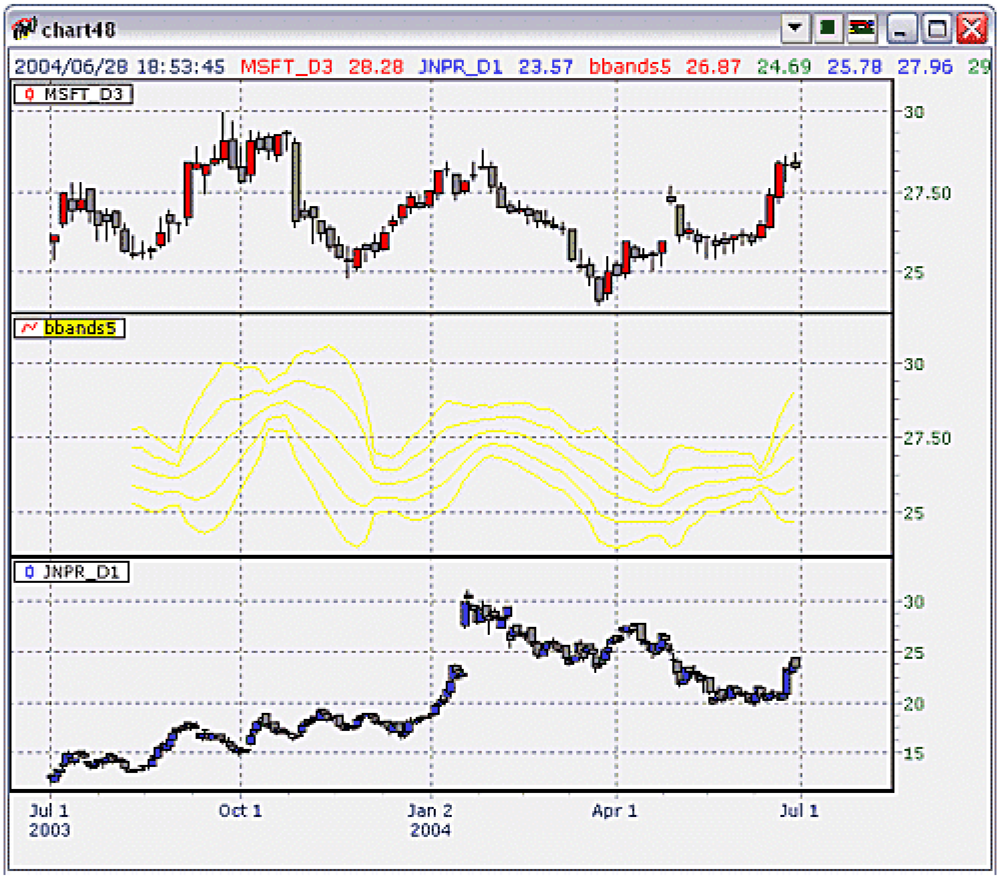
To add a pane, first un-select all data series and indicators, then choose **Pane/Pane Axis>Add** from the pop up menu.

Moving Data Series and Indicators between Panes

You can move data series and indicator between panes.

You can edit the pane setting in by selecting the data series or indicator, and change the pane setting. Or simply drag and drop data series and indicators between panes, as illustrated in the following figures.

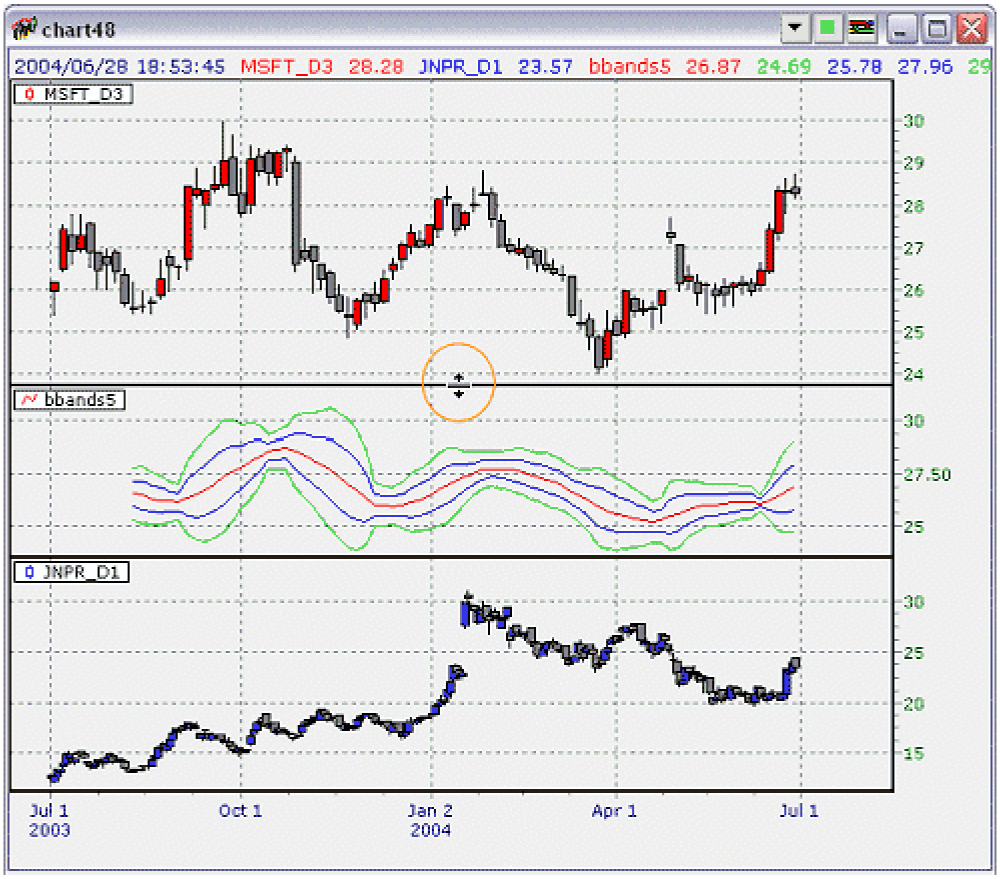




When you move an object from a pane, if you drop it on the edge (the heavy line that separates two panes) between panes, the chart will create a new pane with the object inside the new pane.

Resizing Pane

You can resize panes by dragging the edge between panes, as illustrated in the following figure.



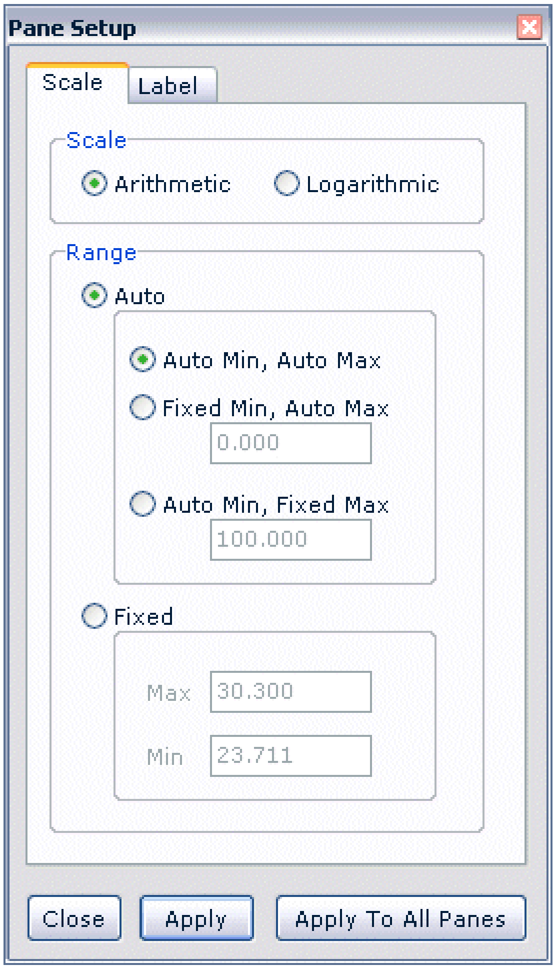
Editing Panes

There are pane properties you can set. For example, scaling, range, etc.

To edit a pane:

- Double click in the price axis area of the pane, or
- Un-select everything, right click on pane you want to edit to open the pop up menu, and choose **Pane/Price Axis>Edit**.

A window will be open to let you edit the pane.



Scale

● Arithmetic ○ Logarithmic

Range

● Auto

● Auto Min, Auto Max

○ Fixed Min, Auto Max 0.000

○ Auto Min, Fixed Max 100.000

○ Fixed

Max 30.300

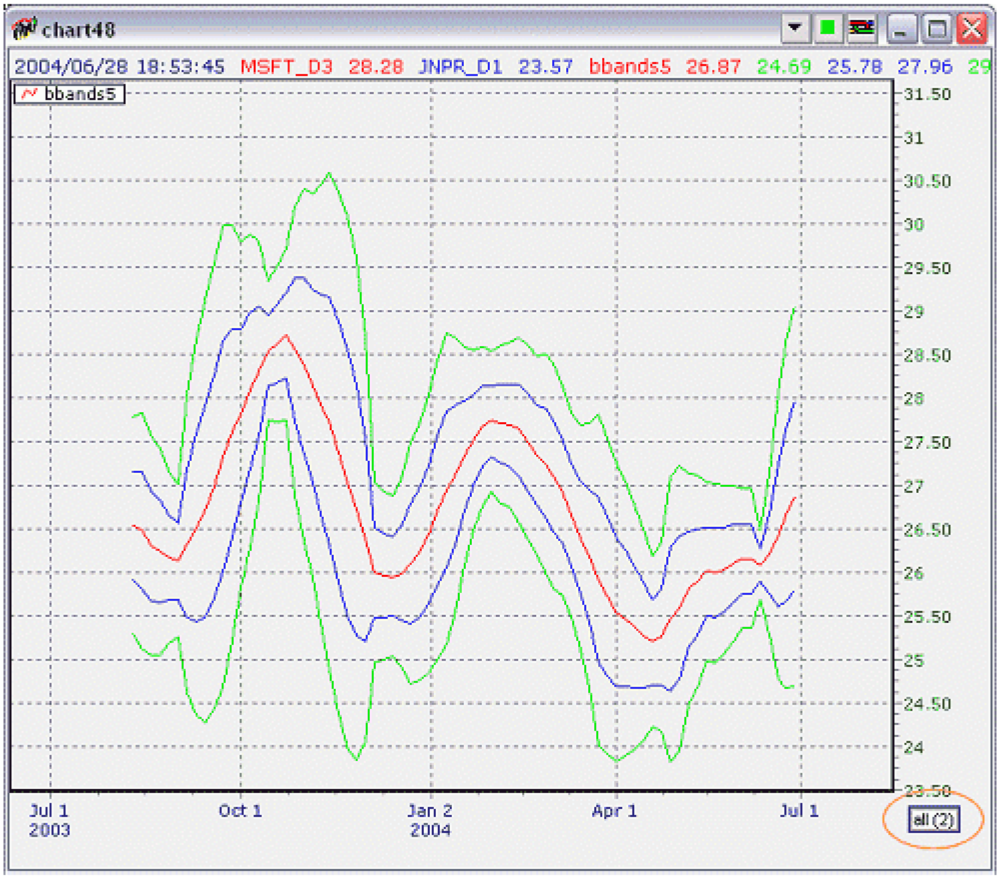
Min 23.711

Pane range can be adjusted directly by dragging on the price axis. If you hold the SHIFT key while dragging, you will resize the price axis. If you have directly manipulated the price axis, auto scaling of the pane will be turn off. You can turn auto scaling back on by pressing the **scale** button.

Single Pane Mode

Single pane mode lets you expand a pane in a chart to fill the whole chart. This is a convenient feature for users who want to navigate through panes in a chart quickly. To turn on single pane mode, simply double click on any empty region in a chart.

When in single pane mode, you can perform all chart operations as usual, except operations that are pane related, e.g. deleting a pane.

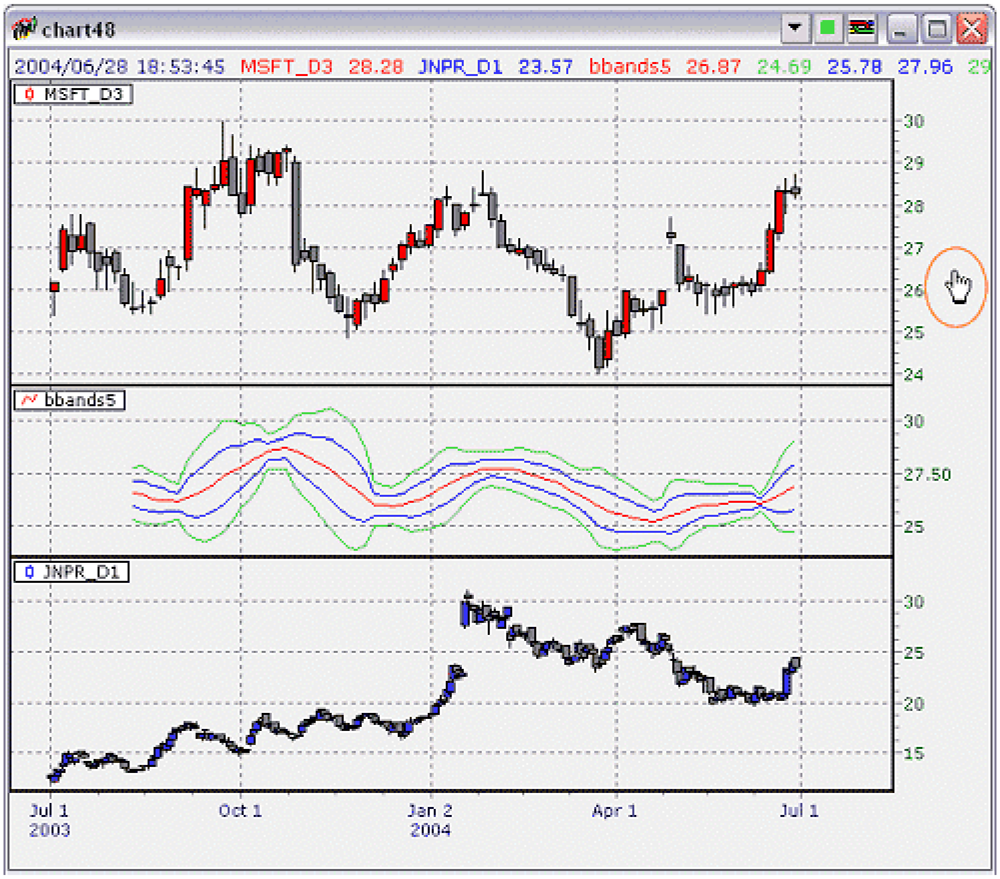


To switch to another pane, double click again on an empty region.

There is an **all** button in the lower right corner of the chart. The number in the bracket is the current pane that is displayed. Pressing the **all** button will switch you back to all pane mode.

Change Price Axis Scaling and Labelling

You can pan the price directly by dragging on the price axis. If you hold the SHIFT-key while dragging, you will scale the chart.



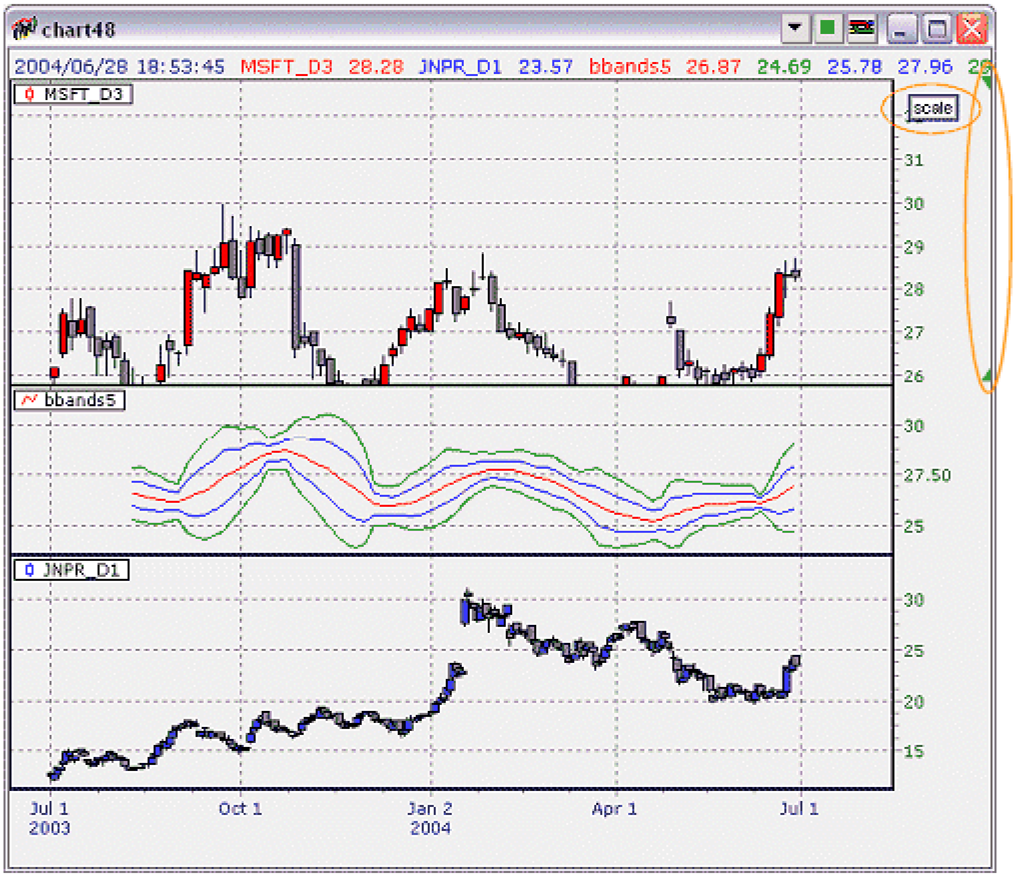


chart48
2004/06/28 18:53:45 MSFT_D3 28.28 JNPR_D1 23.57 bbands5 26.87 24.69 25.78 27.96 28
MSFT_D3
scale
31
30
29
28
27
26
bbands5
30
27.50
25
JNPR_D1
30
25
20
15
Jul 1
2003
Oct 1
Jan 2
2004
Apr 1
Jul 1

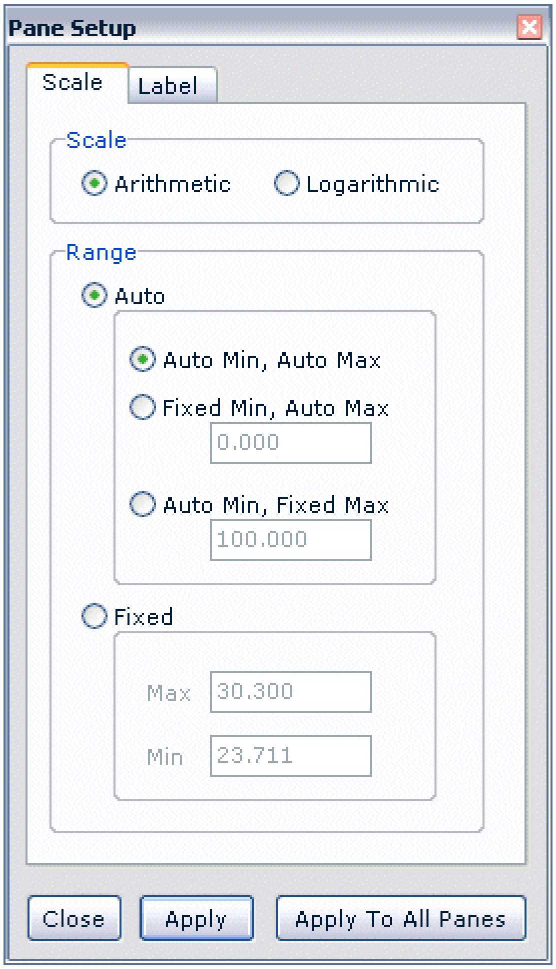
After you have pan or scale the price, the chart will turn off auto scaling in order to work with the new scale. If you want to turn auto scaling back on, press the **scale** button.

To tell a pane is scaled, observe the **scale** button, and the triangular scale markers in the pane.

The scale button can be hidden. See ***Hiding the Scale Button in Price Axis*** (see "Scale Button in Price Axis" on page 1082).

You can specify exact scaling parameters by double clicking on the price axis or by choosing **Edit Pane** from the pop up menu, and edit the settings in a dialog.

Scale Tab



Pane Setup

Scale

● Arithmetic ○ Logarithmic

Range

● Auto

● Auto Min, Auto Max

○ Fixed Min, Auto Max 0.000

○ Auto Min, Fixed Max 100.000

○ Fixed

Max 30.300

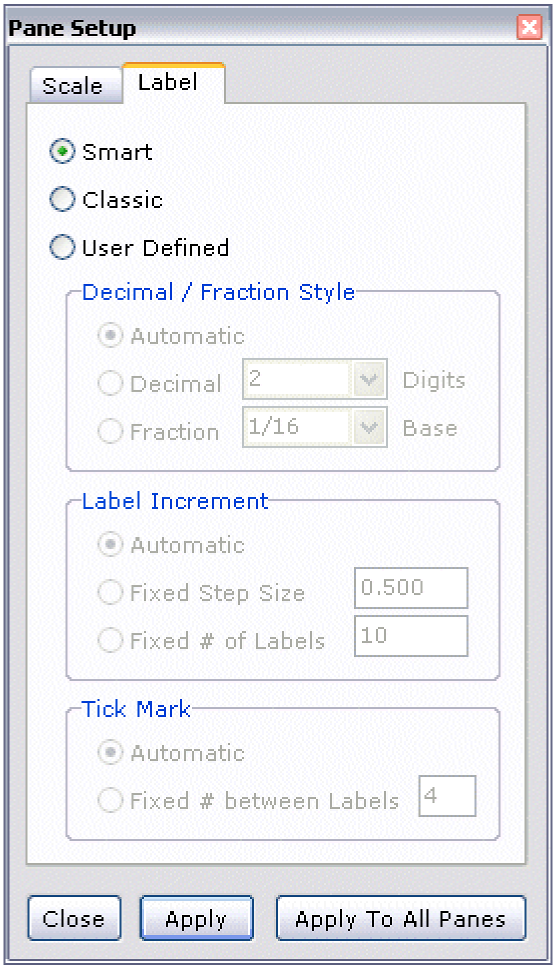
Min 23.711

This tab provides the scaling options for the price axis.

Setting	Meaning
Scale	Whether the scale is arithmetic (linear) or logarithmic.
Range / Auto	Provide automatic scaling of the price axis as the price changes. You have the option to fix the minimum or maximum value.
Range / Fixed	Manually fixed the minimum and maximum value of the price axis.

A common usage is to use **Auto / Auto Max, Fixed Min** and set the value to 0 for panes that contain histogram such as volume data. This setting will make the histogram flushed with bottom axis.

Label Tab



Pane Setup

Scale | Label

● Smart
○ Classic
○ User Defined

Decimal / Fraction Style
● Automatic
○ Decimal 2 Digits
○ Fraction 1/16 Base

Label Increment
● Automatic
○ Fixed Step Size 0.500
○ Fixed # of Labels 10

Tick Mark
● Automatic
○ Fixed # between Labels 4

Under this tab you can change the way the axis labels are displayed.

Setting	Meaning
Smart	Fully automatic. Time chart decides the decimal place and significant digits for you. This is the default.
Classic	Fully automatic. This mode is to provide backward compatibility to charts saved in NeoTicker® 2.3 and earlier. New charts should use **Smart** mode instead.
User Defined	This mode provides full manual settings.

The rest of the settings is available only in **User Defined** mode.

Decimal/Fraction Setting	Meaning
Automatic	Time chart decides the decimal place and significant digits for you.
Decimal	You will specify the decimal places
Fraction	You will specify the base of fraction. Fraction base is not displayed in the axis in the chart. For example, '16 15/32' is displayed as '16^15' in the axis. This is to save horizontal space.

Label Increment Setting	Meaning
Automatic	Time chart decides the increment between labels.
Fixed Step Size	You will specify the number of labels between labels.
Fixed # of fLabels	You will specify a fixed step size. IMPORTANT: if you specify a step size that is not compatible with the decimal/fraction style, time chart will consider the decimal/fraction style has precedence over the step size. The resulting axis labeling can be confusing
Automatic	Time chart decides the decimal place and significant digits for you.

Tick Mark Setting	Meaning
Automatic	Time chart decides the number of tick marks between two labels.
Fixed # between Labels	You will specify the number of ticks between labels.

Using the User Panel in Time Charts

In this section, we will examine the functionality of time chart's user panel in more details.

Prerequisite

You should know how to create charts.

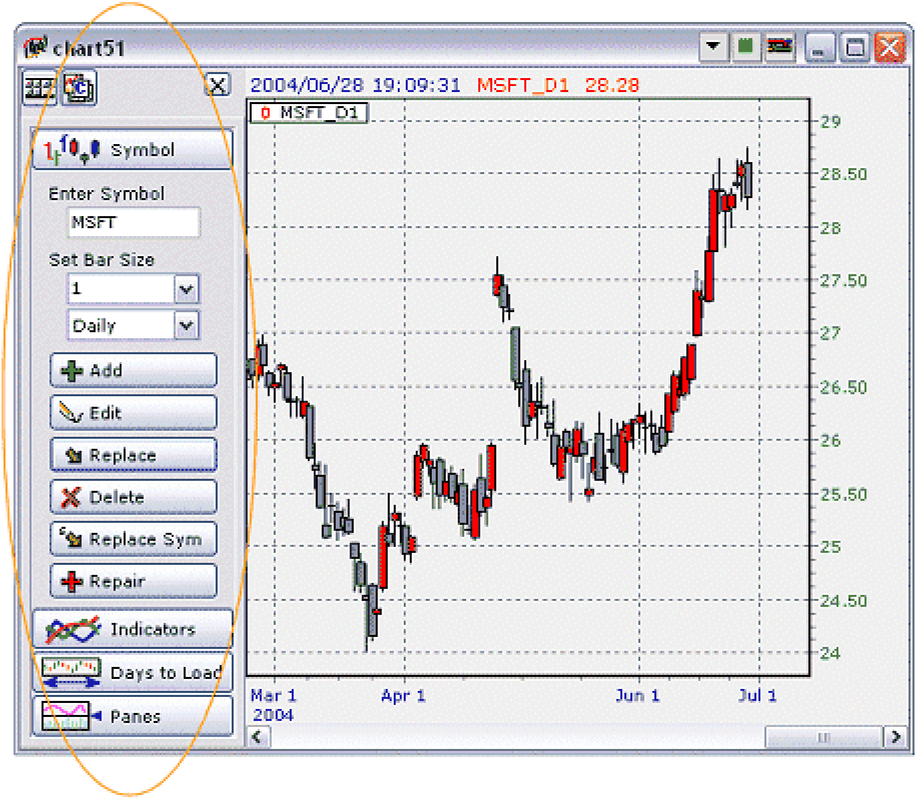
Goal

After reading this section, you will be able to:

- Add multiple data series to a chart
- Repairing damaged data
- Add indicators to a chart
- Add indicators on indicators
- Configuring number of days to load in a chart
- Managing panes
- Opening the tool bar and chart manager
- Customize user panel behavior
- Understand the role and limitation of user panel in a chart

Understanding User Panel

NeoTicker® provides a user panel in time chart to help users to quickly perform chart operations. The following figure is a chart with the user panel on.



Enter Symbol
MSFT

Set Bar Size
1
Daily

2004/06/28 19:09:31 MSFT_D1 28.28

User panel captures many of time charts' features. It is especially handy for adding data series and indicators to a chart. To keep user panel easy to use, user panel does not contain the more advance features. For example, if you want to change the data source, tick filter for a data series, you need to edit the data series using the data series dialog. Similarly, if you to change the links of an indicators, you need to edit the indicator using the indicator dialog. For more information, see ***Working without the User Panel*** (on page 1070).

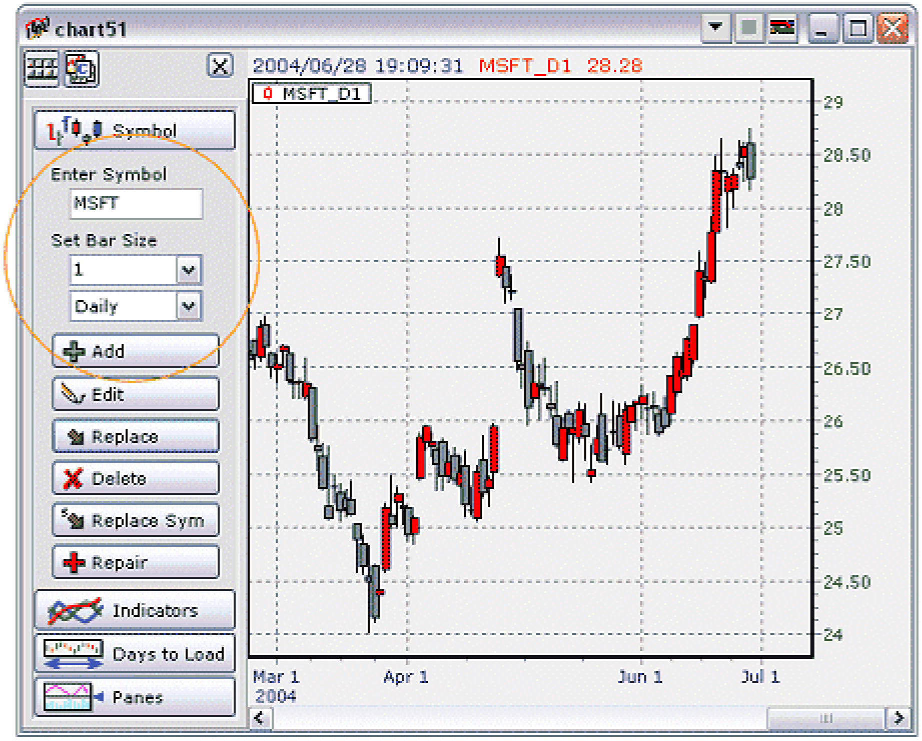
Hiding and Showing User Panel

To show more chart space, you can hide the user panel. To hide, simply click the close button ☒ on the user panel.

To show the user panel, press ESC to un-select everything in the chart, then right click on the chart to open the pop up menu, choose **User Panel**.

Adding and Replacing Data Series

Open the symbol tab by clicking on **Symbol** on the user panel. You can work on data series using this panel. Data series is sometimes refer to as charts in other software. The reason NeoTicker® makes a distinction is NeoTicker® can plot multiple data in the same window. Calling data series as charts can be confusing.

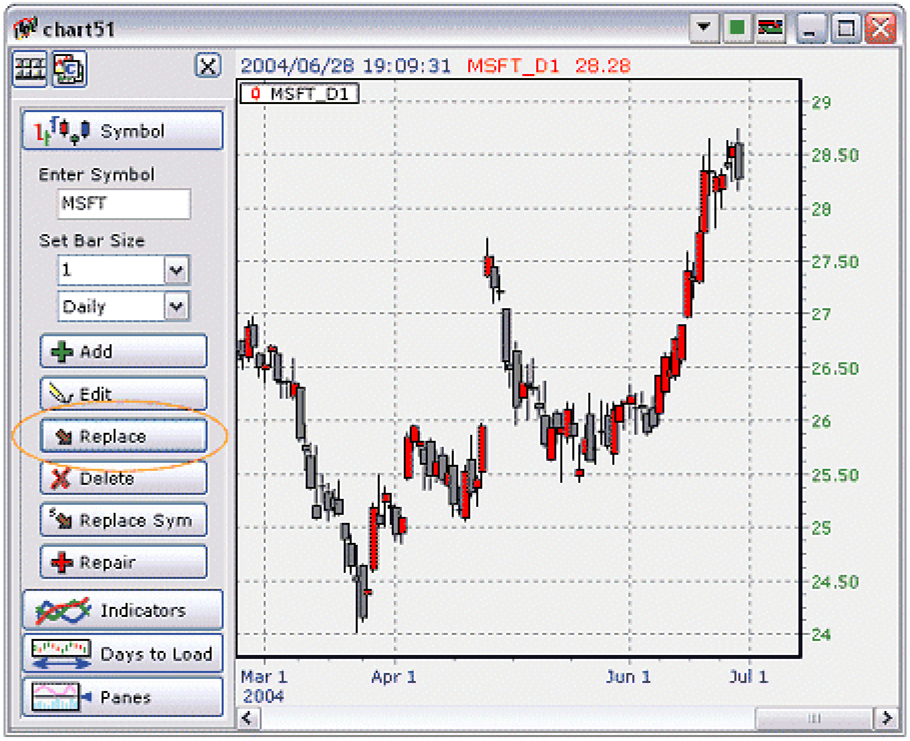


Enter Symbol: MSFT

Set Bar Size: 1, Daily

To add a data series, simply enter the symbol name, bar size and time frame information, and then press the **Add** button or press the ENTER key. Notice that for the bar size, you are not limited to some fixed values. You can use any values like 3 to construct 3-minute bars.

If you want to look at another symbol, you can enter the new symbol into the user panel and press the **Replace** or **Repl Sym** button. The **Replace** button will replace the symbol, number of bars and time frame. The **Repl Sym** button only replace the symbol, but keep the bar size and time frame information intact.



2004/06/28 19:09:31 MSFT_D1 28.28

Enter Symbol: MSFT

Set Bar Size: 1, Daily

Adding Multiple Data Series of Different Time Frames

NeoTicker® allows you to add multiple data series to a time chart window. Having multiple data series on the same chart allows you perform advanced analytic on these data, such as spread calculation and user defined weighted index.

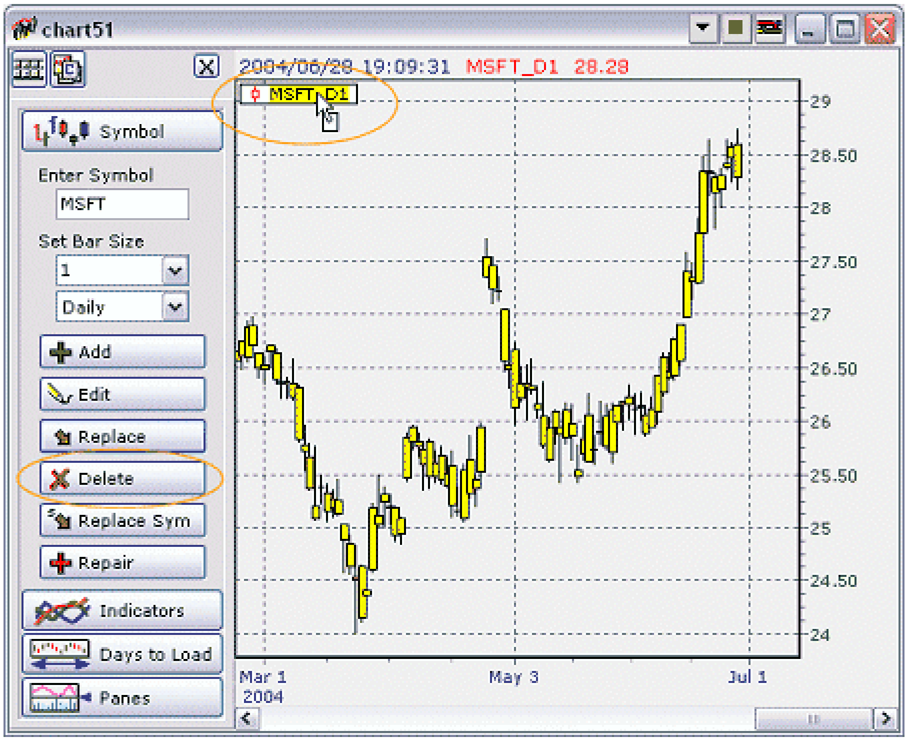
To add another data series, simply enter the symbol, number of bars and time frame information and press the **Add** button. Note that the number of bars and time frame information can be different from other data series.

One of the most powerful feature of NeoTicker®'s time chart is its ability to mix data series of different time frames into the same chart, even in the same pane.

For example, you can mix a 1-day daily chart and a 3-day daily chart. Time chart window will correctly draws all bars at the right time. The following figure is a time chart that shows MSFT in 1-day and 3-day daily time frames.

Deleting a Data Series

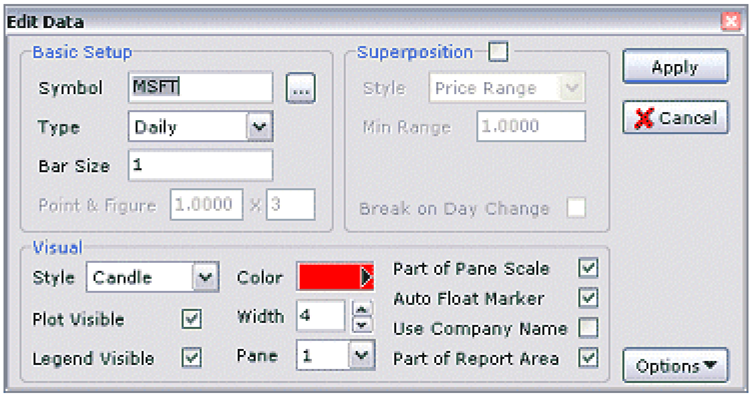
To delete a data series, you must first select it because otherwise NeoTicker® does not know which data series you intend to delete. To select a data series, click on the data series itself or the data series' legend.



Then press the **Delete** button or the DELETE key on the keyboard to delete the data series.

Editing a Data Series

Editing a data series allows you to fine tune the behavior of a data series. Press the **Edit** button or the space bar on the keyboard. The Edit Data dialog will be opened.



Basic Setup
Symbol MSFT
Type Daily
Bar Size 1
Point & Figure 1.0000 X 3

Superposition □
Style Price Range
Min Range 1.0000
Break on Day Change □

Visual
Style Candle
Color 🟥
Width 4
Pane 1
Plot Visible ☑
Legend Visible ☑
Part of Pane Scale ☑
Auto Float Marker ☑
Use Company Name □
Part of Report Area ☑

You can change simple settings such as color and width to advance settings such as scaling and tick filtering for the data series. To find out how to use the Edit Data window, refer to ***Adding/Editing Data Series*** (on page 1089).

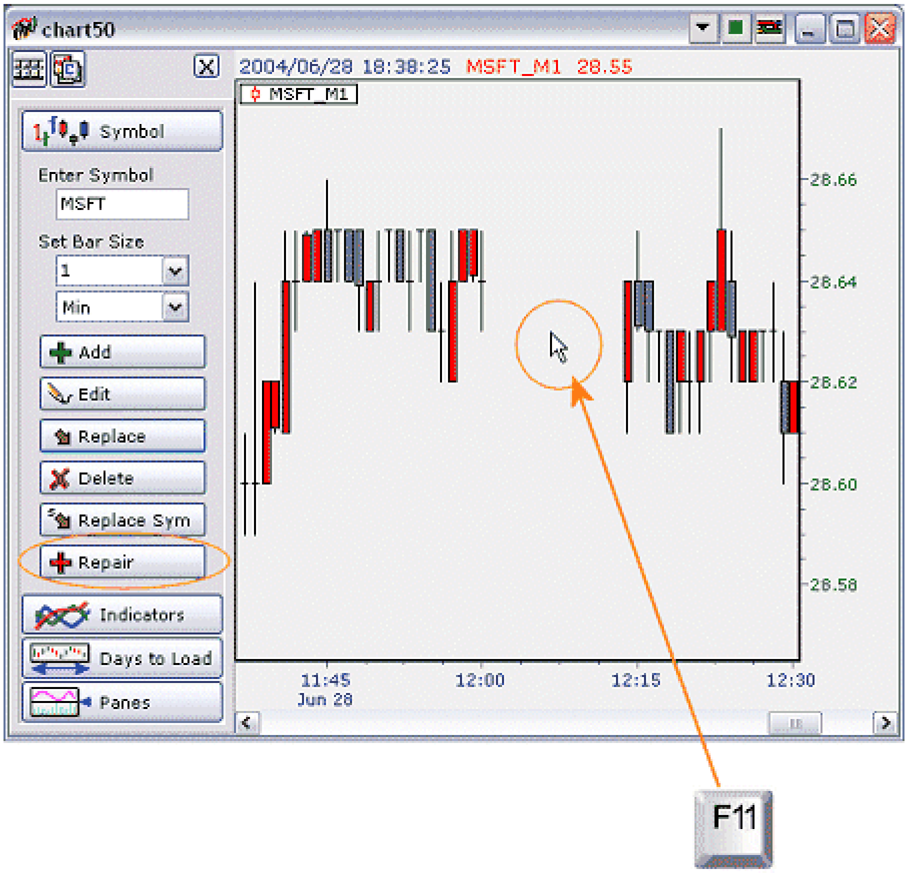
The visual appearance of data series can be changed directly with the legend. See ***Using Legend to Change Visual Appearance*** (see "Using Legend Icon to Change Visual Appearance" on page 1040).

If you have more than one data series in the chart and want to edit a specific data series, select the data series first by clicking on its legend before pressing the **Edit** button.

Repairing a Data Series

Repairing only works with Internet data feed with an historical data server such as NAQ, Quote.com, eSignal and myTrack. What repairing does is deleting the cached data from your hard drive and reloading the data from the data vendor.

NeoTicker® allows you repair a data series due to server outage. Damaged data series appears with holes in the data.



To repair a data series, select the data series and press the **Repair** button. The Repair Data Wizard will be opened and simply follow the directions on the wizard.

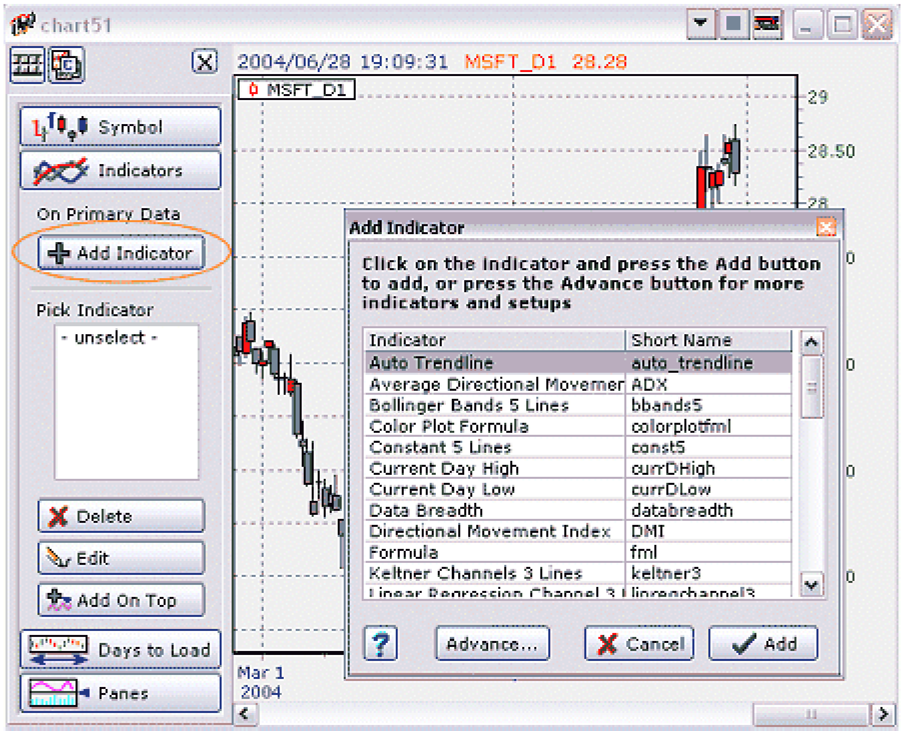
Alternatively, you can use the F11 key to automatically repair a data series without the wizard. If you use F11, NeoTicker® will guess the data and the data type to be repaired.

Important: Repairing does not always work because the data holes can be originated from the data vendor. In this case, the holes cannot be repaired. The feature is useful for recent data because it is most likely the data is available. For older data, your data vendor may no longer stores the data you requested.

Adding an Indicator

To work on indicators (also known as studies in some other software), press on the **Indicator** tab.

To add an indicator, press on the Add Indicator button. An indicator window will be open to let you choose among the commonly used indicators.

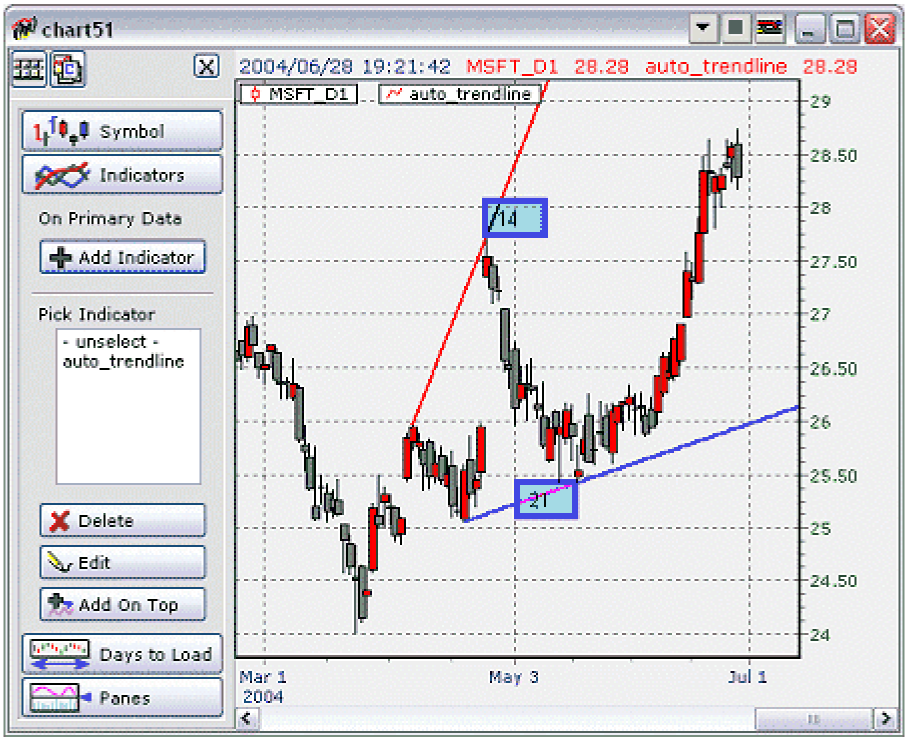


Click on the Indicator and press the Add button to add, or press the Advance button for more indicators and setups

Indicator	Short Name
Auto Trendline	auto_trendline
Average Directional Movemer	ADX
Bollinger Bands 5 Lines	bbands5
Color Plot Formula	colorplotfml
Constant 5 Lines	const5
Current Day High	currDHigh
Current Day Low	currDLow
Data Breadth	databreadth
Directional Movement Index	DMI
Formula	fml
Keltner Channels 3 Lines	keltner3
Linear Regression Channel 3 L	linregchannel3

If you need to fine tune the indicator or to choose from the full list of indicators (over a hundred indicators are available), press the **Advance** button.

Suppose you have multiple data series on the chart, how do you know which data series the indicator is applying to? To apply the indicator to a specific data series, select the data series first by clicking on its legend before pressing the **Add** Indicator button.



If you want to add another indicator, simply press the **Add** Indicator button again. There is no pre-set limit on the number of indicators you can add to a time chart.

Editing an Indicator

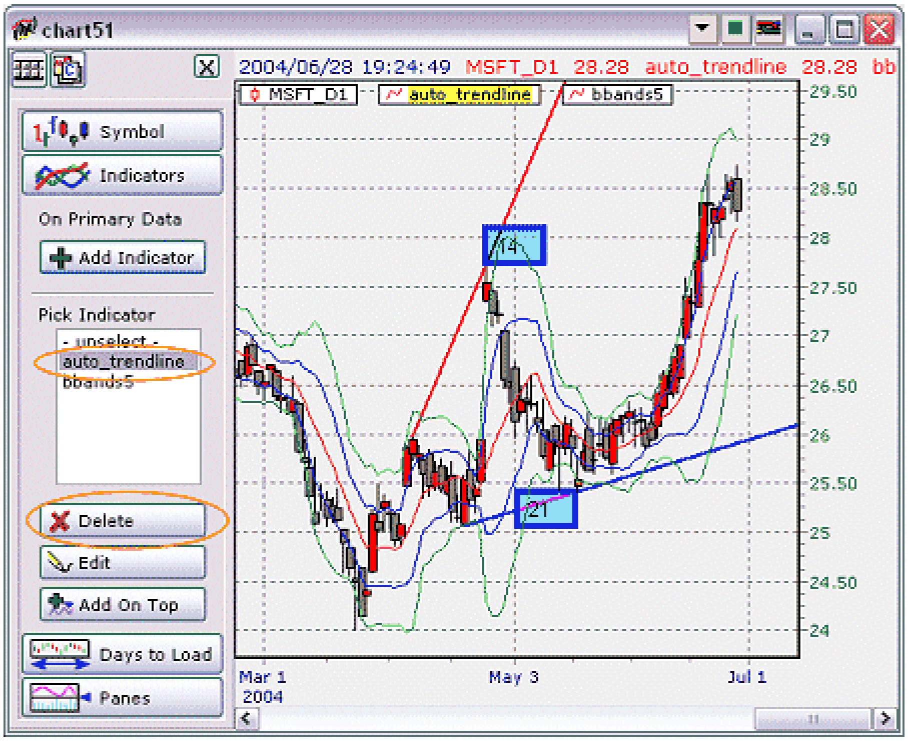
Press the **Edit** button in **indicator** in the user panel to edit an indicator.

You can edit the parameters and visual settings for the indicator. If you check the **full dialog** box, you can access additional settings such as indicator links.

The visual appearance of indicator can be changed directly with the legend. See ***Using Legend to Change Visual Appearance*** (see "Using Legend Icon to Change Visual Appearance" on page 1040).

If you have more than one indicator and want to edit a specific one, you need to first select the indicator. To select an indicator, you can simply click on the indicator's legend. You can also select an indicator inside user panel.

Deleting an Indicator



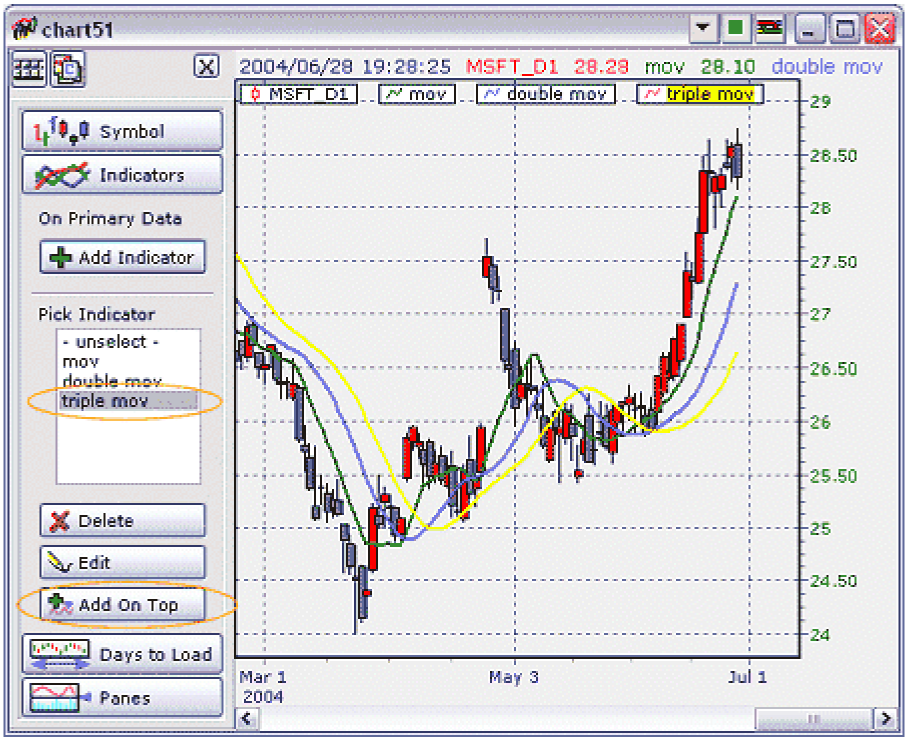
To delete an indicator, select the indicator and press the **Delete** button.

Indicator on Indicator

NeoTicker® allows you to apply an indicator on another indicator. For example, you can apply Bollinger Band on volume. Or you can apply moving average on RSI.

To add an indicator on another indicator, first select the source indicator, then press the **Indicator on Indicator** button. Adding an indicator to another indicator is no different from adding an indicator to a data, except the source is now an indicator, not a data series.

The following figure shows a data, with three levels of exponential moving average applied on it.



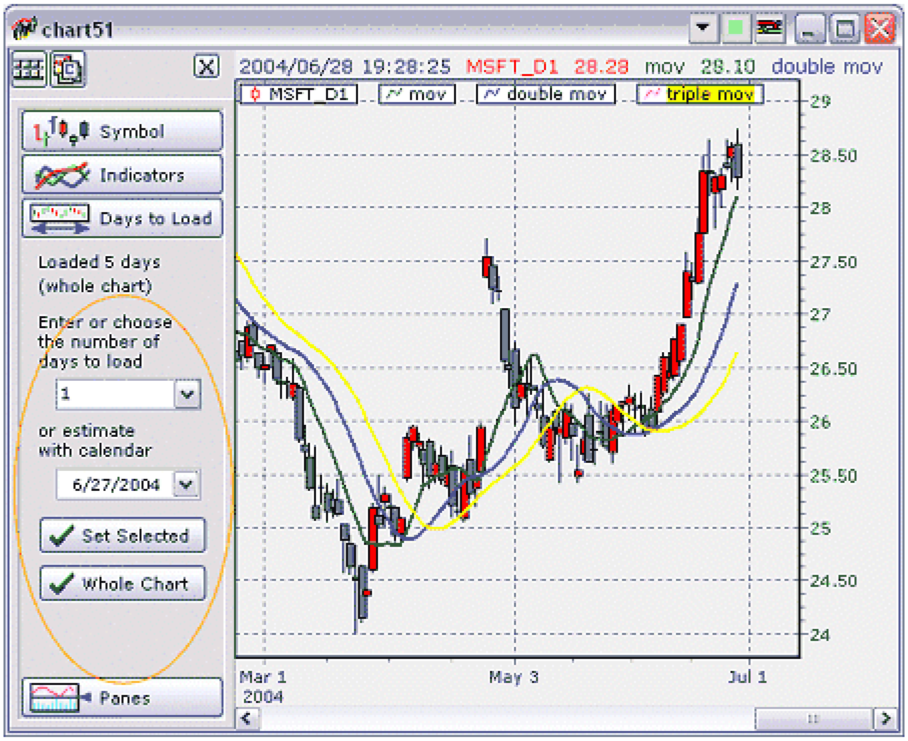
Configuring Days to Load

NeoTicker® lets you control the exact number of days of data to load in a chart. This is in contrast to some data-on-demand design of some charting software.

On the surface, data-on-demand seems to be more intuitive. However, data-on-demand severely restricts the class of indicators that can be implemented. The reason is simple. Suppose you have indicators that depends on long period of historical data. Data-on-demand software cannot calculate these indicators until you scroll the chart back enough, or worse, gives you a wrong calculation because the data is not yet available.

NeoTicker® on the other hand lets the user to have full control. NeoTicker® does a good job of estimating the days of data to load for a particular time frame and gives you the extra option of specifying exactly the number of days to load, for the whole chart, or for a specific data series.

Press on the **Days to Load** tab to configure the days to load.



2004/06/28 19:28:25 MSFT_D1 28.28 mov 28.10 double mov

Loaded 5 days
(whole chart)

Enter or choose the number of days to load: 1

or estimate with calendar: 6/27/2004

To configure the days to load for a data series, click on the data series, edit the the days to load, then press the **Set selected** button.

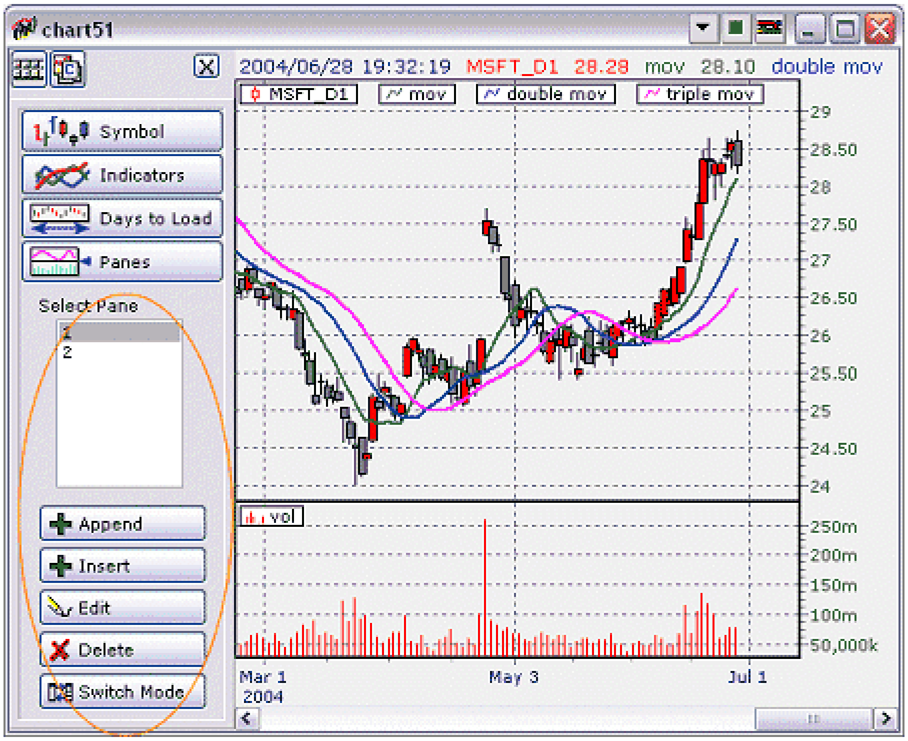
You usually do not have to worry about setting the days to load for the whole chart. The whole chart days to load is applied to data series that do not have an explicit days to load parameter set. For example, data series created by using the Add Data dialog. To configure the days to load for the whole chart, edit the days to load, then press the **Set whole chart** button.

Calendar Days vs. Trading Days

You can configure time chart to load calendar days or trading days, for more information, refer to ***Days of Data to Load*** (on page 1029).

Managing Panes (Vertical Segments)

Time chart is vertically segmented into panes in NeoTicker®. If you have added indicators such as volume to a chart, NeoTicker® may already have created panes for you.



You are free to drag data and indicators between panes. To drag and drop an item, simply press and hold the mouse button on to the legend of an item, move the mouse to the destination pane, and release the mouse button.

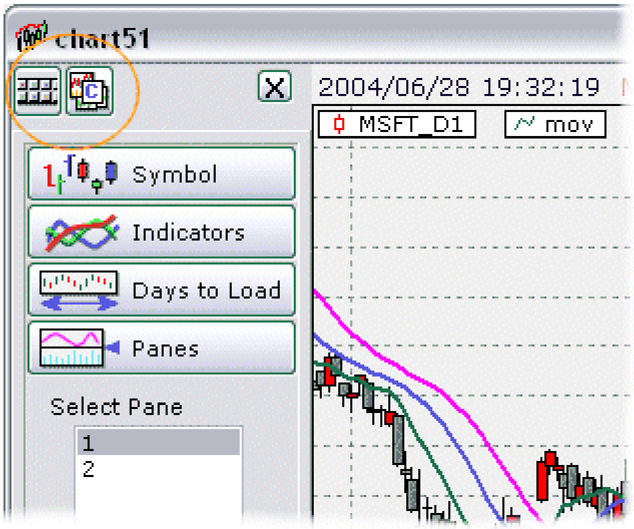
Pressing the **Panes** tab allows you to manage the panes. You can insert new panes by pressing the **Append** or **Insert** button and deleting panes by pressing the **Delete** button.

Panes containing scaling information. For example, the range of values displayed in the pane, whether or not the pane is auto scaled to the data and indicators in the pane, and whether linear or log scale is used. If you need to edit these information, press the **Edit** button.

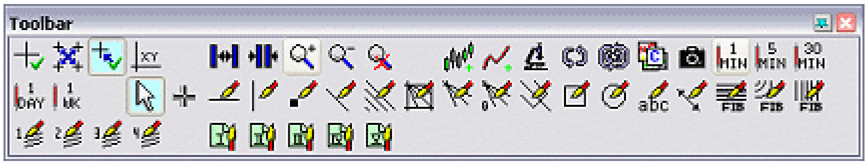
You can read more about panes in ***Working with Panes*** (on page 1043).

Opening the Tool Bar and Chart Manager

Tool bar and chart manager are two windows that are commonly used to assist working with time charts. User panel has buttons to help you quickly launch these windows.

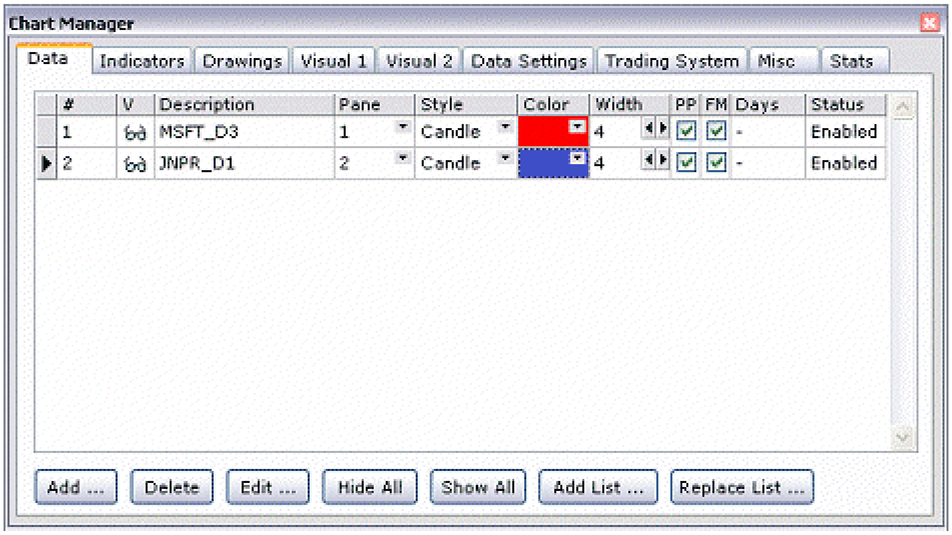


The tool bar lets you quickly access many chart functions. To bring a tool bar near the time chart window, press the tool bar button ▦.



To find out more about the tool bar, refer to ***Tool Bar*** (on page 1209).

The chart manager is a window to let you edit many properties of the chart. To open the chart manager, press the chart manager button .



#	V	Description	Pane	Style	Color	Width	PP	FM	Days	Status
1	👓	MSFT_D3	1	Candle	🟥	4	☑	☑	-	Enabled
2	👓	JNPR_D1	2	Candle	🟦	4	☑	☑	-	Enabled

To find out more about the chart manager, refer to ***Chart Manager Window*** (see "Chart Manager" on page 1013).

Configuring User Panel Behavior

Behavior of user panel can be configured in the user preference. Go to the main window, choose **Program>User Preference**, click on the **Time Chart** tab, under **User Panel** group.

To enable/disable user panel for new charts, check **Show user panel for new charts**.

To enable/disable sub-minute time frame in user panel, check **Show sub-minute timeframe**.

Working without the User Panel

You do not need the user panel to work on a chart. In fact, user panel only offers you the most commonly used chart features. To access the complete set of features of time chart, you should get yourself familiar with time chart's pop up menus. To open the pop up menu, right click on the chart.

Differences between Adding Data using the User Panel and Add Data Window

If you have read other sections in this manual, you may be familiar with adding data series using the add data window.

Adding data series using the add data window is the manual way of adding data to a chart and you have complete control over all data settings. Naturally, there are differences between adding data using the user panel and the add data window.

User panel performs intelligent calculations on the days to load, depending on the time frame. In general, the user panel does a good job of guessing the number of days. The days to load is set to the data series itself, not to the whole chart. If you need to change the days to load, you can use the days to load pane or edit the data series directly.

On the other hand, when you add data with the add data window, the days to load is default to the last setting you set to the add data window. Typically this will be default to the chart's days to load.

Time Chart Configuration

Auto Chart Scrolling

Time chart scrolls automatically as new real-time data arrives. Auto scrolling happens only when the last bar of the first data series is close to the right edge of the chart.

You can turn on/off auto scrolling and adjust the tolerance.

1 Open the Chart Manager.

2 Click on the **Visual 2** tab.

3 Settings are under **Auto Scroll**.

4 Press **Apply** button after you finish making changes.

Color and Font Settings

To adjust font and color settings for time chart.

1 Open chart manager.

2 Press **Visual 1** tab.

3 Most color and font settings are interactive. You do not need to press the Apply button in chart manager to make the setting effective.

The **Small**, **Medium**, **Large** button let you quickly change fonts. You can override font settings for individual items in the chart.

You can quickly reverse the chart's color by pressing the **Inverse** button.

Company Name Support

Data series can display company names in legend. This feature is data feed dependent.

1 Edit Data Series.

2 In the Edit Data dialog, besides **Legend**, choose one of **Sym**, **Name**, **Name(Sym)** or **Sym(Name)** for different combinations of symbol and company name display.

Conserve Memory

If you load many data series into the chart, the chart can consume a lot of memory. This is especially true for intraday charts with lower time frames (seconds, ticks).

You can set all data series to specific length to conserve memory.

1 Open Chart Manager.

2 Press **Data Settings** tab.

3 In **Conserve Memory**, choose one of the options.

4 Press **Apply** button.

Below is the explanation of the options.

None

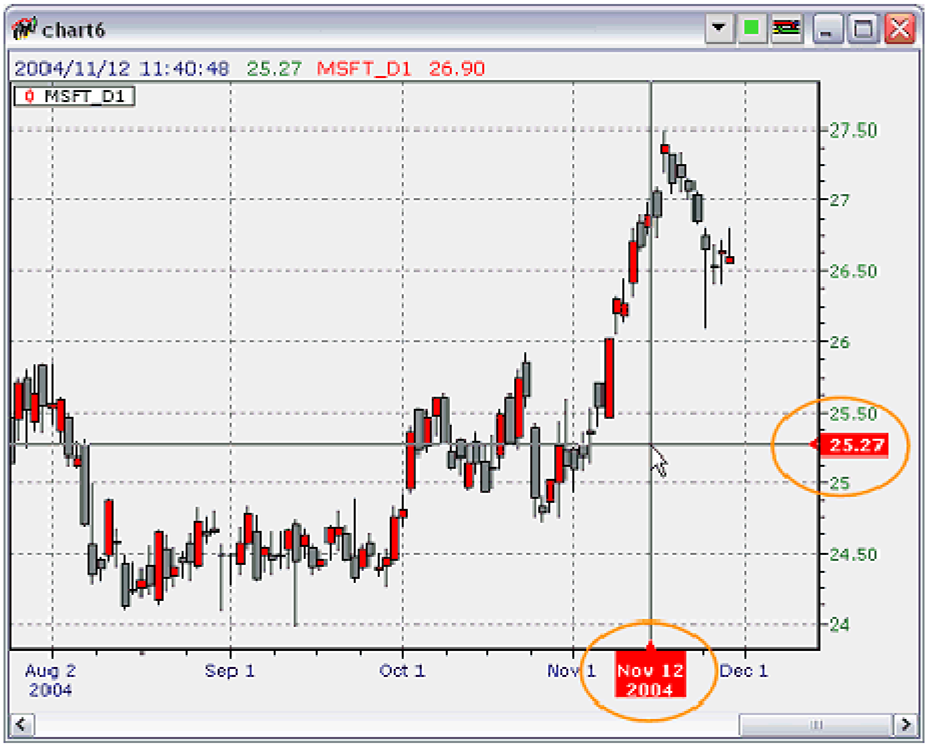
Data series are not cut. This is the default.

Fixed # of bars after reload

Data series are cut to the specified number of bars after data loading or after **Apply** button is pressed. Data series will increase in size as real-time data arrives.

Note that after a data series has been cut, earlier data is no longer available for indicator calculation.

Crosshair Float Marker



Crosshair Float Markers are the float markers for crosshair. By default, it is set to **Pretty** mode.

You can set different Crosshair Float Markers modes by:

1 Choose **Program>User Preference** to open user preference window.

2 Press the **General** tab.

3 Under **Crosshair Float Marker**, choose **None**, **Fast (graphics card intensive)** or **Pretty (CPU intensive)**.

Choose None mode if you find Crosshair Float Markers too distracting, or if you run NeoTicker® on a slower computer.

Fast mode utilizes faster graphics card hardware for drawing. If you find drawing under Pretty mode too slow, and if your computer has an separate AGP graphics card with 64M video RAM or higher.

Pretty mode emphasize more on display quality than speed. It is usually sufficient for simple to moderately complex charts.

Data Alignment

Data alignment controls where data series is placed (left or right) when there is not enough bars to fill the chart. Default alignment is set to right. To configure data alignment:

1 Open ***Chart Manager*** (on page 1013).

2 Press **Visual 1** tab.

3 Choose the option under **Data Alignment**.

Default Zoom Policy

Default zoom policy can be set:

1 Open the Chart Manager.

2 Click on the **Visual 2** tab.

3 Settings are under **Default Zoom Level**.

4 Press **Apply** button after you finish making changes.

The settings are used when you reset the zoom level to default by un-zooming.

Drawing Tools Keep on Drawing

After you drew something with a drawing tool, you have to select the drawing tool on the tool bar to draw again.

If you prefer to keep on drawing until manually turning off the drawing. You can configure the tool bar for this behavior.

Choose **Manager>Tool Bar** from the main window.

Turn on the option **Keep the drawing tool so you can use it again** option. Press the **OK** button.

After you turn on this option, you can keep on drawing using the same drawing tool until you press the pointer cursor or crosshair cursor button on the tool bar. You can also use the TAB key on the keyboard to toggle away from the drawing tool.

Dynamic Grid

Time chart has a dynamic grid in it that can display quotes, formula and simple charts.

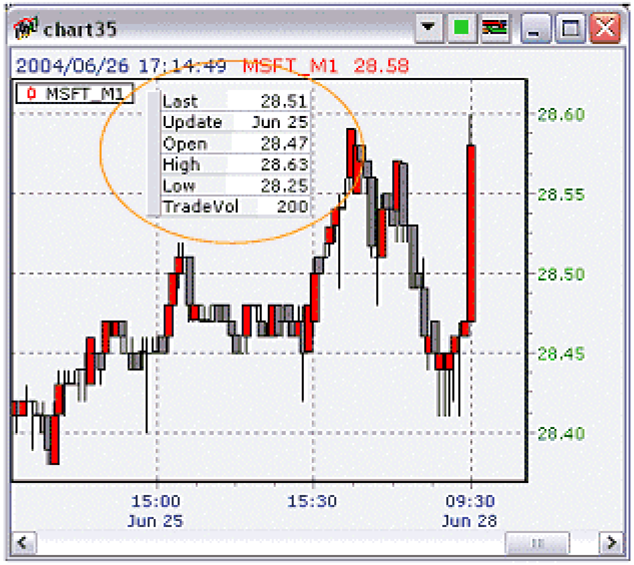


chart35

2004/06/26 17:14:49 MSFT_M1 28.58

Last	28.51
Update	Jun 25
Open	28.47
High	28.63
Low	28.25
TradeVol	200

Dynamic grid can be dragged around and resize. If you prefer not to have dynamic grid in the chart, right click on the chart and choose **Hide** from the pop up menu.

To show the dynamic grid again, choose from pop up menu **Dynamic Grid**.

For more information about dynamic grid set up, refer to ***Dynamic Grid Operation Guide*** (on page 545).

Even Bar Space between Bars

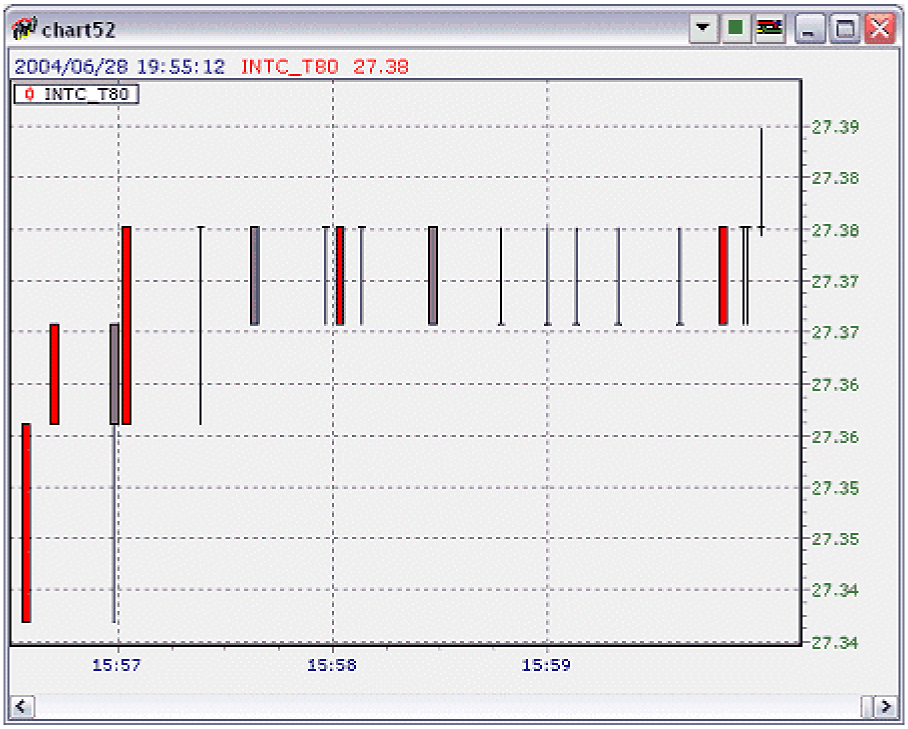
A video tutorial is available. See ***Tick Chart Bar Space Video Tutorial*** (http://newsletter.neoticker.com/?p=165).

To change between time driven bars and even space bars:

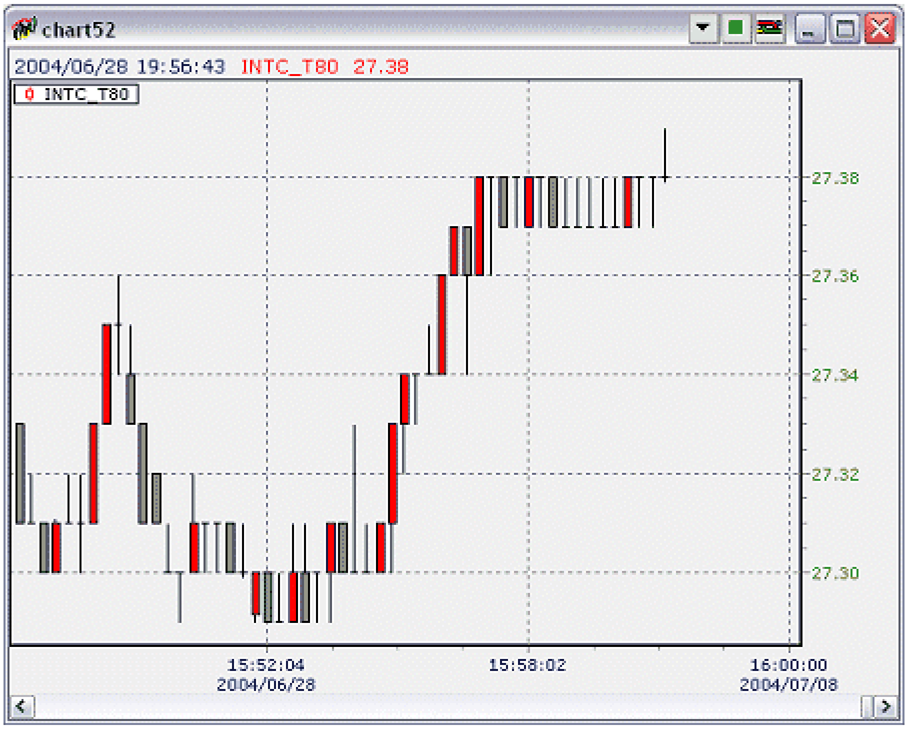
1. Open chart manager.
2. Press the **Data Settings** tab.
3. Choose between **Time Driven** and **Bar Driven** (even bar space).
4. Press the **Apply** button.

Background

The following chart is a time driven 80-tick chart. Notice that the bar spacing are not even and the physical position of each bar properly reflect the relative time position. Non-even spacing is useful if you want to mix time data series (e.g. 1-min INTC) with the 80-tick data series in the same window.



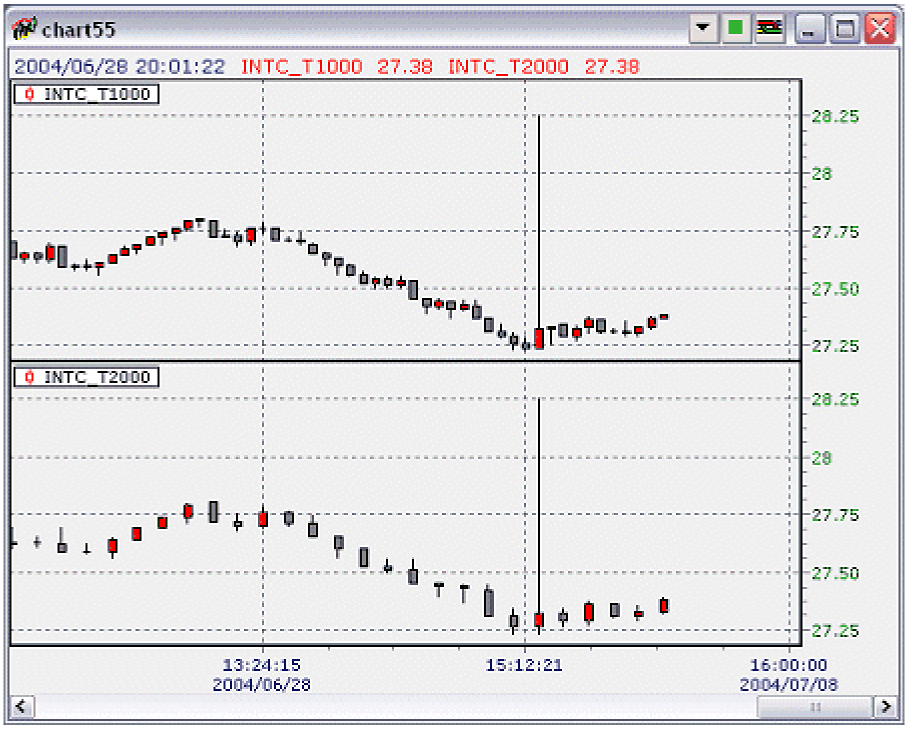
The following is the same chart with bar driven style (even bar space). In this chart, time is no longer even in the time axis. This is useful for users who want to focus on the ticks traded, not time.



Several important issues related to using bar driven time axis style.

- All data series must be a larger multiple and the same period of the first data series of the same symbol. The reason behind is obvious - there is no objective rule to line up a second data series to the first if these conditions are not match. The following table illustrates what is valid and what's not:

First Data Series	Another Data Series	Validity
80-tick INTC	160-tick INTC	Valid - 160 is a larger multiple of 80, symbol is t same
80-tick INTC	40-tick INTC	Invalid - 40 is not a larger multiple of 80
80-tick INTC	160-min INTC	Invalid - tick and min are not the same period
80-tick INTC	160-tick MSFT	Invalid - Symbols are not the same



Basically, you are free to adjust any data series time frame as long as all the data series in the chart remains compatible.

- Data of time frame are loaded without invalid bars. The concept of invalid bars is not applicable to bar driven time axis.
- In real-time, the last bar of the higher time frame (e.g. the bar of 160-tick in a 80-tick, 160-tick example) is lined up with the lower time frame bar. This is because it is that there are not enough tick within the current trading day to construct the higher time frame bar. The presents an issue with indicator writer who wants to mix the time frame for analysis. Because last bar is lined up, pbarindex should be used to correctly match data series index. For an example of pbarindex, see ***Last Bar Position during Real-time update for EOD Data Series*** (on page 1079).

Extra Space for Price Axis Labels

If you need extra space in the price axis (e.g. the indicator you use returns a very large value), you can configure time chart to allow extra space for the price axis labels.

1. Open chart manager.
2. Press **Visual 1** tab.
3. Toggle the **Wide Price Axis** label.

Hiding and Showing the Scroll Bar

You can hide the scroll bar to get more usable space in your charts.

1 Open the Chart Manager.

2 Click on the **Visual 1** tab.

3 Toggle **Time Scroll Bar** inside **General**.

Alternatively,

1 Double click on time axis to open Time Axis Setup.

2 Press **Scroll Bar** tab.

3 Toggle **Time axis scroll bar is visible**.

4 Press **OK** button.

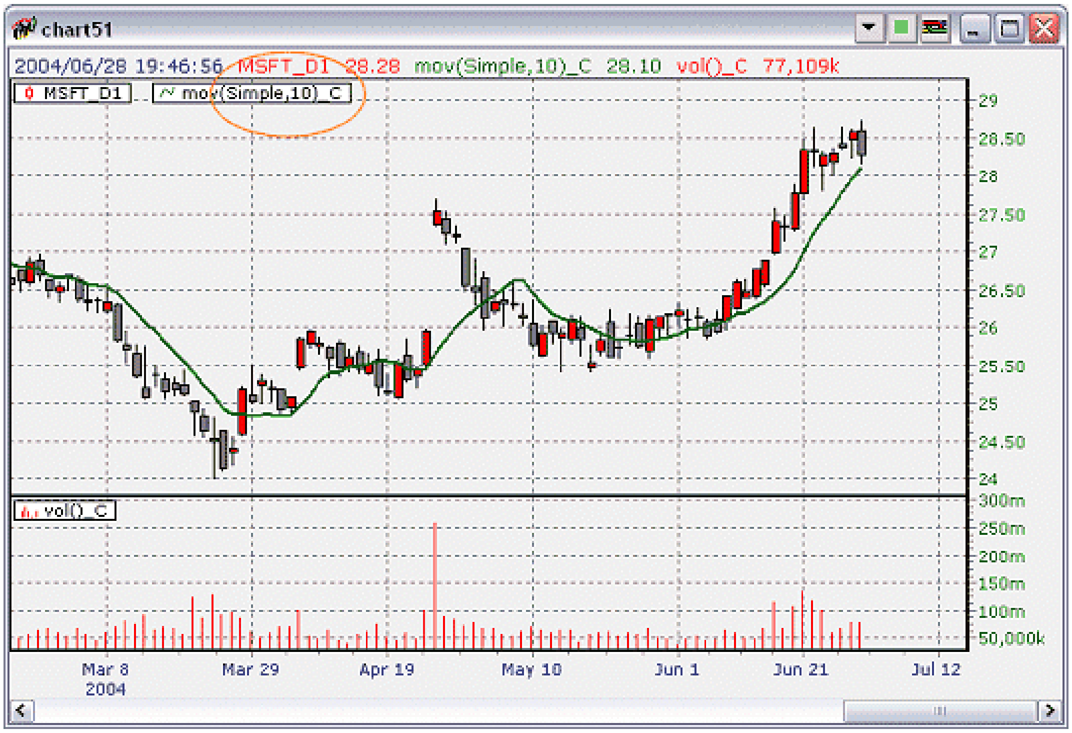
Indicator Parameters in Time Charts

You can tell time chart to display all the indicators' parameters in the chart legend. This is global setting that affects all charts.

1 Open user preference by choosing **Program > User Preference** in the main window.

2 Press **Time Chart** tab.

3 Toggle the **Display indicator parameters in legends** option under **General.**



Last Bar Position during Real-time update for EOD Data Series

If the time chart has EOD data series (e.g 1-day), time chart offers two options on where the last bar of the EOD data series is drawn when the data series is updated in real-time:

- Option 1 - Draw at exact time position, i.e. at the closing of the trading day.
- Option 2 - Draw lining up at the time stamp of the last tick that comes in.

To configure:

1 Choose Program>User Preference from the main window

2 Press the Real-Time Data tab

3 Choose **Always stamp the end time of the bar duration** for option 1

4 Choose **Stamp the last valid daily bar within the bar duration** for option 2

Which Style should You Use

Option 1 provides best consistency in all indicator calculations involves mixing data of different time frame. The reason is the EOD bar is not moving around during real-time update. No special pre-caution is needed when you perform formula calculation or writing indicators.

Option 2 provides better visual presentation. Consider if you have 1-min data series with 1-day data series in the same chart. The 1-day bar will be positioned at the end of trading day. The 1-day bar can be very far away time-wise with respect to 1-min bar and make reading difficult.

Caution when Lining up Bars for the Second Option

If you choose Option 2 and write formula and scripts that mixes data series, you should use the Previous Bar Index indicator (pbaridx) to help you figure out how the bars are lined up. Previous Bar Index takes two links. It returns 1 when the bar of the second link is exactly at the position of the first link.

For example, in formula2, if data1 1 is 1-min and data2 is 1-day, the following formula properly lines up the two data series for a previous day high comparison:

```
data1 > prevDHigh(pbaridx(0, data1, data2), data2)
```

Price Axis Configuration

Price axis is configured on a per pane basis, i.e. you can have different price axis configuration for each pane. For more information on configuring price axis, see ***Change Price Axis Scaling and Labelling*** (on page 1049).

Redraw Frequency

By default, time chart is redraw on timer.

Redraw is separate from tick-by-tick calculation. Even if the chart is redraw on timer, all ticks are accounted for in calculations.

You can increase/decrease the redraw frequency to make the best use of your CPU. To adjust:

1. Open the Chart Manager.
2. Click on the **Visual 2** tab.
3. Settings are under **Redraw Frequency.**
4. Press **Apply** button after you finish making changes.

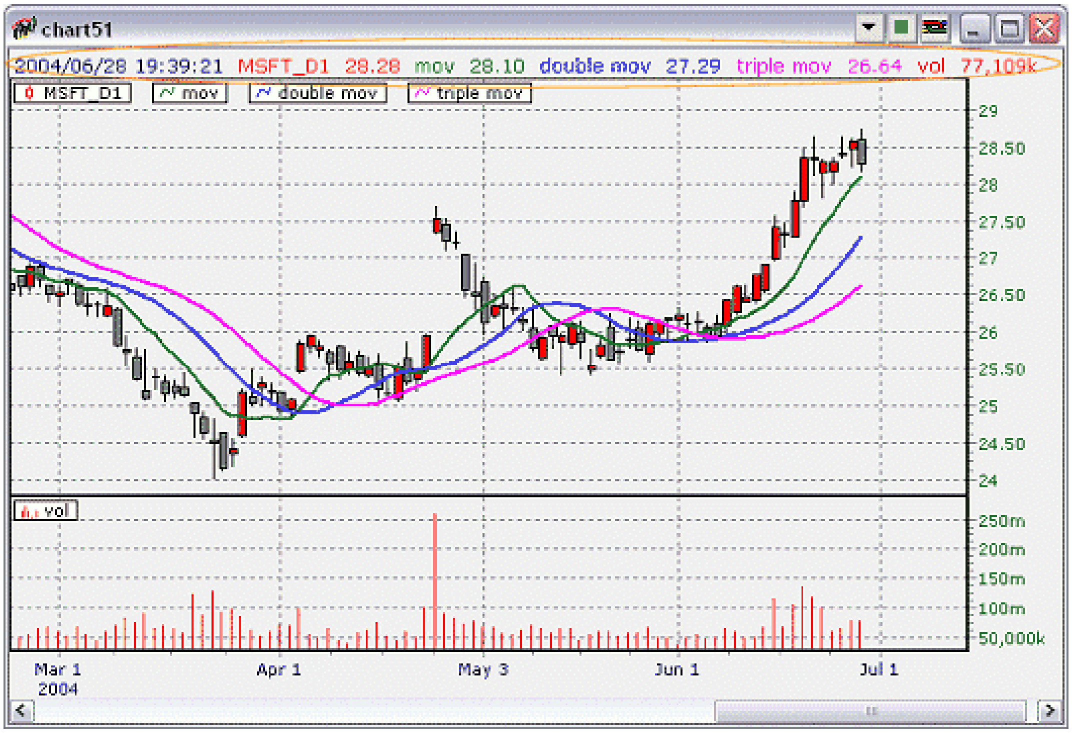
Use **Tick Redraw** (chart is redraw for every tick that comes into the chart) only for sparsely traded instrument and you do not plan to open many charts. Tick redraw will quickly add to CPU usage even for a high end Pentium 4 class computer.

Report Area

Report area is the top part of the chart that displays prices and indicator values.

If crosshair is off, report area shows the last value of the data series and indicators.

If crosshair is on, report area shows the values where the crosshair points to.



2004/06/28 19:39:21 MSFT_D1 28.28 mov 28.10 double mov 27.29 triple mov 26.64 vol 77,109k

To configure report area:

1 Open chart manager

2 Press **Visual 1** tab

3 Report area is configured under **Last Time/Sales**. You can choose between **None**, **Data & Indicator** and **Data Only**.

Right Side Spacing

Time chart maintains a right side space in the chart. You can control the spacing by:

1 Open the Chart Manager.

2 Click on the **Visual 2** tab.

3 Settings are under **Right Side Spacing**

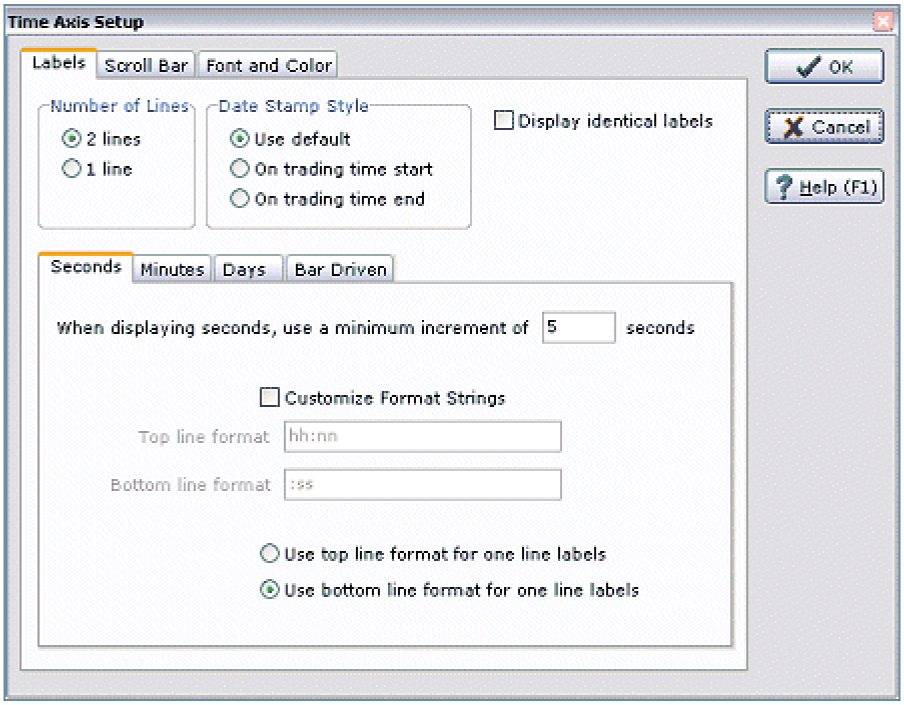
4 Press **Apply** button after you finish making changes.

Scale Button in Price Axis

To hide scale button in price axis:

1. Choose **Program>User Preference** from main window.
2. Press the **Time Chart** tab.
3. Under **General**, toggle **Show scale button for fixed scale panes.**

Time Axis Configuration



Time Axis Setup

Number of Lines
● 2 lines
○ 1 line

Date Stamp Style
● Use default
○ On trading time start
○ On trading time end

□ Display identical labels

When displaying seconds, use a minimum increment of 5 seconds

□ Customize Format Strings

Top line format hh:nn

Bottom line format :ss

○ Use top line format for one line labels
● Use bottom line format for one line labels

You can configure time axis by double clicking in the time axis area. The above dialog will appear.

One Line vs. Two Lines Label

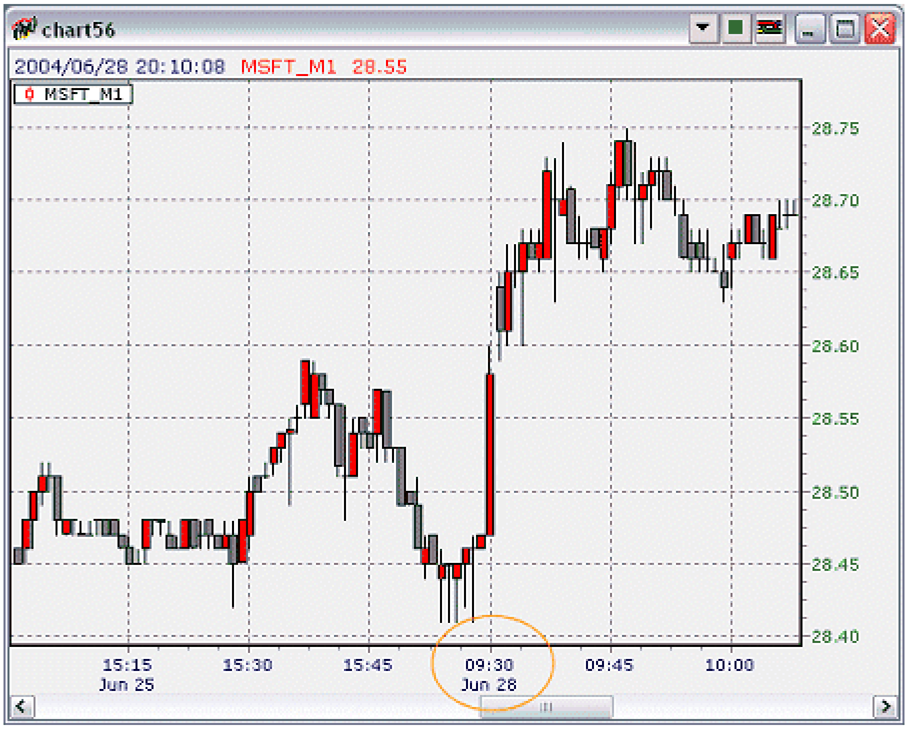
You can control how many lines the time axis labels occupy.

1. Double clicking the time axis to open the time axis setup dialog.
2. Press the **Labels** tab.
3. Choose **2 lines** or **1 line** under **Number of Lines**.

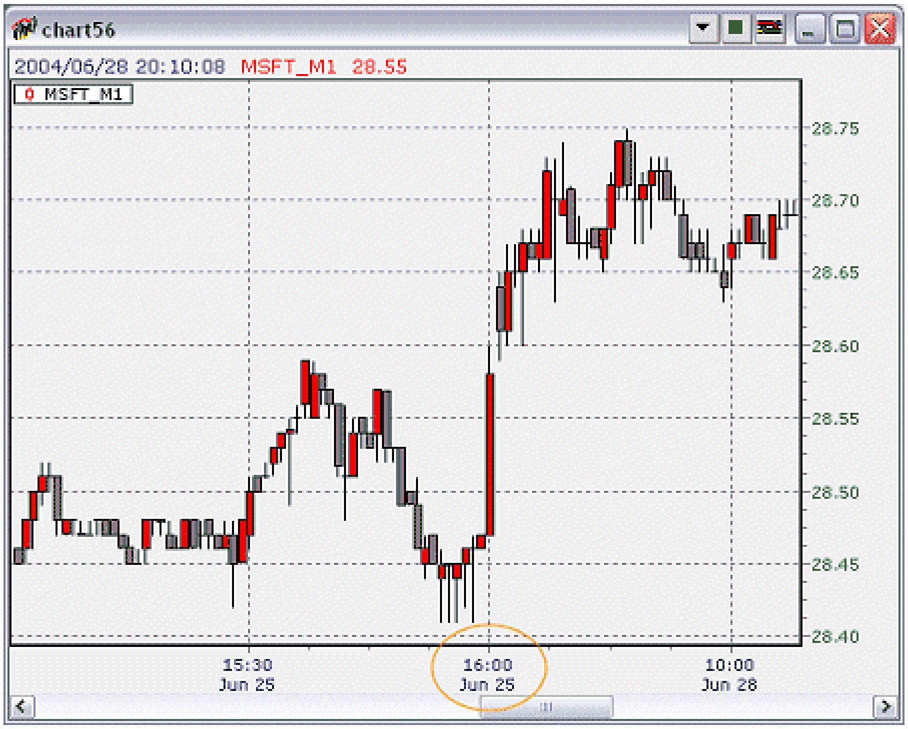
If you choose **1 line**, you need to determine whether the top line format string or the bottom line format string is used for the label, by choosing **Use top line format for one line label** or **Use bottom line format for one line label** in **Seconds**, **Minutes**, **Days** and **Bar Driven** tabs.

Date Stamp Style for Labels

By default, date is stamp at the beginning of trading day. The Jun 28 label at 9:30 in the following chart is an example:



Some users may prefer date stamp at the end of trading day, especially for daily chart. The Jun 25 label at 16:00 in the following chart is an example:



You can set application wide date stamp option by:

1 In main window, choose **Program>User Preference**.

2 Press the **Time Chart** tab.

3 Choose **On Trading Time Start** or **On Trading Time End** under **Time Axis Date Stamp Style**.

You can override the option in individual charts by:

1 Double clicking the time axis to open the time axis setup dialog.

2 Press the **Labels** tab.

3 Choose **Use Default**, **On trading time start** or **On trading time end** under **Date Stamp Style**.

Date stamp style does not affect NeoTicker® EOD. NeoTicker® EOD automatically uses stamp at end of trading day.

Display Identical Labels

In time axis, if a label is identical to the previous label, it is not displayed. You can force identical label to display by:

1 Double clicking the time axis to open the time axis setup dialog.

2 Press the **Labels** tab.

3 Enable **Display identical labels**.

Label Formatting

Under **Labels** tab, there are controls to let you precisely control labels are formatted. Specially, they are under the **Seconds**, **Minutes**, **Days** and **Bar Driven** tabs.

Time axis uses different formatting rules depend on the time range of the axis. The rules are as follows:

- Seconds. When time axis covers a short time duration and seconds are displayed. This rule is used only for time driven bars.
- Minute. When time axis covers a time range used by typical intraday charts. This rule is used only for time driven bars.
- Days. When time axis covers a time range used by typical daily or above chart. This rule is used only for time driven bars.
- Bar Driven. This rule is used only for bar driven charts, e.g. even bar space n-tick charts.

Each rule has two formatting strings. The formatting string determine the top label and bottom label for the time axis.

The formatting string use the same formatting code as default date time value. You can see the section ***Setting a Default Date Time Format*** (see "Default Date Time Format" on page 503) for the description of formatting code.

Some formatting rules support a minimum increment (e.g. seconds). Minimum increment is only a suggestion. If time chart determines that there is not enough space for all the labels if minimum increment is used, a multiple of the minimum increment will be used. For example, in the seconds rule, if a minimum increment is 10 second, you can expect the labels to look like '0, 10, 20,...', or '0, 20, 40...', but not '0, 15, 30,...'.

Scroll Bar Tab

You can set whether the scroll bar is visible here.

Font and Color Tab

You can set time axis font and color here.

Trading Time and Holiday List Settings

Trading time and holidays affect the data that is included in the chart. This in turns affect indicator calculation.

For example, if a data appears outside of trading time specified, time chart will ignore it. In turn, indicator calculation will not see the data.

If data bar is missing (for example, a day is actually a holiday but is not specified as such), the data for the time period will be included in the time chart. The data will be marked at invalid but still occupies the position. Indicators see invalid data bars. How invalid bars are handled is indicator dependent.

To change the trading time and holiday list settings,

1 Open Chart Manager.

2 Press the **Data Settings** tab.

3 Choose **Time Frame Auto Switched Based On Primary Symbol** to make the time frame switch automatically with the primary symbol. Primary symbol time frame is defined either in Symbol Info Manager (higher priority) or as default time frame in User Preference (lower priority). See ***Default Time Frame*** (on page 506).

4 Choose **Predefined Time Frame** to use a pre-defined time frame.

5 Choose **Customized** for manual settings, i.e. **Holiday List** and **Trading Days**.

6 Change settings and press **Apply** button.

User Panel Configurations

For user panel configurations, refer to ***Configuring User Panel Behavior*** (on page 1069).

Value Axis to Log Scale

You can change the value axis to log scale:

1 Un-select everything by pressing the ESC key

2 Choose **Pane/Price Axis>Logarithmic** from the pop up menu.

To change back to regular scaling, choose **Pane/Price Axis>Arithmetic**.

Visual Style for Candlestick, Bar and Box

Version 4.1 introduce sharp edge visual style for candlestick, bar and box.The new style makes large candlesticks, bars and boxes look better and is the default.

If you encounter performance problems when using the new sharp edge visual style, you can switch back to using the classic style:

1 In Main window, choose **Program>User Preference** to open User Preference window.

2 Click on **Time Chart** tab.

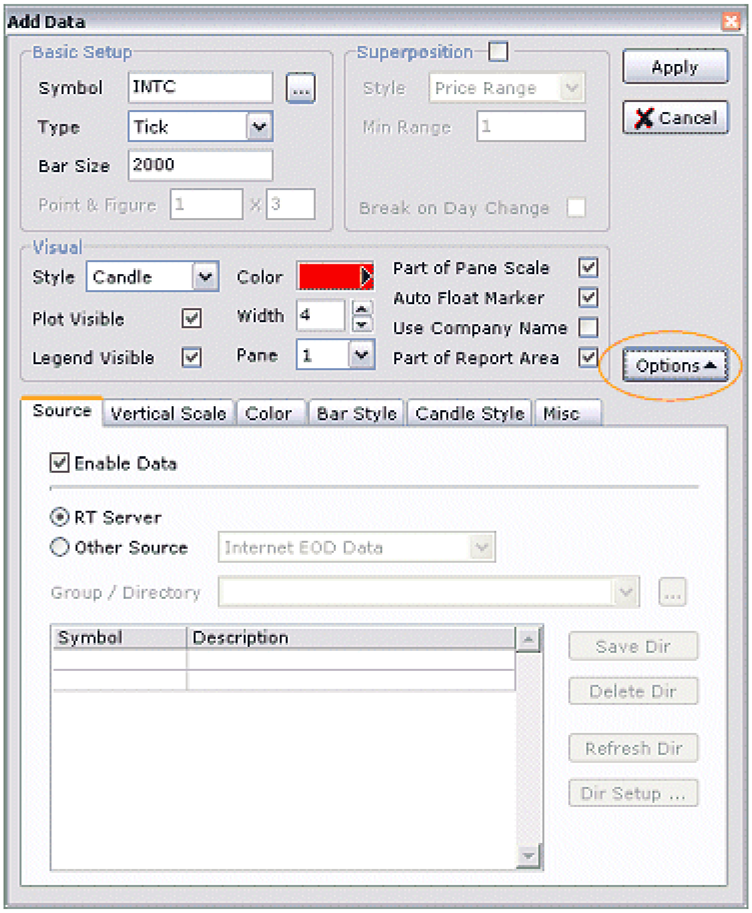
3 Under **Candle / Bar / Box Visual Style**, choose the option.

Data Series and Indicators

Adding/Editing Data Series

When you add a data series, NeoTicker® opens an add data window. The same window is opened when you edit data series.

Pressing the **Options** button to reveal/hide all the options in the dialog.



Add Data

Basic Setup

Symbol INTC

Type Tick

Bar Size 2000

Point & Figure 1 X 3

Superposition □

Style Price Range

Min Range 1

Break on Day Change □

Visual

Style Candle

Color 🟥

Width 4

Pane 1

Plot Visible ☑

Legend Visible ☑

Part of Pane Scale ☑

Auto Float Marker ☑

Use Company Name □

Part of Report Area ☑

☑ Enable Data

● RT Server

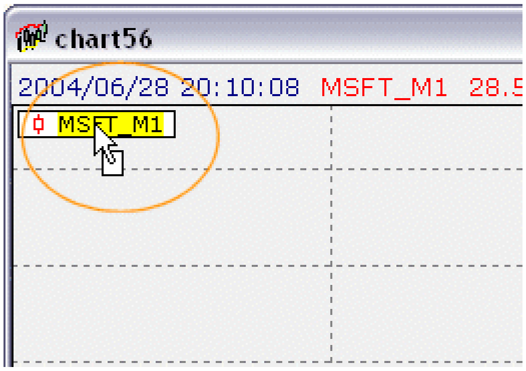
○ Other Source Internet EOD Data

Group / Directory

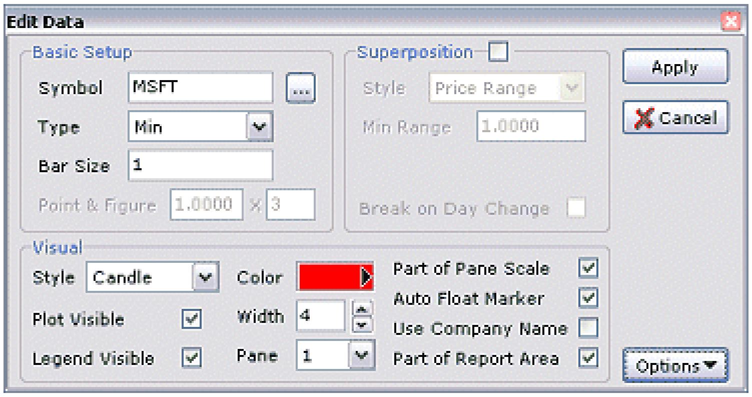
Symbol	Description

Replacing and Editing a Data Series

Click on the legend of the data series. Notice that the data series becomes yellow. This is called highlighting. You can perform specific function on a highlighted item.



Press the space bar. The Edit Data window is opened. The Edit Data window is identical to the add data window except it lets you edit an existing data.



Edit Data

Basic Setup
Symbol: MSFT
Type: Min
Bar Size: 1
Point & Figure: 1.0000 x 3

Superposition □
Style: Price Range
Min Range: 1.0000
Break on Day Change □

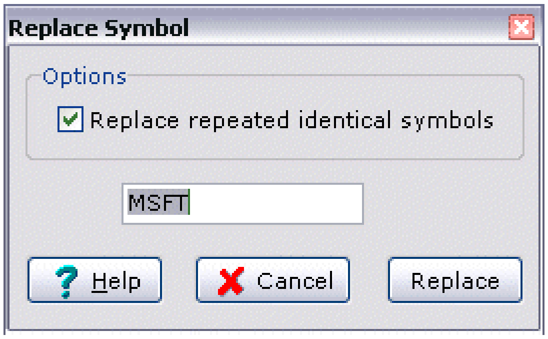
Visual
Style: Candle
Plot Visible ☑
Legend Visible ☑
Color: 🟥
Width: 4
Pane: 1
Part of Pane Scale ☑
Auto Float Marker ☑
Use Company Name □
Part of Report Area ☑

Make the setting changes. Press the **Apply** button to confirm.

- The full set of options of the dialog is shown if you press the **Options** button.
- Some feature such as color provides an immediate feedback and you do not have to press the **Apply** button for these changes to take effect.

Quickly Replacing Multiple Identical Symbols

If you have multiple identical symbols in chart (e.g. 1-min, 5-min, 10-min MSFT for multiple time frame analysis), you can quickly replace the symbols by pressing F9.



☑ Replace repeated identical symbols

MSFT

F9 is similar to chain replace, except it only works on the active window. So it is particularly handy for replacing multiple identical symbols in time charts.

Deleting Data Series

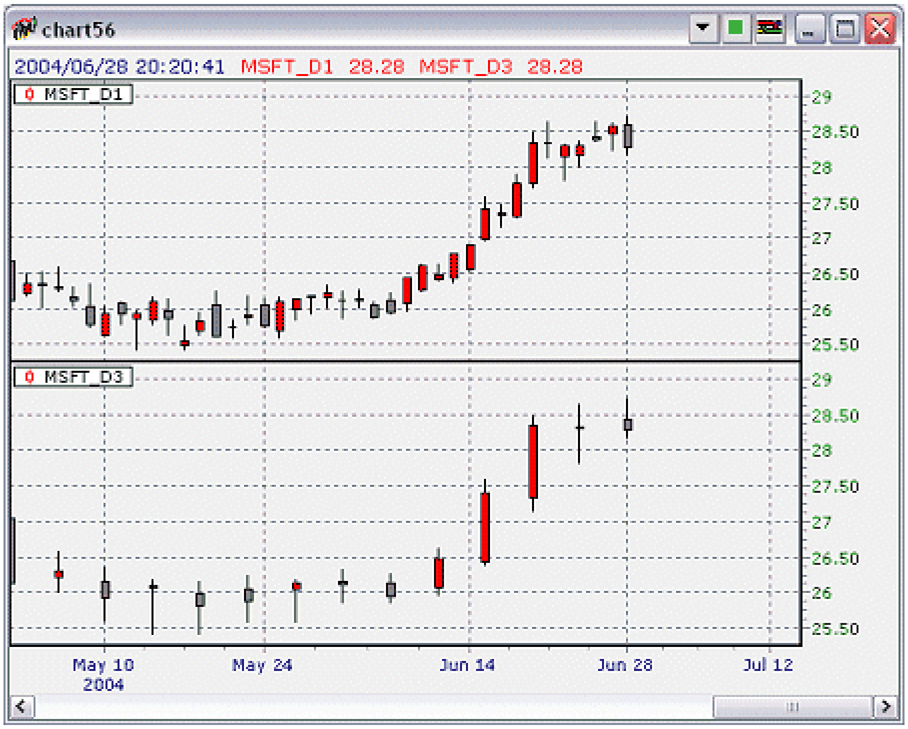
You can delete a data series by selecting the data series, and choose **Delete** from the pop up menu or press the DELETE key.

Alternatively, you can use the chart manager window to delete a data series.

Multiple Time Frames

One of the most powerful feature of NeoTicker®'s time chart is its ability to mix data series of different time frames into the same chart, even in the same pane.

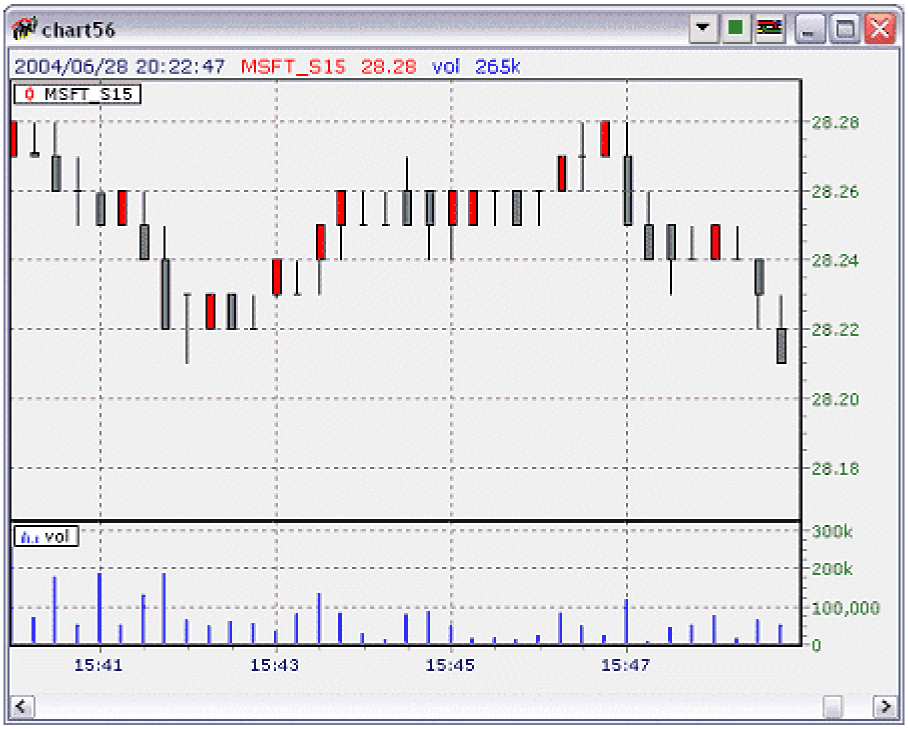
For example, you can mix a 1-day daily chart and a 3-day daily chart. Time chart window will correctly draws all bars at the right time. The following figure is a time chart that shows MSFT in 1-day and 3-day daily time frames.



Second Bars

You can use second time frame for charting and scanning. For example, you can construct 15-second bars to summarize the ticks every 15 seconds. This feature allows you to construct bars that are more refined than one minute bar yet unlike tick driven bars, are lined up uniformly in time.

By default, you can only specify second bars by add/edit data window. To use second bars with user panel, you must enable the user panel to show sub-minute time frames. See ***Configuring User Panel Behavior*** (on page 1069).



Unlike minute bars, second bars must be constructed from ticks. This is because currently no data vendor provides historical second bars. Constructed second bars from ticks has several consequences.

Pros:

- Because bars are constructed from ticks, indicators and trading systems that depend on tick-by-tick updates behave exactly the same way during data loading and real-time update. There is no need to specify update by tick manually.

Cons:

- Second bar loading is much slower than minute bar loading. To avoid unnecessary delay, only load the data you need
- Second bars contain the noise of typical tick data. Data filtering that is happened on minute bar level will not be applied to second bars

Multiple Tick Driven Bars

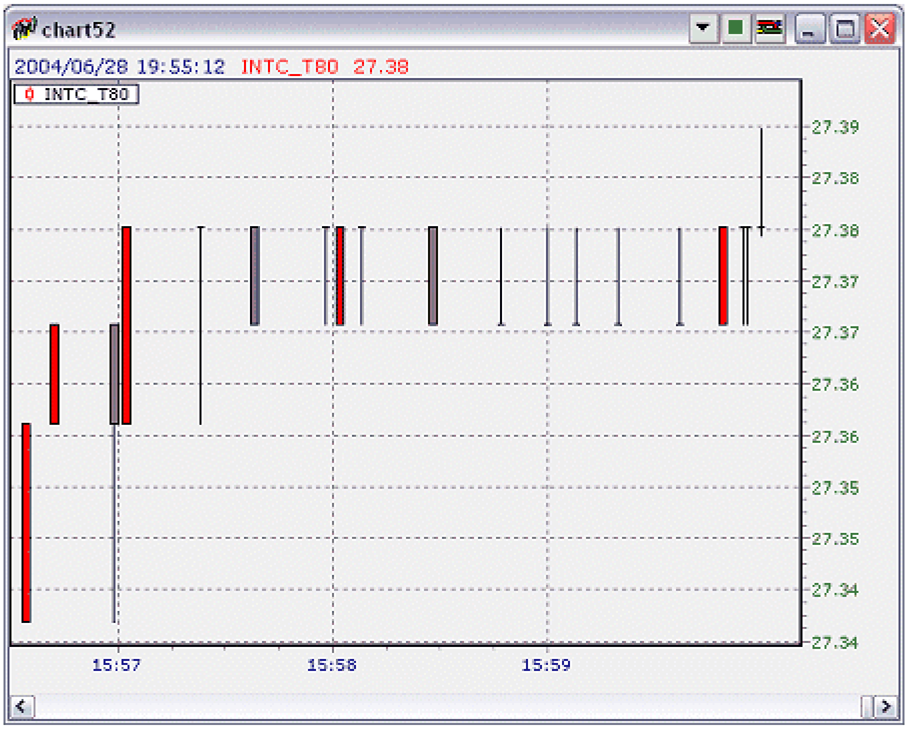
You can chart multiple tick driven bars (e.g. 1000-tick chart). In a tick chart, the tick count for bars are reset to 0 at the beginning to each trading day.

By default, you can only specify tick driven bars by add/edit data window. To use tick driven bars with user panel, you must enable the user panel to show sub-minute time frames. See ***Configuring User Panel Behavior*** (on page 1069).

Time Driven Tick Chart

Time driven tick chart is the default. The tick driven bars are lined up correctly in time. This allows you to add any data series (e.g. 15-sec, 1-min, 500-tick, etc) to the chart with no restriction.

Note that because the bars are constructed by a constant number of ticks but lined up in time, the space between each bar is non-uniform.



Bar Driven Tick Chart

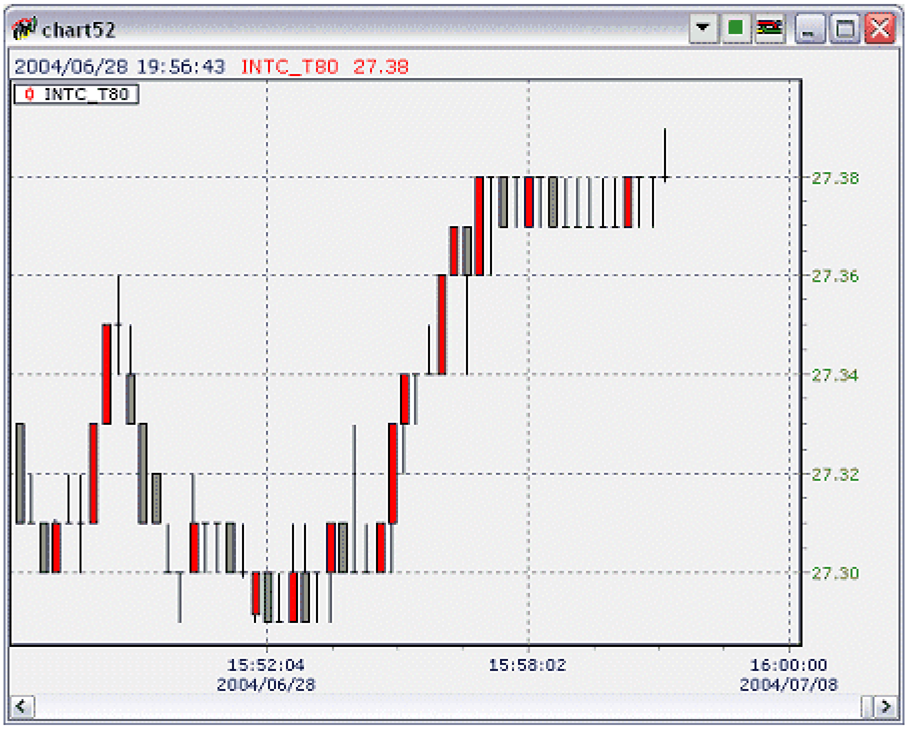
Many users prefer an even space tick chart where the space between each bar is even. To setup:

1 Open chart manager.

2 Press **Data Settings**.

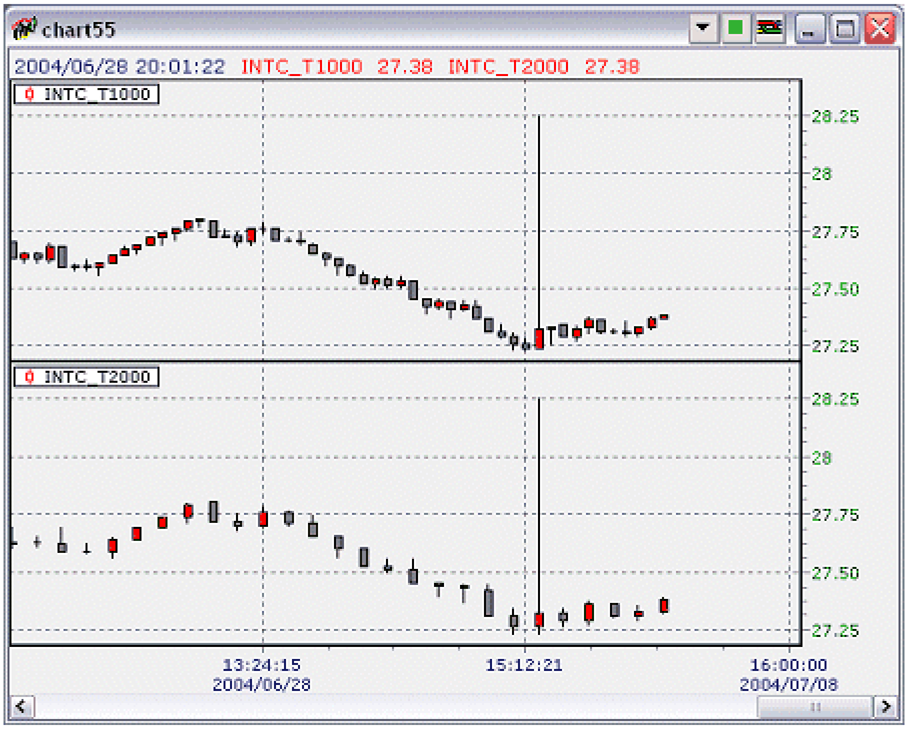
3 Choose **Bar Driven**.

4 Press **Apply** button.



Note that while the space between bars is even, the time interval in the time axis is non-uniform.

You can add more tick data series to the same chart, but with some restrictions. See ***Even Bar Space between Bars*** (on page 1075) for more details.



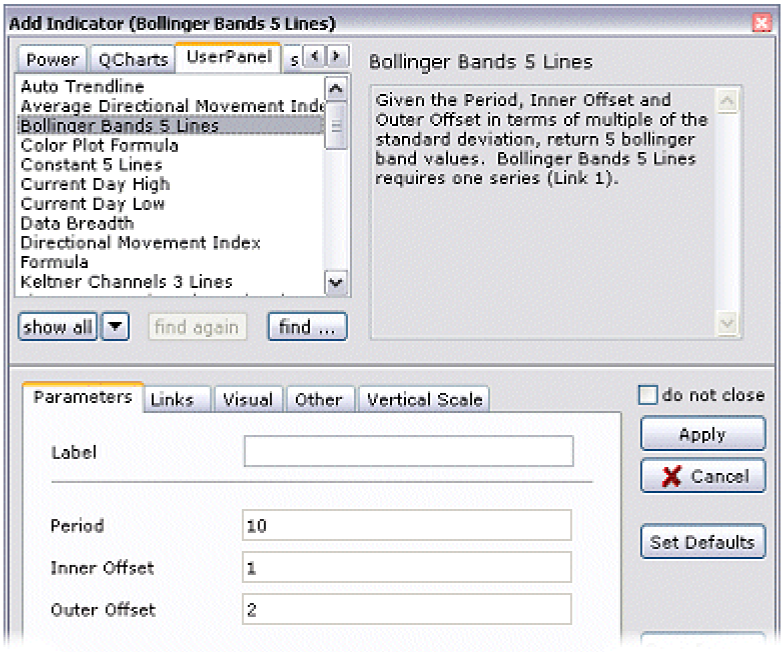
Adding an Indicator

Indicators are also known as studies. You can add indicator by first selecting the source of the indicator. The source of the indicator can be a data series, or another indicator. After the selection:

- Press the add indicator button on the tool bar, or
- Choose **Add Indicator** from the pop up menu.

Another alternative to add indicator is to use the user panel. Refer to ***Using the User Panel in Time Charts*** (on page 1055).

A dialog is opened to let you choose the indicator and its parameter. Press on the **all** tab (or press the **show all** button) and select an indicator (Bollinger Bands 5 Lines for example). Fill in parameters. Press the **Apply** button.

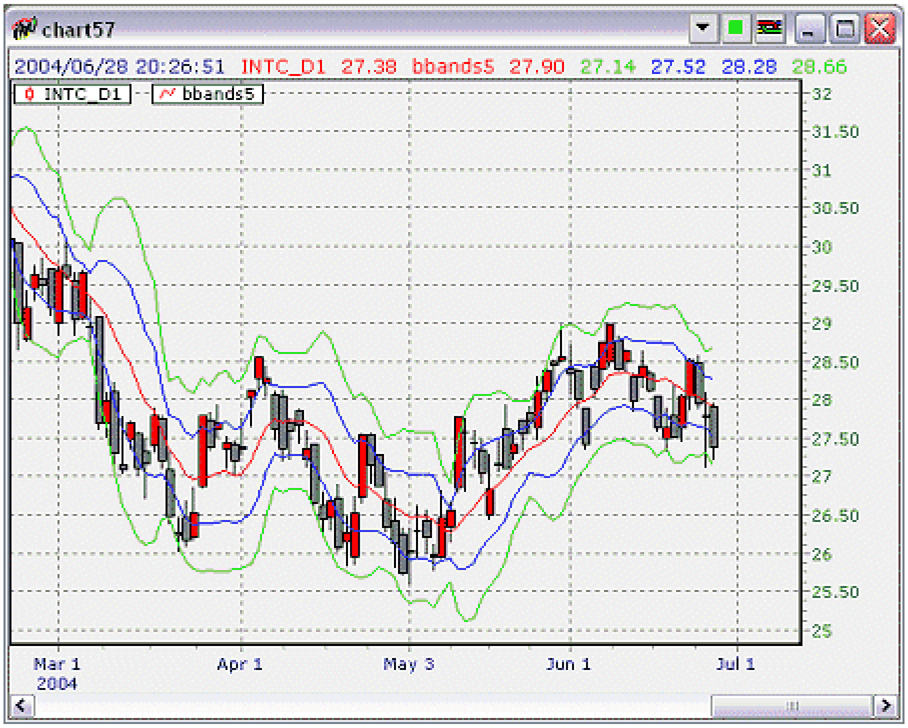


Given the Period, Inner Offset and Outer Offset in terms of multiple of the standard deviation, return 5 bollinger band values. Bollinger Bands 5 Lines requires one series (Link 1).

Label	
Period	10
Inner Offset	1
Outer Offset	2

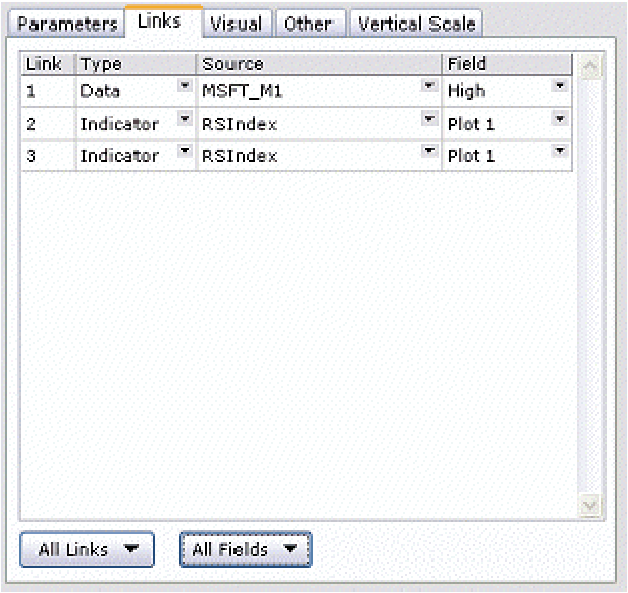
□ do not close

The indicator is plotted to the chart.



Setting Up Links

When you add an indicator, especially on a complex chart, pay attention to the Links tab. This tab specifies the data source of the indicator. For example, the indicator fml3 requires 3 data links:



Link	Type	Source	Field
1	Data	MSFT_M1	High
2	Indicator	RSIndex	Plot 1
3	Indicator	RSIndex	Plot 1

If you press the **All Links** button, you can quickly set up the links to different data series or indicators in the chart.

If you press the **All Fields** button, you can quickly set up **Field** of all links to a common field.

Creating Multiple Data Series

You can create multiple data series with a symbol list.

1. Add a data series to a chart. This list provides the default settings to symbols you are going to add. So you want to set the bars, period, days to load, visuals, etc.
2. Open the chart manager.
3. Press the **Data** tab in chart manager.
4. (Optional) Turn off visibility of the data series you created in the first step.
5. Press the **Add List** button.
6. You will be prompt to select a symbol list. Each symbol in the symbol list will be added to the chart as a data series, using the settings of the data series you created in the first step.

Pressing the **Replace List** button will replace the data series in the chart with the symbols in the symbol list. When you replace a symbol list, only existing data series are replaced. If there are not enough data series for the symbol list, new data series will not be added.

Editing Indicator

You can edit an indicator by selecting the indicator, and choose **Edit** from the pop up menu or press the space bar. An edit indicator window will be opened. The edit indicator window is identical to the add indicator window. You can change the parameters in the edit indicator window and apply the changes to the indicator.

Alternatively, you can use the chart manager window to edit an indicator.

Deleting Indicator

You can delete an indicator by selecting the indicator, and choose **Delete** from the pop up menu or press the delete key.

Alternatively, you can use the chart manager window to delete an indicator.

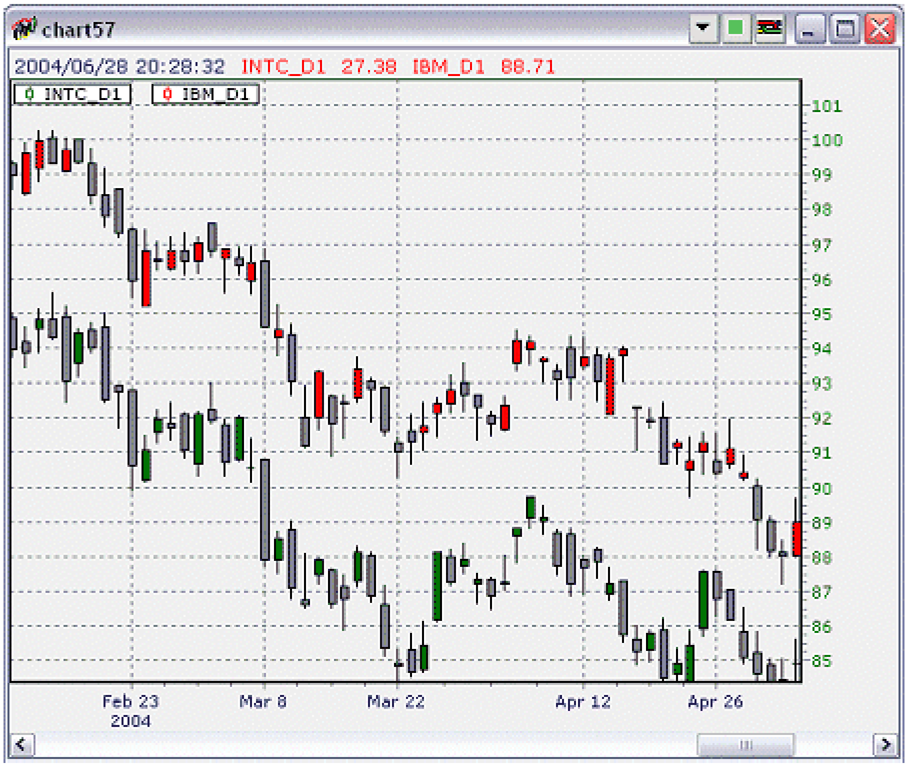
Overlay

Overlay data series and indicators do not share the price axis with the chart. Thus their axis can be moved and scaled independently, making it possible to place series of very different values in the same pane.

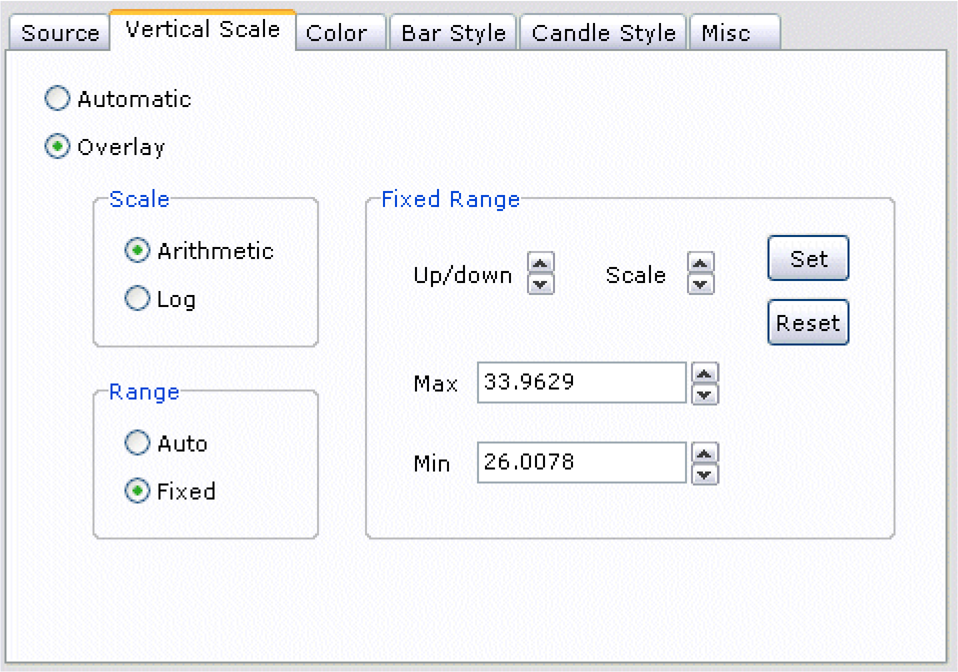
The following example illustrates how to use overlay:

1 Create a new chart and add the data series for IBM and INTC. IBM typically traded at around $100 while INTC typically traded at around $30. By making INTC into an overly data series, INTC can be placed in the same pane as IBM without over stretching the pane axis.

2 Select INTC, and choose **Overlay** from the pop up menu.



To modify an overlay object, open the edit data window (or edit indicator window for an overlay indicator). The settings are under the **Vertical Scale** tab. For data series, you may need to press the **More Options** button to see the tab.



○ Automatic

● Overlay

Scale

● Arithmetic

○ Log

Range

○ Auto

● Fixed

Fixed Range

Max 33.9629

Min 26.0078

Update by Tick Indicators and Dependency

Most indicators are updated by tick. What this means is the indicator value is evaluated once when every tick arrives. To enable/disable this option:

1. Edit indicator.
2. Press on the **Other** tab.
3. Under Update Scheme, choose **By Tick** or **On Bar Completion**.
4. Press the **Apply** button.

Considerations

Update by tick is usually the preferred setting. However, under certain scenarios, you may want to turn update by tick off:

- Indicators that perform a lot of calculations that tend to slow down the whole computer.
- Underlying data series is of long time frame.
- Indicator is a trading system that automates an ordering system. Unless your trading system is designed to handle tick updates (see ***Updated-by-Tick Trading Systems*** (on page 1512)), update by tick can cause multiple firing of orders and misfiring orders on "false" signals.

Update by Tick and Dependency

The general rules for update by tick when an indicator is depending on one or more linked data series and indicators:

- Rule 1: When there is one link and it is data, Update by Tick option is relevant. The option will control whether the indicator will be updated by tick.
- Rule 2: When there is one link and it is indicator, Update by Tick option is irrelevant. The indicator will be updated when the linked indicator is updated.
- Rule 3: When there are more than one link, whether Update by Tick option is relevant depends on primary link.
- Rule 4: When there are more than one link, when any link is updated, the indicator is triggered to update, provided the indicator and the link can be line up in time.
- Rule 5: If indicator **Update Dependency** is set to **Primary Link Only**, the indicator will update like a single link indicator, i.e. Rule 3, 4 are no longer applicable.

The following examples illustrate how the above rules work:

Example 1: Indicator A linked to data

When data arrives, Indicator A is updated by tick if Update by Tick option is set to true (Rule 1).

Example 2: Indicator A linked to Indicator B. Indicator B is not updated by tick

After a bar in Indicator B is completed, Indicator A is triggered to update. Update by Tick option is irrelevant (Rule 2).

Example 3: Indicator A linked to Indicator B. Indicator B is updated by tick

After Indicator B is updated by tick, Indicator A is triggered to update. Update by Tick option is irrelevant (Rule 2).

Example 4: Indicator A linked to Indicator B (link 1) and Indicator C (link 2). Indicator B is updated by tick. Indicator C is not updated by tick

After Indicator B is updated by tick, it will trigger Indicator A to update (Rule 3).

When Indicator C's bar is completed, it will trigger Indicator A if A and C are on the same time slot. It is possible that C lags behind A. In this case, C will not triggers A (Rule 4).

For order entry system where firing consistency is required, you should turn update dependency to primary link only (Rule 5).

Example 5: Indicator A linked to data (link 1) and Indicator B (link 2). Indicator A has Update by Tick option set to true. Indicator B is updated by tick

When Indicator B is updated, Indicator A is triggered to update (Rule 4).

Example 6: Indicator A linked to data (link 1) and Indicator B (link 2). Indicator A has Update by Tick option set to false. Indicator B is updated by tick

Indicator A is not updated when data arrives (Rule 3).

When Indicator B is updated, Indicator A is triggered to update (Rule 4).

Non-Update By Tick Indicators

When an indicator is not updated by tick, in real-time usage, the indicator value at a bar is updated only after the bar is completed, i.e. when a new tick arrives that triggers the creation of a new bar. When loading in historical data, the last bar of the historical data is considered as complete and indicator calculation is triggered. When a new tick arrives, if necessary, the indicator value at the last bar will be calculated again. In this case, the indicator value at the last bar is evaluated twice to ensure indicator value consistency.

For most users, this is desirable, especially for indicators that work on higher time frames and is updated in real-time.

Under some circumstances, this behavior may not be desirable. For example, if your indicator is custom-built and the code assumes that the indicator is calculated only once per bar, calculating the indicator value twice at the last bar can cause problem.

You can turn on/off this extra calculation option by:

1 Open the indicator editor to edit the indicator.

2 Under the **Other** tab, choose **One Extra Update On Completion of Last Bar**.

3 Press the **Apply** button.

Displacing Indicators Horizontally

You can displace an indicator horizontally. When an indicator is displaced, it is shift horizontally by certain number of bars. Displacing an indicator is useful for indicators that are used for price projections.

Displacing an indicator is only an visual effect. It does not affect the underlying calculation.

To adjust displacement:

1 Edit the indicator.

2 Press the **Visual** tab.

3 Change the **Displace** parameter.

Limitation

You cannot displace tick series in time driven charts. You can displace tick series only in bar driven charts (even bar space charts).

Indicator Pane Setting

When you add an indicator, you can adjust the pane settings on where the pane is create. Time chart provides many smart options for placing a new indicator in pane.

When you add indicator:

1 Press the **Visual** tab.

2 Change the **Pane** setting. You can let time chart decide which pane to place indicator, create a new pane, or create in an existing pane.

When you edit an indicator, you can place the indicator in any existing pane.

Indicator that Links to Different Time Frames

For indicators that can take multiple links (add, subtract, formula2, etc), you can link to sources that are of different time frames. For example, you can use formula2 to take the average of 3-day and 5-day MSFT.

When you mix two time frames together, the prefer setting is to use the lower time frame series as the first link (e.g. use 3-day as the first link if you are mixing 3-day and 5-day series). From our experience, this setting is most intuitive.

The indicator will be of the same time frame as the first link. The bars of other links will be resolved. For example, when you take average of 3-day and 5-day MSFT and set the 3-day series as the first link, the average will have a 3-day time frame. At each bar, the average will:

- Take every bar of the 3-day series.
- Resolve to the closest bar in the 5-day series that is chronologically earlier or equal to the 3-day bar.
- Perform the average calculation.

There are some additional issues if you mix real-time and EOD bars related to the time stamps of the EOD bars. See ***Last Bar Position during Real-time update for EOD Data Series*** (on page 1079) for more details.

Make Indicator Behaves Like Data Series

You can tell a multiple plot indicator to behave like a data series. The indicator will plot with candlesticks and behaves like a data series when you apply indicator on top of this virtual data series.

To set:

1. Edit the indicator.
2. Press the **Visual** tab.
3. Choose a setting under **Style**.
4. Press **Apply** button.

The styles available (called meta styles) depends on how many plots is in the indicator. For more information, see ***Indicator Meta Plot Style*** (on page 1465).

A tutorial is available to show you a concrete example, see ***Tutorial: Creating Your Own Index using Formula Language*** (on page 389).

Indicator Minimum Bars Setting

Some indicator requires a minimum number of bars before it become stable. The unstable period can distract from chart reading. You can tell such indicator to start calculation only after certain number of bars are available in the first link.

1 Edit the indicator.

2 Press the **Other** tab.

3 Set the parameter **Minimum Bars**.

4 Press the **Apply** button.

Recalculating Indicator

To recalculate an indicator (this is required when you modified the definition script/formula-based indicator), either:

- Choose **Refresh** from the pop up menu.
- Press the refresh button ⟳ on the tool bar.

Hiding and Showing Data Series and Indicators

You can hide a data series or indicator. Although a hidden series is not drawn, it is not removed from the chart. Calculations that depend on the series are still performed. You can always show the series again.

You can hide a visible object directly by:

1 Select the object to be hidden.

2 Choose **Hide** from the pop up menu.

Alternatively, you can use ***Chart Manager*** (on page 1013) to hide a series. To hide a series using Chart Manager, press the eye glasses besides the series in the chart manager window.

To show a hidden series, press the eye glasses besides the hidden object in Chart Manager.

Control Days to Load for Individual Data Series

You can also control days to load for a data series with user panel

You can control the number of days to load for individual data series. By default, a data series follows the chart's number of days to load. You can specify the days to load for individual data series:

1 Select the data series and press the space bar to open the editor.

2 Press the **Misc** tab. If you don't see the **Misc** tab, press the **More Options** button.

3 Check **Override Days to Load** and fill in the days. in the data series' edit window (launched by selecting the data series and press the space bar).

4 Press the **Apply** button.

Controlling days to load is useful when you want to plot data series and indicators of very different time period in a single chart. For example, you want to use a 200-day daily moving average as a reference in a minute chart. In this case, you can setup a 1-day minute chart, add the daily data series, change the daily data series' days to load to more than 200 days, and apply the moving average indicator.

Calendar Days vs. Trading Days

Individual data series follows the calendar days/trading days setting for the whole chart. See ***Days of Data to Load*** (on page 1029).

Part of Pane Scale

You can control whether a data series or indicator series should be included when calculating the pane scale (the y-axis value range of the pane). This feature comes in handy when you want to enforce focus on one data series in a pane where the indicators could be too far away from the actual price.

1 Select the data series or indicator.

2 Right click to open the pop up menu and choose **Part of Pane Scale**.

Float Markers

Float markers are the price value tags on the price axis. To turn on float markers:

1 Make sure the float markers are on for the whole chart. Un-select everything, in pop up menu, make sure **Visual>Floating Marker** is checked.

2 To turn on/off the float marker for data series and indicators, select the data series/indicator, from the pop up menu, choose **Auto Float Marker**.

Setting up Float Markers Visual Style

Float markers visual style is user customizable.

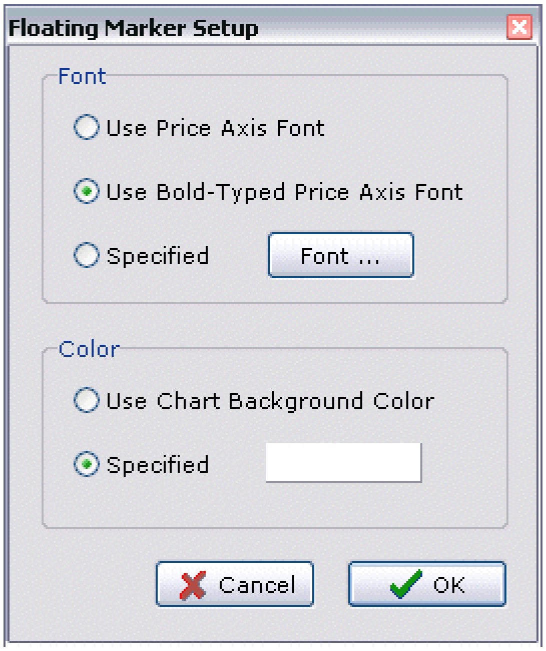
1 Open Chart Manager.

2 Press the **Visual 1** tab.

3 Select **Floating Markers** in the list above the **Change** button.

4 Press the **Change** button.

A dialog will be opened to let you edit float marker visual styles.



Floating Marker Setup

Font

○ Use Price Axis Font

● Use Bold-Typed Price Axis Font

○ Specified

Color

○ Use Chart Background Color

● Specified

Customize the Look of Data Series

Overall data series look is set in the chart manager, under the **Visual 1** tab. These settings are applied to all data series.

To customize the look of individual data series.

1 Select the data series you want to change. Press the space bar to edit the data.

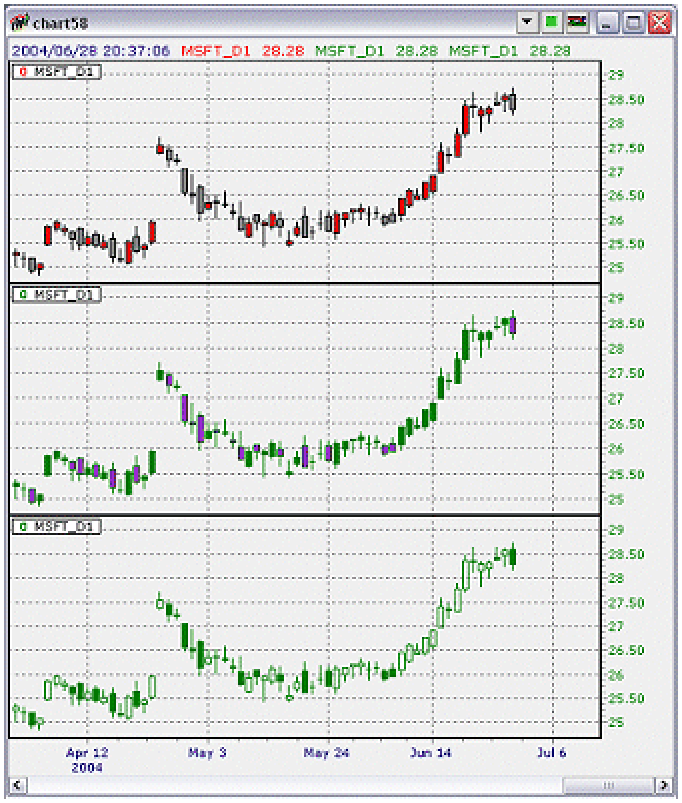
2 In the editor, you should see several tabs for setting options: **Color**, **Bar Style**, **Candle Style**. If you don't see these tabs in the editor, press the **Options** button.

3 Under the **Color** tab, you can change the color scheme of the data series.

4 Under the **Bar Style** tab, you can change the physical appearance if your data series is plotted using **Bar** style.

5 Under the **Candle Style** tab, you can change the physical appearance if your data series is plotted using **Candle** style.

The second data series shows custom colors (up bar is green, down bar is purple). The third data series shows hollow style.



Color Tab

See ***Color Data Series*** (on page 1111).

Bar Style Tab

In bar style, a bar has open marker, close marker and body. Under this tab, you can adjust the visibility and sizing of the items.

To adjust color of a bar, use the **Color** tab.

Candle Style Tab

In candle style, a candle has line, border and body. Under this tab, you can adjust the visibility and sizing of the items.

To adjust color of a candle, use the **Color** tab.

Color Data Series

To make changes to the coloring of individual data series:

1 Select the data series you want to change. Press the space bar to edit the data.

2 Press the **Color** tab. If you don't see these tabs in the editor, press the **Options** button.

Changing color does not require you to re-apply the data series. Changes are interactive and you can view the changes right away in the chart.

What can be Changed - Body and Add-on's

A data series can be displayed in many visual styles: Candle, Bar, Line, Dot, Box, Square, Range, Volume, Open Int, Tick Vol, P+F, Kagi, 3LB, Flex Blocks, Flex Kagi, Flex P+F.

All these visual style has one thing in common. They have something to draw. The central piece of item to be drawn is known as the body. For example, the body of line style is the line itself. The body of P+F is the circles and crosses. When you modify the color of the body, you have the most flexibility in terms of coloring.

Three visual style have add-on's: candle, bar, box. The add-ons are colored separately from the body. Candle style add-on's are border and extended line. Bar style add-on's are up marker and down marker. Box style add-on is border.

Changing Body Color

To change body coloring, change the setting to **Formula** under **Body Coloring** group. Then press the down triangle and select one of the pre-defined coloring formula.

Formula	Meaning
By Day	Change color when day changes
By Hour	Change color when hour changes
By MidDay	Change color after mid day
By Minute	Change color after minute changes
By Month	Change color after month changes
By Quarter	Change color after quarter changes
By Week	Change color after week changes
By Year	Change color after year changes
Positive Negative	Use one color for positive value. One color for negative value
Trend 4 Bars	One color for up trend (value larger than the low of 4 bars ago). One color for down trend (value lower than the high of 4 bars ago)
Up Down	Use one color when value goes up. One color when value goes down

Writing Body Color Formula

You can create your own body coloring rule by using formula:

1 Choose a body color formula to start with

2 Make modifications by pressing the ... button

3 Press the down triangle button and choose **Save As**, choose a name for your rule

After you made changes to the formula, you need to press the **Refresh** button to refresh the data series.

Formula is a simple expression returning different colors based on conditions you specified. Body color formula supports all generic functions and have the following additional context functions.

Long Form	Short Form	Meaning
`Open`	`O`	Open price of bar
`High`	`H`	High price of bar
`Low`	`L`	Low price of bar
`Close`	`C`	Close price of bar
`Volume`	`V`	Volume of bar
`Tick`	`T`	Number of ticks in bar
`OpenInt`	`OI`	Open interest of bar
`Date`	No short form	Date stamp of the bar
`Time`	No short form	Time stamp of the bar
`DateTime`	No short form	Date time stamp of the bar
`FlatColor`	No short form	Flat color set in color table
`UpColor`	No short form	Up color set in color table
`DownColor`	No short form	Down color set in color table
`PreviousColor`	No short form	Previous color of the bar
`DefaultColor`	No short form	Default color of the series

Consider the formula for Trend 4 Bar as an example:

```
' Trend 4 Bars
if (c > l (4), UpColor,
   if (c < h (4), DownColor,
   PreviousColor)
)
```

The formula compares the close with the low and high of 4 bars ago. If the close is higher than the low, UpColor is returned, marking an uptrend. If the close is lower than the high, DownColor is returned. If neither condition matches, it returns the previous color, marking the continuation of previous trend.

For general discussion on formula, see ***Understanding Formulas*** (on page 257).

Color Table

A data series has a default color scheme. It uses the default color (color that usually gets your attention) plus a default down color (usually gray) to color bars.

Color Table provides more coloring options than the default coloring scheme can provide. The extra colors are used by:

- Body color formula as FlatColor, UpColor and DownColor
- Visual styles that have add-on's, e.g. candle, bar
- Other visual styles

To use Color Table for body color formula, you must enable **Formula** under the **Body Coloring** group. See the section above on how to select and modify body color formulas.

To use Color Table for multi-color and add-ons, you must enable **Color Table** under **Color Scheme**. The following table lists how coloring works when Color Table is enabled for add-on's and multi-colors.

Visual Style	What happens when Color Table is enabled
Candle	Candle border uses flat color. Up candle uses up color. Down candle uses down color. Note: up/down color can be over-ridden by body coloring
Bar	Bar body uses flat color. Up marker uses up color. Down marker uses down color. Note: bar body can be over-ridden by body coloring
Others	Visual styles use Color Table colors over default color

Candle and Box Options

Candle and box styles can set hollow style by checking **Hollow** under **Candle / Box Body**.

When using hollow style, candle/box are not filled. Instead, body color is used to draw borders.

For candles, you can set the extended line of candles to draw using either body or border color. This option is under **Candle Ext Line**.

Break on Invalid Bars

This option is applicable only to line and square style. When **Break on Invalid Bar** is set, lines are broken at invalid data points (e.g. data with no trades).

Color Indicators

To make changes to the coloring of individual indicator:

1 Select the indicator you want to change. Press the space bar to edit the indicator.

2 Press the **Color** tab.

Changing color does not require you to re-apply the indicator. Changes are interactive and you can view the changes right away in the chart.

Style

Choose **Formula** to enable indicator coloring.

Def, Up, Dn

Settings for default color, up color and down color.

Formula

Press the down triangle button to choose from a pre-built coloring rule. The meaning of the color formula is listed below.

Formula	Meaning
20-80	Use default color for value between 20-80. Down color for value below 20. Up color for value above 80
30-70	Use default color for value between 30-70. Down color for value below 30. Up color for value above 70
Above Below Plot 1	Use up color if value is above value of plot 1. Use down color otherwise
Above Below Plot 2	Use up color if value is above value of plot 2. Use down color otherwise
Positive Negative	Use up color if value is positive. Use down color if value is negative
Up Down	Use up color if value is going up. Use down color if value is going down

Writing Indicator Color Formula

You can create your own indicator coloring rule by using formula:

1. Choose a indicator color formula to start with
2. Make modifications by pressing the ... button
3. Press the down triangle button and choose **Save As**, choose a name for your rule

Formula is a simple expression returning different colors based on conditions you specified. Indicator color formula supports all generic functions and have the following additional context functions.

Function	Meaning
`Plot`	Value of current plot
`PlotN`	Value of a specific plot, e.g. Plot1, Plot2, etc.
`Value`	Same as Plot
`ValueN`	Same as PlotN
`Date`	Date stamp of the bar
`Time`	Time stamp of the bar
`DateTime`	Date time stamp of the bar
`UpColor`	Up color
`DownColor`	Down color
`PreviousColor`	Previous color of the indicator
`DefaultColor`	Default color

Consider the formula for Above/Below Plot 1 as an example:

```
' Above/Below Plot 1
if (plot >= plot1, UpColor,
   if (plot < plot1, DownColor,
      PreviousColor
   )
)
```

The formula compares the current plot value with value of plot 1. If current plot value is above plot 1, UpColor is returned. If current plot value is below plot 1, DownColor is returned. If neither condition matches, it returns the previous color, marking the continuation of previous trend.

For general discussion on formula, see ***Understanding Formulas*** (on page 257).

Hollow Candlesticks

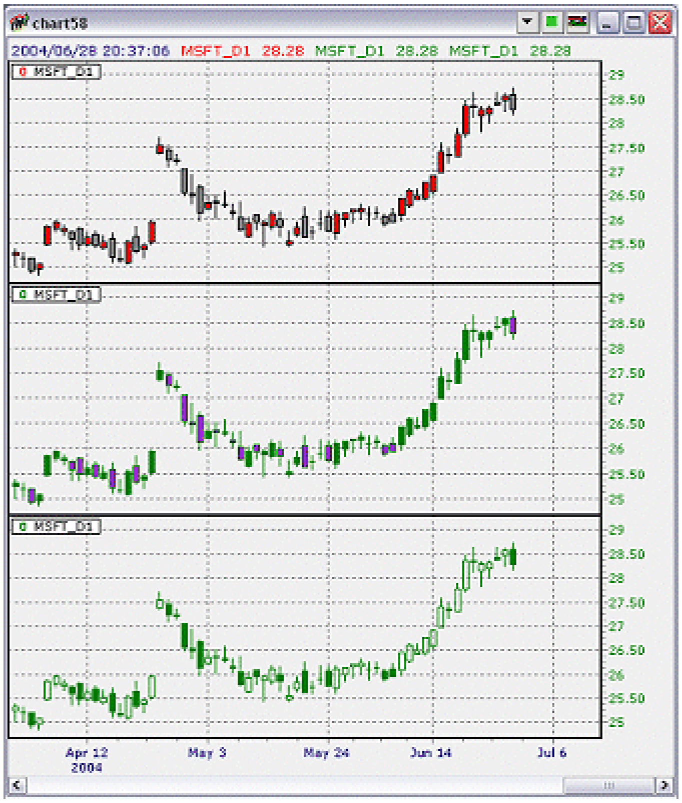
Some traders prefer a hollow candlestick style (monolithic in color). To turn on hollow candlestick:

1 Select the data series you want to change. Press the space bar to edit the data.

2 In the editor, press the **Color** tab. If you don't see the **Color** tab in the editor, press the **Options** button.

3 Under **Candle / Box Body**, check **Hollow** option.

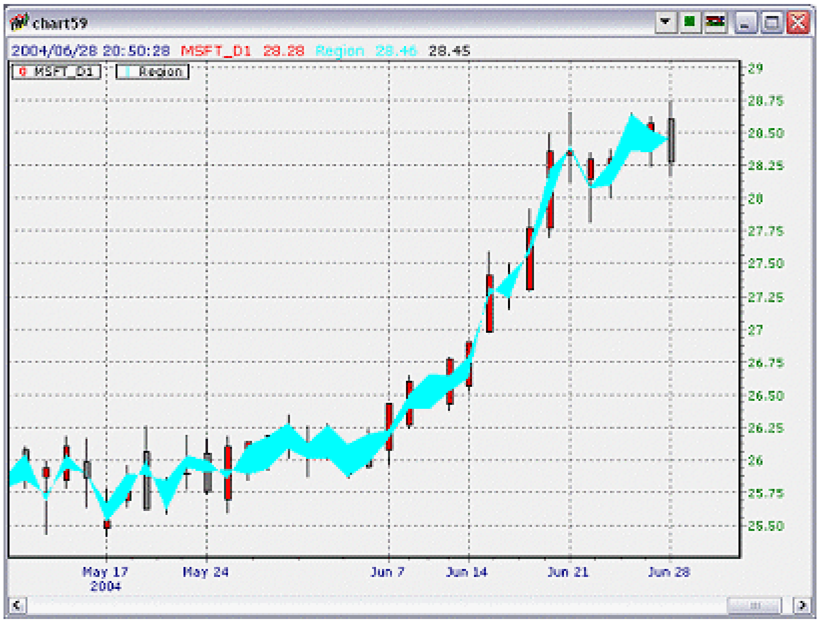
The third data series in the chart is using hollow candles (up is in hollow, down is in red).



Re-arranging the Display Order of Data Series and Indicators

You can specify the display order of data series and indicators in a chart. Display order refers to the order of the data series and indicators when they are being drawn on the chart. Items at the bottom of the displayed order is drawn first, then covered by items that are drawn later. Changing display order is useful when the indicator's visual style makes it visually obscuring to other items on the chart and you want to make other items visible.

Consider the following chart. The lines of the indicator is very thick and is obscuring the data underneath it.



To re-order the display order, choose **Object Ordering** from the pop up menu.

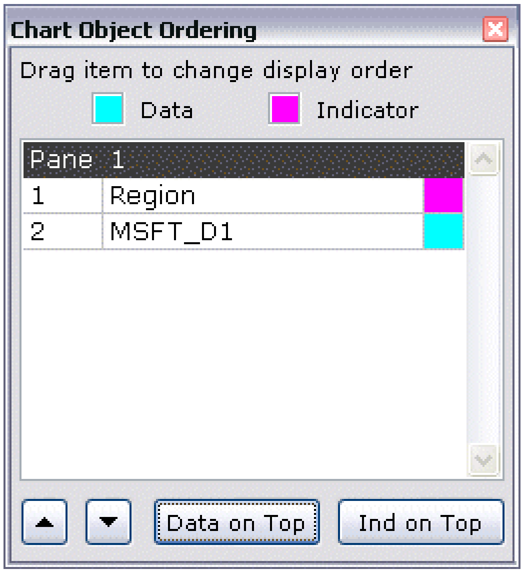


Chart Object Ordering

Drag item to change display order

🟦 Data 🟪 Indicator

Pane 1

1	Region	🟪
2	MSFT_D1	🟦

Now click on the indicator in the window and press the down arrow. The indicator will be moved below the data.

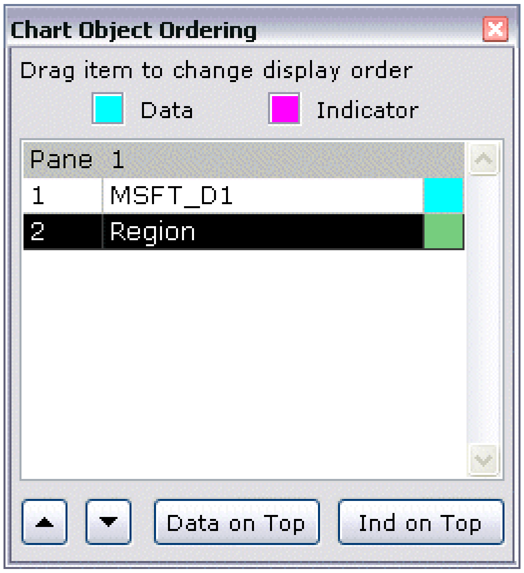
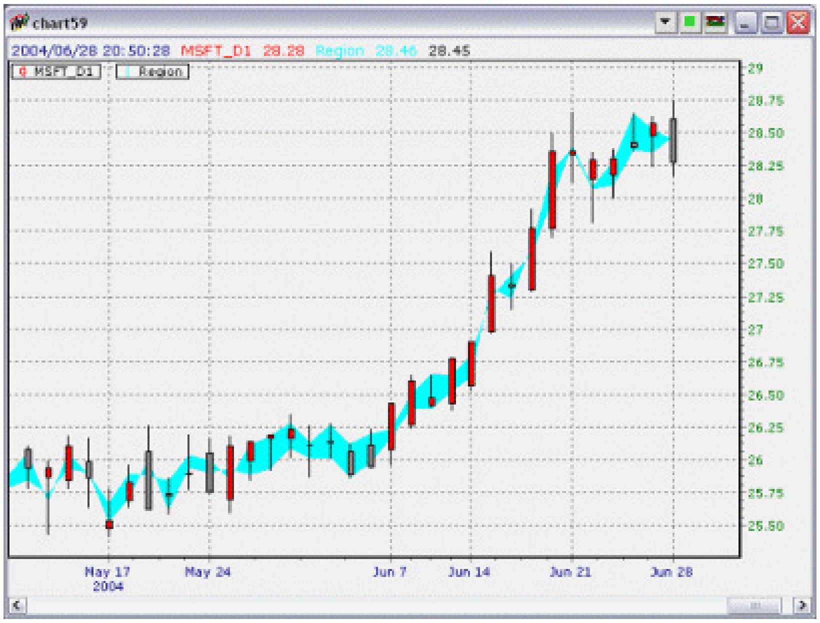


Chart Object Ordering

Drag item to change display order

🟦 Data 🟪 Indicator

Pane 1		
1	MSFT_D1	🟦
2	Region	🟩



Set Object Ordering

If a data series or indicator is selected, you can right click to open the pop up menu, and choose one of the item under **Set Object Ordering** to send the selected data series/indicator to front or back.

Specifying Tick Filtering for Data Series

Ticks that are coming in real time are often noisy and can affect chart display and trading systems if the noise level is high. The amount of noise depends on data vendor and the market condition.

A general tick filter is available to filter out a lot of the noisy already. For more information, refer to ***Bad Ticks Filtering*** (on page 568)

In a chart, you can specify a formula-based, highly customizable tick filter for data series. To enable tick filter in a chart:

1 Select the data series you want to apply

2 Press SPACE BAR to edit the data series

3 Press the **More Options** button

4 Press the **Misc** tab

5 Choose the tick filter in the **Real-Time Tick Filter** drop down

6 Press the **Apply** button

Tick filter will repair ticks as it comes in, and will attempt to correct existing bar. To apply tick filter on existing bars, you need to reconstruct the minute bars from ticks, that way, the ticks will go through the tick filter. Here is the procedure:

1 In the edit data dialog, **Misc** tab, check **Enable Minute Bar Load by Tick**.

2 Press the **Apply** button.

Tick filters are highly user customizable and can do much more than filtering bad ticks. For instance, you can filter out small volume trades to look only at trades from the "big players". Tick filter can even modify time stamp information and perform time warping on charts.

Minute Bar Load by Tick

You can force a data series to construct its minute bar by tick.

This feature is useful if your data vendor does not provide clean minute data, or when you need accurate ticks for trading system development.

> Re-constructing minute bar from ticks are extremely time consuming and should be used carefully if you use an Internet data feed.

To enable this feature:

1 Open editor for data series.

2 Press the **More Options** button to reveal all options in the editor.

3. Press the **Misc** tab.
4. Check the **Minute bar load by tick** option.
5. Press the **Apply** button.

The data series will be reconstructed using tick data rather than from the historical minute bar database.

The setting is for the tick data that will come in. To force historical minute data to be re-constructed using tick data, use the replay by tick feature. See ***Replay by Tick*** (on page 1123).

Replay by Tick

This feature reconstructs the historical bar tick-by-tick without using minute bar data. This exactly simulates how the chart behaves in real-time. Indicators that rely on tick-by-tick behavior will work exactly the same for historical data and real-time data.

1. Open chart manager.
2. Press the **Misc** tab.
3. Press the **Replay** button.

Completely Reloading all Data

You can completely reload all data series and recalculate all indicators by pressing the complete reload button on the tool bar. Complete reload is necessary only if you have aborted an earlier data loading operation and the data is incomplete.

You can also reload under the **Misc** tab in chart manager.

Broken Lines

Break on Invalid Data

If you set the visual style to line or square, the line is continuous, i.e. there is no breaking point in the line.

You can change the line to not connecting between data points.

For data series, you can choose to have the line not connected between invalid data points (for example, in data series, when there is no trade):

1. Select data series, press space bar to edit.
2. Press the **Color** tab. If you don't see the **Color** tab, press the **Options** button.
3. Check **Break on Invalid Bar** under **Misc**.

For indicator, you can choose to have the line not connected between invalid points, or when the indicator specified break points.

1. Select indicator, press space bar to edit.
2. Press the **Visual** tab.
3. Under the **Break** column, for break on invalid, choose **Invalid**. For break on indicator specified break points choose **Visual**. To disable this option, choose **None**.

This option is interactive. You do not need to press **Apply** button to make the setting effective. This option applies to line and square style only.

Indicators that Set Visual Break

Some indicators are written to set visual break themselves (by setting the `ItSelf.VisualBreak` property). To use these indicators in the chart, you will need to set the breaking property to **Visual**. To find out how to program these indicators, see the `VisualBreak` and `VisualBreakEx` properties in ***ItSelf Object, Properties and Methods*** (see "Properties and Methods" on page 1525).

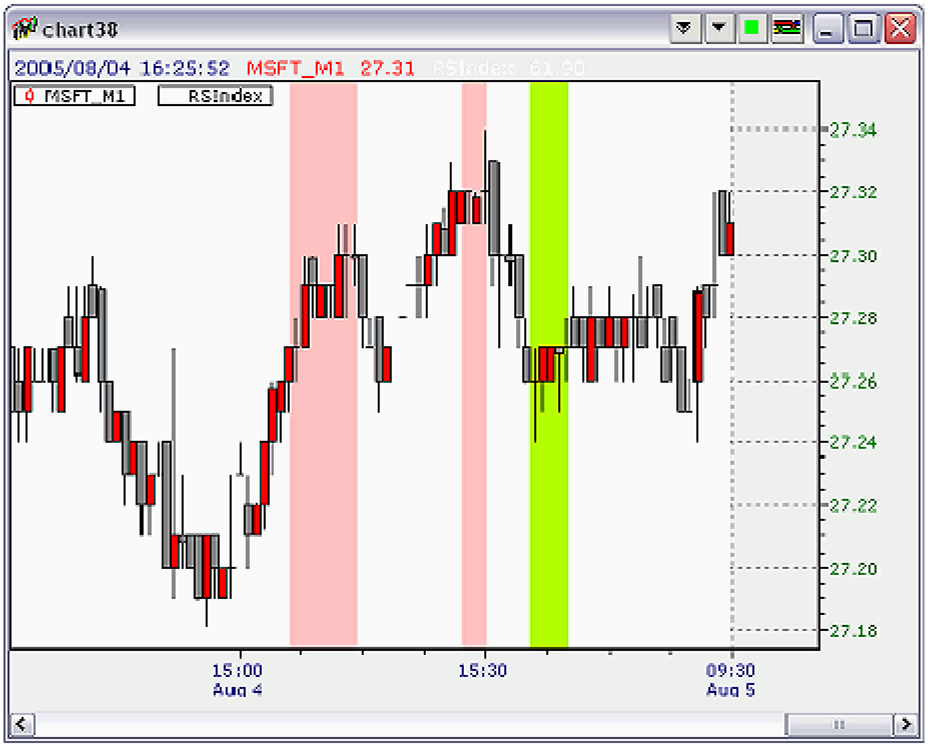
Background Drawing Style for Indicator

You can select the **Backgrd** drawing style for an indicator, for example in the indicator legend. A background drawing style make the indicator acts as a tool to highlight chart background.

The following is an example:

1. In a chart, add MSFT and RSI to the chart. Put them in the same pane.
2. Make RSI not part of pane scale by disabling **Part Of Pane Scale** in RSI's pop up menu.
3. Edit RSI by opening the **Edit Indicator** dialog.
4. Under **Color** tab, change **Style** to **Formula**, **Def** to white, **Up** to light red, **Down** to light green, **Formula** to **20-80**. This will color the RSI indicator according to the 20-80 rule.
5. Under **Visual** tab, change **Style** to **Backgrd**. Press **Apply** button.
6. In the chart, select RSI, right click to open pop up menu, and choose Set **Object Ordering>Send To Back**.

In this setup, MSFT is highlighted in light red when RSI goes above 80, highlighted in light green when RSI goes below 20, as shown in figure below.



Drag and Drop

You can drag and drop data series and indicators between panes.

You can drag and drop data series between charts.

Drawing Tools

Drawing tools are objects you draw on the chart to help you to read a chart. For example, you can draw trend lines and trend channels on a chart.

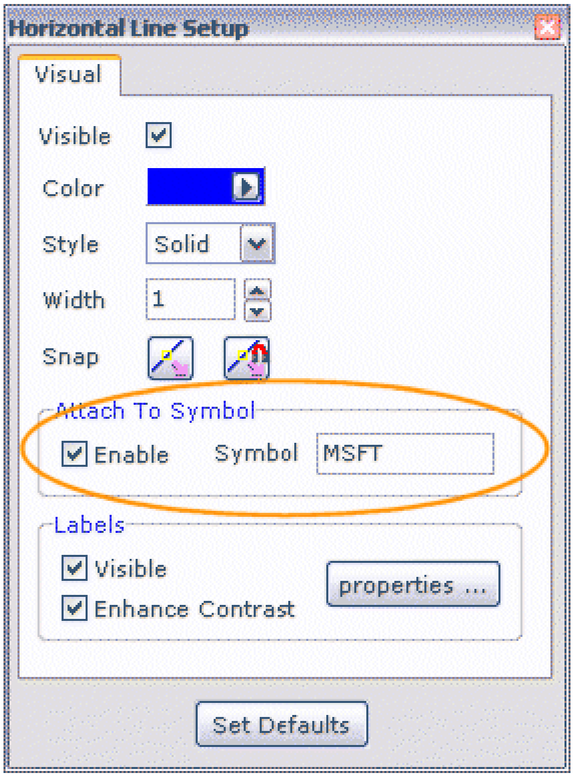
Attaching to Symbol

By default, when you draw a drawing tool, it is attached to pane's first data series symbol. If the pane does not have a symbol, the drawing tool will attach to the chart's first data series symbol.

If you switch symbol, the drawing tool will be hidden. Note that a hidden drawing tool is still listed in Chart Manager. The visibility can very well be on, but because the symbol in the chart is no longer the one the drawing tool attaches to, it is not shown in the chart.

If you switch back to the original symbol, the drawing tool will be shown again.

To adjust symbol attachment, edit the drawing tool, the settings are under **Visual** tab, **Attach to Symbol** group.



Horizontal Line Setup

Visual

Visible ☑

Color

Style Solid

Width 1

Snap

Attach To Symbol

☑ Enable Symbol MSFT

Labels

☑ Visible

☑ Enhance Contrast

You can set preference such that drawing tools are not attached to any symbol by default. When a drawing tool is not attached to a symbol, whether the drawing tool is shown will be determined by the drawing tool's visibility setting.

To turn on/off attachment preference:

1 In main window, choose **Program>User** Preference.

2 Press **Time Chart** tab.

3 Under **Drawing Tools** group, toggle **Attach drawing tools to symbol**.

The following table illustrates whether a drawing tool is shown under different attachment/visibility combinations.

Attached to symbol	Is attached symbol in the chart	Is the drawing tool visible	Chart will show drawing tool
Yes	Yes	Yes	**Yes**
Yes	Yes	No	**No**
Yes	No	Yes/No	**No**
No	N/A	Yes	**Yes**
No	N/A	No	**No**

Snapping

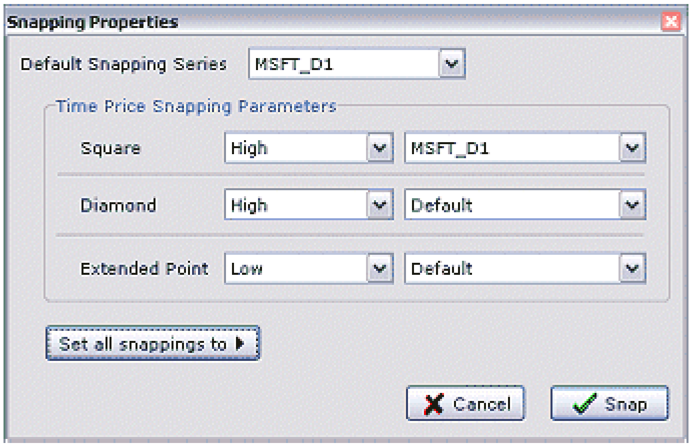
Handles of drawing tools can be snapped to data series and indicators. You can snap a handle once, or set the snapping property of the handle such that the handle is snapped as you move the drawing object.

To snap, either:

- Choose Snap or Snapping Properties from the pop up menu when the drawing tool is selected, or
- Press the snap button or snap properties button in the edit dialog of the drawing tool.

A dialog will open to let you edit the snapping.

You can specify a default series for snapping. The default series is the data series or indicator the drawing object snaps to. Snapping properties for individual handles can be set. You can also specify a different series to snap to for each individual control. You can snap a drawing tool to multiple series, as long as the series are in the same pane.



Snapping Properties

Default Snapping Series: MSFT_D1

Time Price Snapping Parameters

Square	High	MSFT_D1
Diamond	High	Default
Extended Point	Low	Default

You can set the following snapping styles:

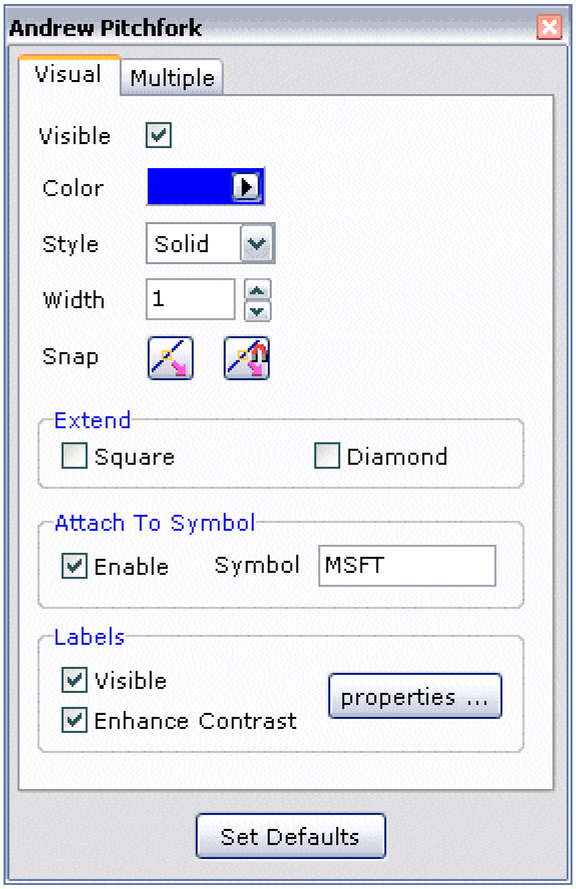
Snapping Style	Meaning
None	Nothing
Time Only	Time values on the time axis
Value	Indicator value or data series close
Open	Open
High	High
Low	Low
Close	Close
Mid High Low	Middle between high and low
Mid Open Close	Middle between open and close
Average OHLC	Average of open, high, low and close
Average HLC	Average of high, low and close
Extreme	High or low, depends on which value is closer

Editing a Drawing Tool

To edit a draw tool, either:

- Select the drawing tool and choose **Edit** from the pop up menu, or
- Double click on the drawing tool.

A dialog will open to let you edit the properties of the drawing tool. The dialog will look slightly different for different drawing tools.



Andrew Pitchfork

Visual | Multiple

Visible ☑

Color

Style Solid

Width 1

Snap

Extend

□ Square □ Diamond

Attach To Symbol

☑ Enable Symbol MSFT

Labels

☑ Visible

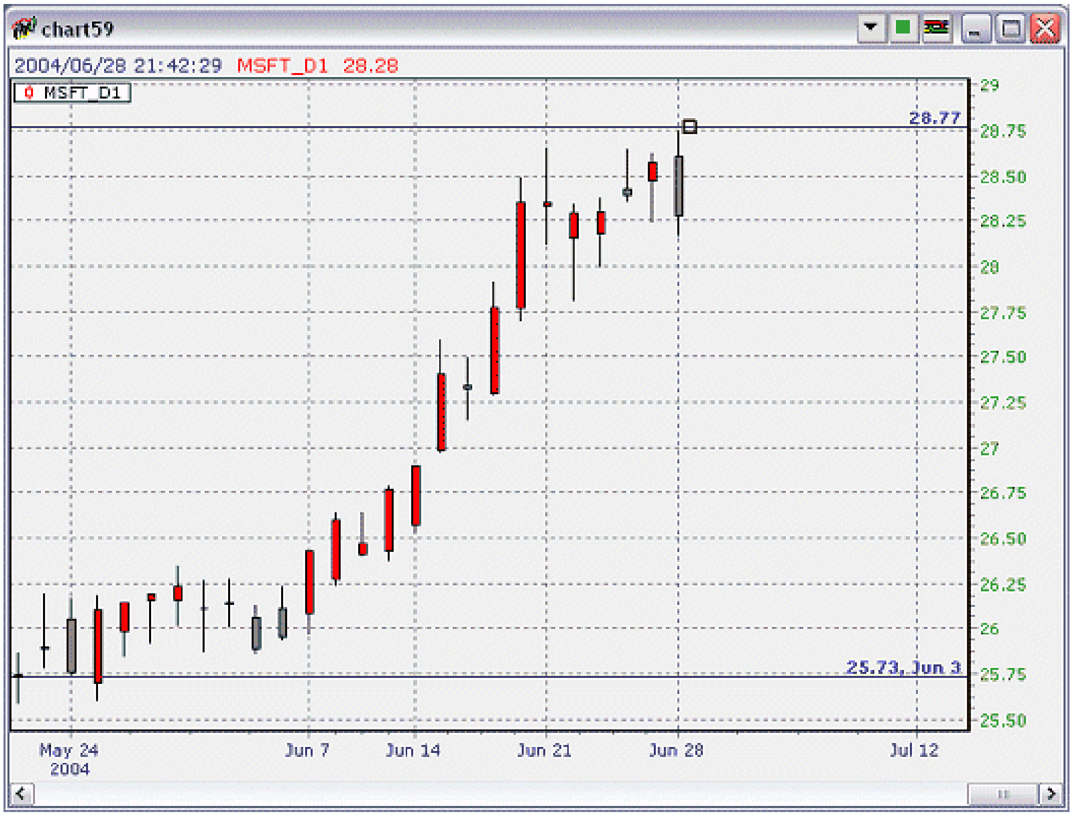
☑ Enhance Contrast

All drawing tool has a **Visual** tab for visual settings. The table below explains the settings and buttons under this tab (not all settings are available to all drawing tools).

Setting/Button	Meaning
Visible	Whether the drawing tool is visible
Color	The color of the drawing tool
Style	The drawing style of the fan lines. It is one of solid, dot or dash. Only solid style is support for width larger than 1.
Width	Width of drawing tool's lines
Fill Color	For drawing tools that has an internal area (e.g. rectangle). This setting specifies the fill color.
Square	Extends the square end
Diamond	Extends the diamond end
Attach to Symbol	Drawing tool is visible only when the symbol it attaches to is in the chart
Border	In some drawing tools, you can turn on/off the border.
Fill	In some drawing tools, you can turn on/off the fill.
Label/Visible	Whether the labels are visible.
Label/Enhance Contrast	Whether the labels are drawn using a high contrast color to chart background.
Label/Properties	Edit the label properties
	Snap the drawing tool
	Edit the snapping properties of the drawing tool
Set Defaults	Press this button to save the settings to the current drawing tool template.

Configuring Drawing Tool Labels

In the following chart, two horizontal lines have been drawn, each with labels.

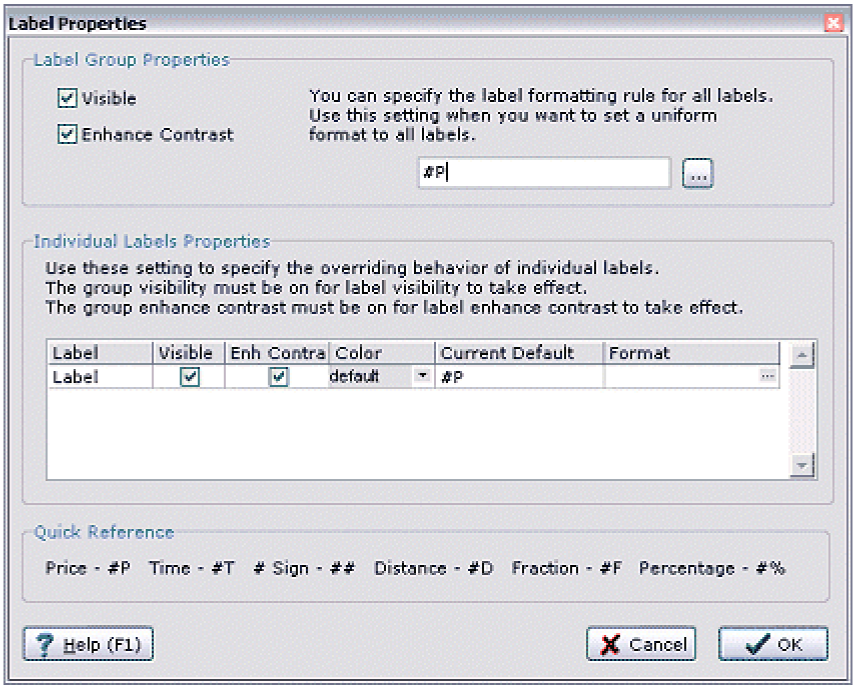


Notice that the labels are different for the two lines. The upper line shows a price, while the lower line shows both price and time (the time corresponds to where the handle is, for the horizontal line).

The labels in a drawing tool are customizable:

1. Select the drawing tool and press the space bar to open the editor.
2. Press the **properties** button under the **Labels** group in the editor.
3. Enter the format code under **Label Group Properties**. This is the default format code for all labels.

Alternatively, you can choose from the drawing tool's pop up menu, **Label Properties**, to edit label properties.



Label Properties

Label Group Properties

☑ Visible

☑ Enhance Contrast

You can specify the label formatting rule for all labels. Use this setting when you want to set a uniform format to all labels.

#P

Individual Labels Properties

Use these setting to specify the overriding behavior of individual labels.
The group visibility must be on for label visibility to take effect.
The group enhance contrast must be on for label enhance contrast to take effect.

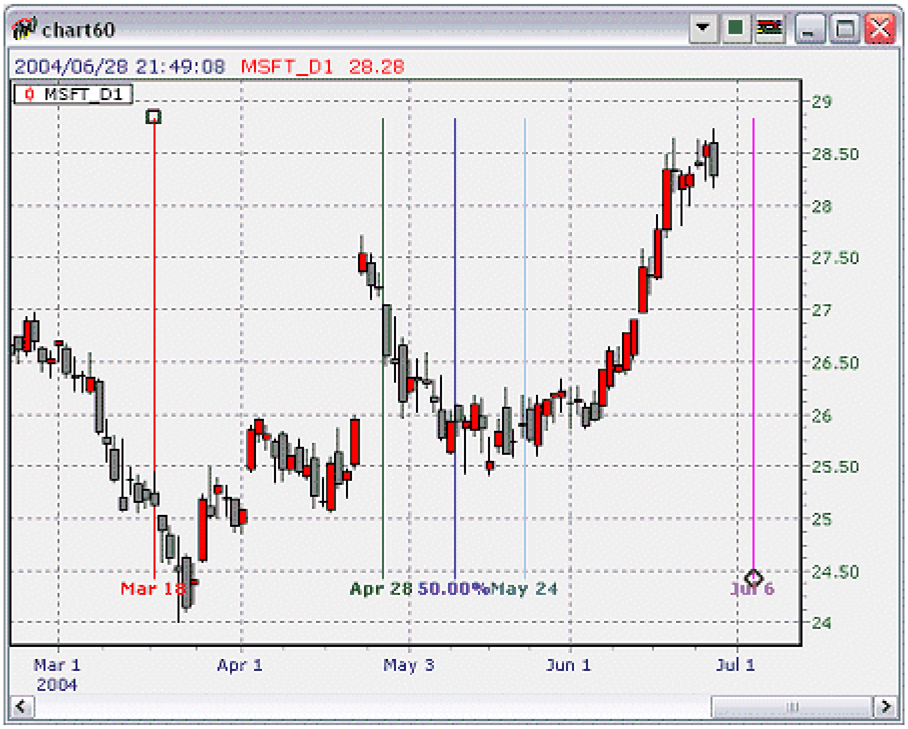
Label	Visible	Enh Contra	Color	Current Default	Format
Label	☑	☑	default	#P	

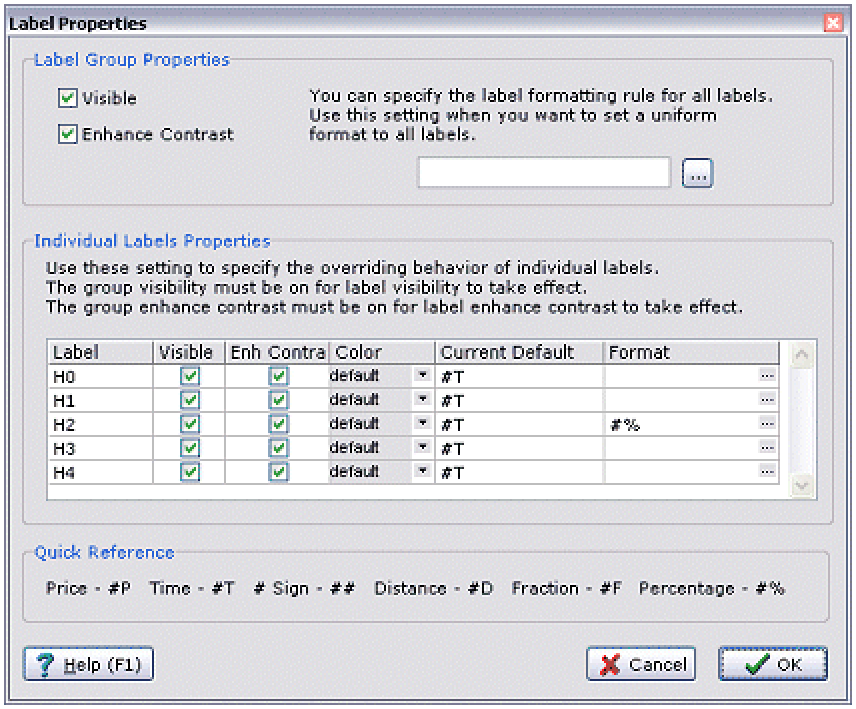
Quick Reference

Price - #P Time - #T # Sign - ## Distance - #D Fraction - #F Percentage - #%

For example, if you want to display price and time, you can use the format `#P, #T`. You can either type in the format code directly, or use the label format wizard to help you construct label format. To open the label format wizard, just press the ... button.

For multiple lines drawing objects such as Fibonacci time, you can choose to format all the labels at once, or override the label for individual labels. In the example below, only the middle line is formatted as percentage (50%), others use the default time format (`#T`).





Label Properties

Label Group Properties

☑ Visible

☑ Enhance Contrast

You can specify the label formatting rule for all labels. Use this setting when you want to set a uniform format to all labels.

Individual Labels Properties

Use these setting to specify the overriding behavior of individual labels. The group visibility must be on for label visibility to take effect. The group enhance contrast must be on for label enhance contrast to take effect.

Label	Visible	Enh Contra	Color	Current Default	Format
H0	☑	☑	default	#T	
H1	☑	☑	default	#T	
H2	☑	☑	default	#T	#%
H3	☑	☑	default	#T	
H4	☑	☑	default	#T	

Quick Reference

Price - #P Time - #T # Sign - ## Distance - #D Fraction - #F Percentage - #%

Copying Drawing Tools

To copy a drawing tool:

1 Select the drawing tool.

2 Right click to open the pop up menu.

3 Choose either **Copy** or **Copy with Snapping Properties**.

If you choose **Copy**, the selected drawing tool will be copied without snapping properties. This is action is desirable if snapping properties can affect the shape of the drawing tool that is being copied.

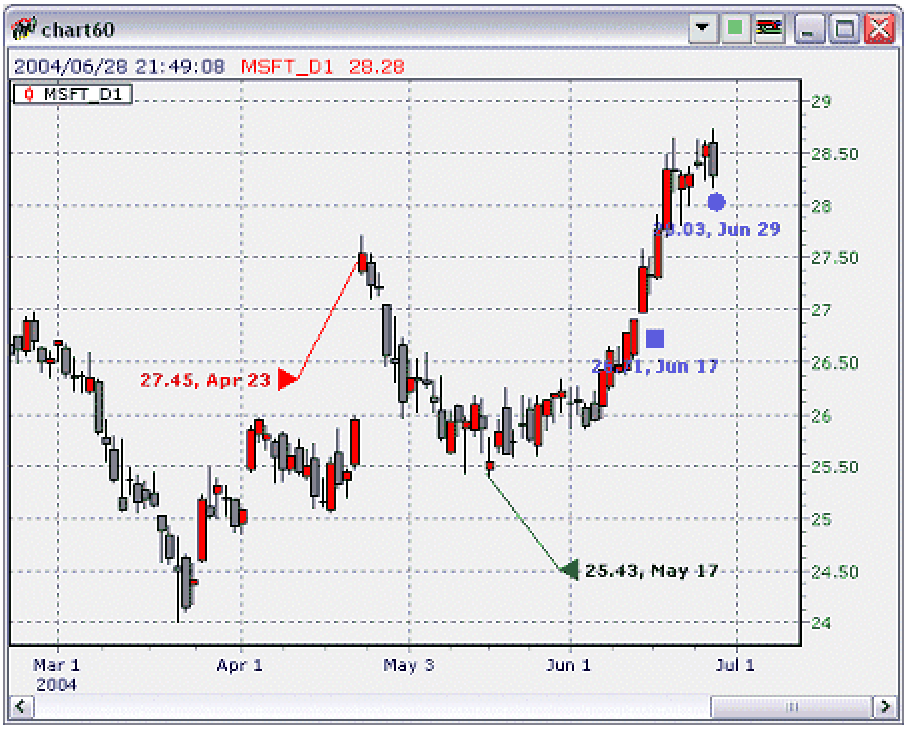
If you choose **Copy with Snapping Properties**, snapping properties are copied. Copying snapping properties can make the copy in a shape different from the original.

You can also use Chart Manager to copy drawing tools.

Offsetting

The following drawing tools support offsetting: rectangle, ellipse, text, marker. Offsetting allows you to offset a drawing tool from its origin to make the chart easier to read. When a drawing tool is offset, the drawing tool labels show the values of the tool at the origin.

The following figure illustrates offsetting markers:



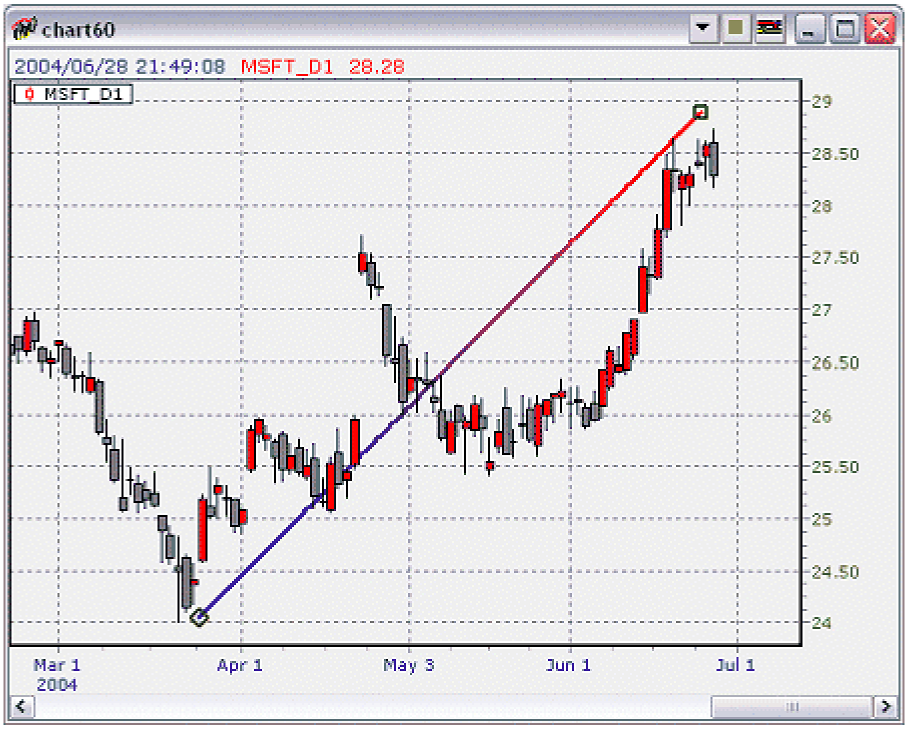
The red and green triangle markers are offset from their origin. The blue dot and rectangle are not offset.

To offset a drawing tool, either:

- Hold the SHIFT key, then drag the drawing tool, or
- Adjust the offsetting values in the drawing tool's editor

Vignette Lines

Vignette lines are lines that has a gradual change in color. It can be used for all straight lines in drawing tools.

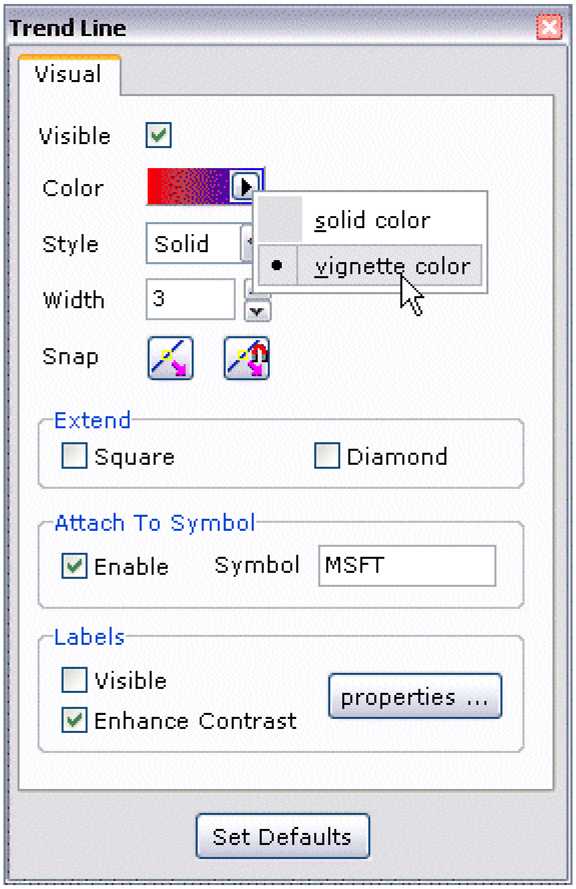


As illustrated in the chart, the vignette line changes color from red to blue. The technique is very useful for traders who needs to quickly glance a chart to get a sense of change and trend-line directions without the use of explicit labels or arrow heads.

Vignetting has been used successfully in many fields of illustration and NeoTicker® is the first application to introduce this technique to technical charting. By allowing the color to change within a line, an extra dimension of information is presented to the user.

Editing Vignette Lines

You can edit the vignette lines in the drawing object's editor.



Trend Line

Visual

Visible ☑

Color

Style Solid

Width 3

Snap

Extend

□ Square □ Diamond

Attach To Symbol

☑ Enable Symbol MSFT

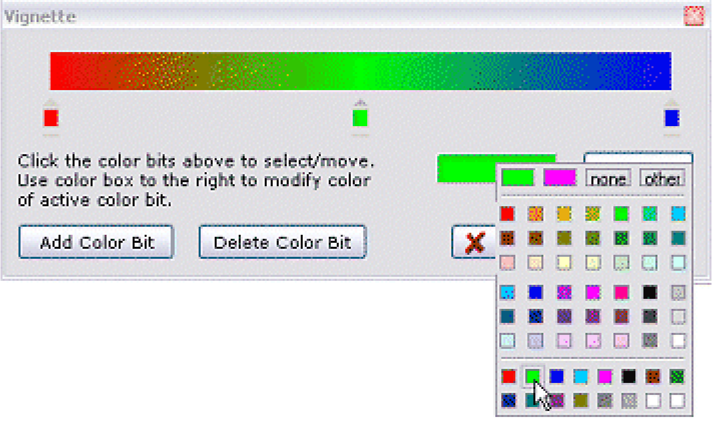
Labels

□ Visible

☑ Enhance Contrast

Notice the right pointing triangle in the color field. When you click on it, you can change the color mode of the drawing object, i.e. between solid color and vignette color.

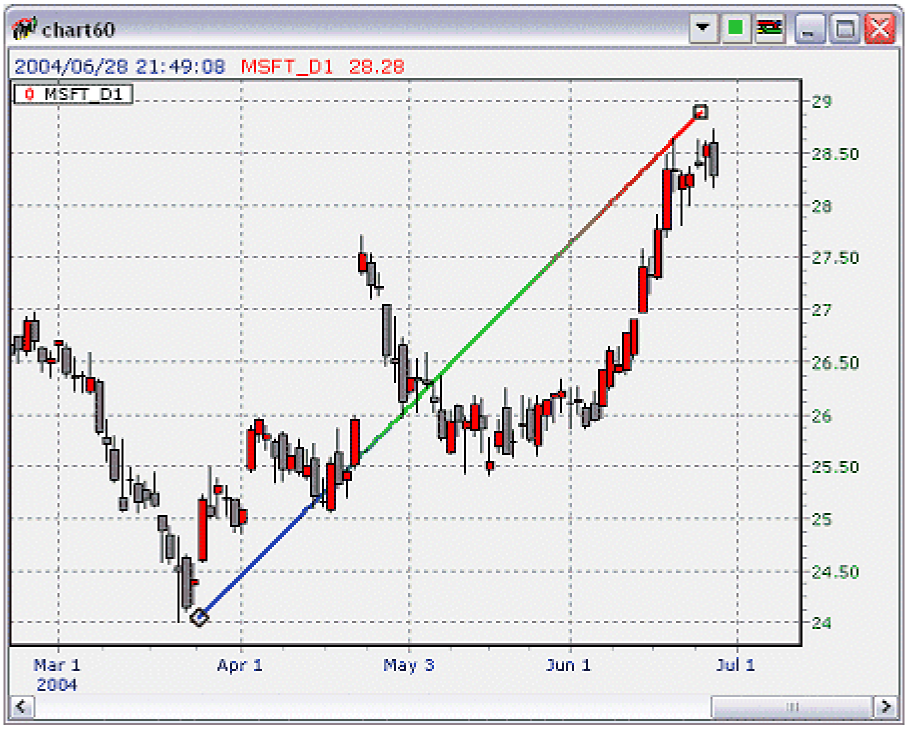
After you have changed to vignette mode, if you click on the color field itself (other than the triangle), an editor will be opened to let you edit the vignette.



Click the color bits above to select/move. Use color box to the right to modify color of active color bit.

The little objects below the color bar are color bits. You can click on a color bit and use the color field to change its color. You can add new color bits and remove color bits from the vignette. Furthermore, you can drag and move the color bits to change the color gradient.

The following chart shows a vignette line that changes from red to green to blue.



Setting up Lines for Multi-line Drawing Tool

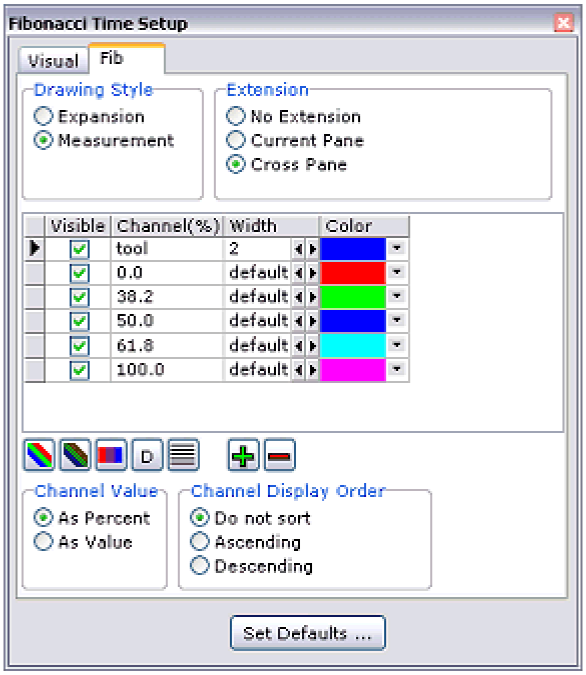
Quick Setup

NeoTicker® allows you to quickly modify a multi-line drawing tool (e.g. Support/Resistance, Fans, etc).

We will illustrate ways to quick modify multi-line drawing tools using the Fibonacci Time tool .

1 Create a Fibonacci Time drawing tool, click on it to select it and press the space bar to open the editor for the Fibonacci Time.

2 Here is the editor under the **Fib** tab.



Fibonacci Time Setup

Visual | Fib

Drawing Style
○ Expansion
● Measurement

Extension
○ No Extension
○ Current Pane
● Cross Pane

Visible	Channel(%)	Width	Color
☑	tool	2	🟦
☑	0.0	default	🟥
☑	38.2	default	🟩
☑	50.0	default	🟦
☑	61.8	default	🟦
☑	100.0	default	🟪

Channel Value
● As Percent
○ As Value

Channel Display Order
● Do not sort
○ Ascending
○ Descending

Set Defaults ...

3 Under the tab you can change the multiples, color, etc for Fibonacci Time.

Here are what the buttons below the multiples do, from left to right.

Light rainbow button lets you quickly assign light colors to the drawing object.

Dark rainbow button lets you quickly assign dark colors to the drawing object.

Vignette button lets you assign a vignette to multiple lines. The exact color of the line depends on the range of the multiple. The lowest multiple corresponds to the left most color of the vignette and the highest multiple corresponds to the right most color of the vignette.

Default button lets you assign the default color of the drawing object to the multiples.

Make Partition button lets you quickly create multiples by partitioning the drawing object.

Add a line.

Delete the selected line.

Mark all channels visible.

Mark all channels invisible.

Sorting the Multiples

You can sort the multiples so you can work on them easily. To sort, choose one of the options under **Channel Display Order**.

Percent vs. Value

By default, the line values are displayed as percentage. You can display line values as it is. The options are under **Channel Value**. For example, 38.2 percent is equal to 0.382.

Editing Line Values

You can directly edit line values in the grid. Attributes as such visibility, values, width, color can be changed.

Vertical Lines Cross Pane

You can set up vertical lines only drawing tools to draw across panes. Fibonacci time and vertical line drawing tools support this feature.

To set up a drawing tool to draw across pane:

1 Open the editor of the drawing tool

2 Check the **Cross Pane** option in the drawing tool editor

Hiding and Showing Drawing Tools

You can hide a drawing tool. Although a hidden drawing tool is not drawn, it is not removed from the chart. It is a way to temporarily turn off the display of the drawing tool.

You can hide a visible drawing tool by:

1 Select the drawing tool to be hidden.

2 Choose **Hide** from the pop up menu.

Alternatively, you can use the chart manager window to hide the drawing tool. To hide a drawing tool using the chart manager window, press the eye glasses besides the drawing tool in the chart manager window.

To show a hidden object, press the eye glasses besides the hidden drawing tool in the chart manager window.

Drawing Tool Stickyness

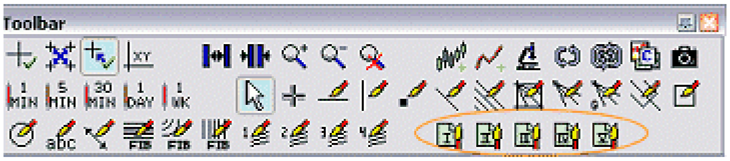
After you draw something with a drawing tool (e.g. trend line), the default behavior for the drawing tool is to switch back to a pointer cursor. You may prefer sticking to the same drawing tools so you can draw, for example, multiple trend lines easily.

To set up:

1 Choose **Manager>Tool Bar** from the main window.

2 Choose one of the option under **After drawing**.

Drawing Tool Templates



The chart tool buttons provides 5 templates for each drawing tool. The templates let you save the settings of a drawing tool. So that the next time you draw, you do not have to set the parameters of the drawing tool again.

To save a drawing tool's settings as template:

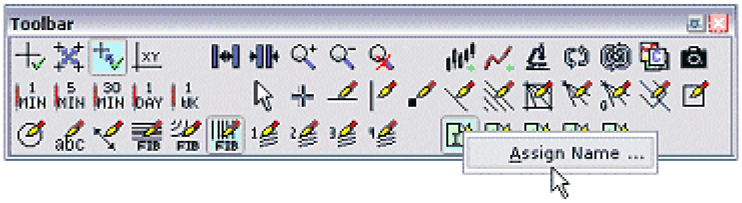
1 Select the drawing tool that you want to make template.

2 Depending on your tool bar settings, you may want to pin down the tool bar (see ***Pin Button*** (on page 1209) for more information).

3 Open the edit window for the drawing tool.

4 Press the **Set Default** button in the edit window, then choose a template.

To help you remember what the template does, you can right click on the template to assign a name to it.



The assigned name will become a tool tip for the template button, i.e., if your mouse stays on the button for a while, a tool tip containing the name will pop up.

You can also assign template names by pressing the **Set Default** button in the drawing tool's edit window.

Load Settings from Template

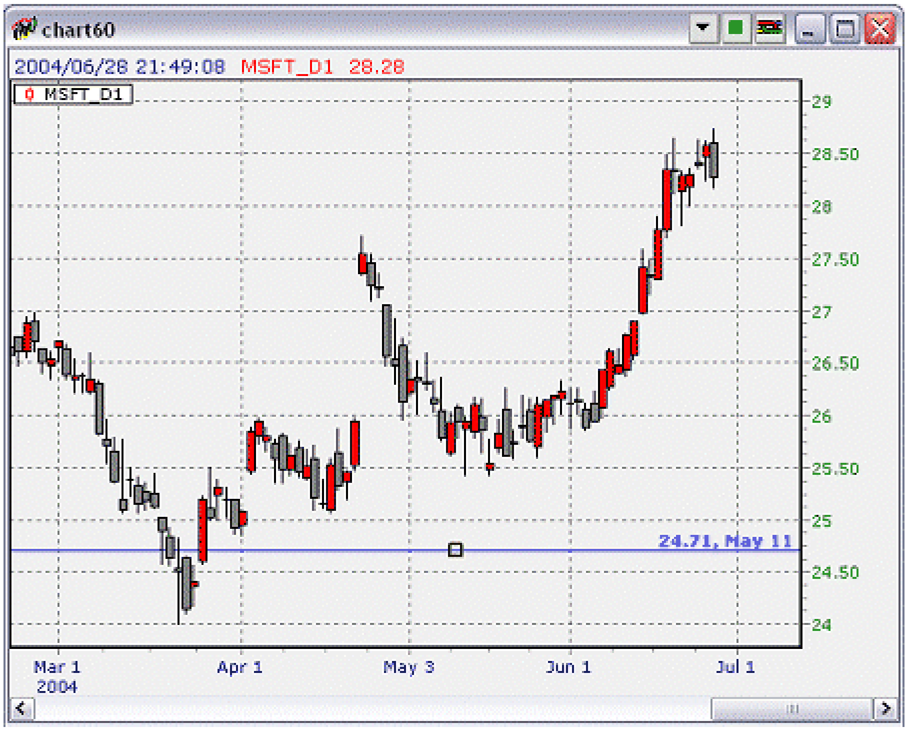
You can apply template settings to the current drawing tool.

1 Right click on the drawing tool to open pop up menu,

2 Choose **Load From Template**, then choose the template.

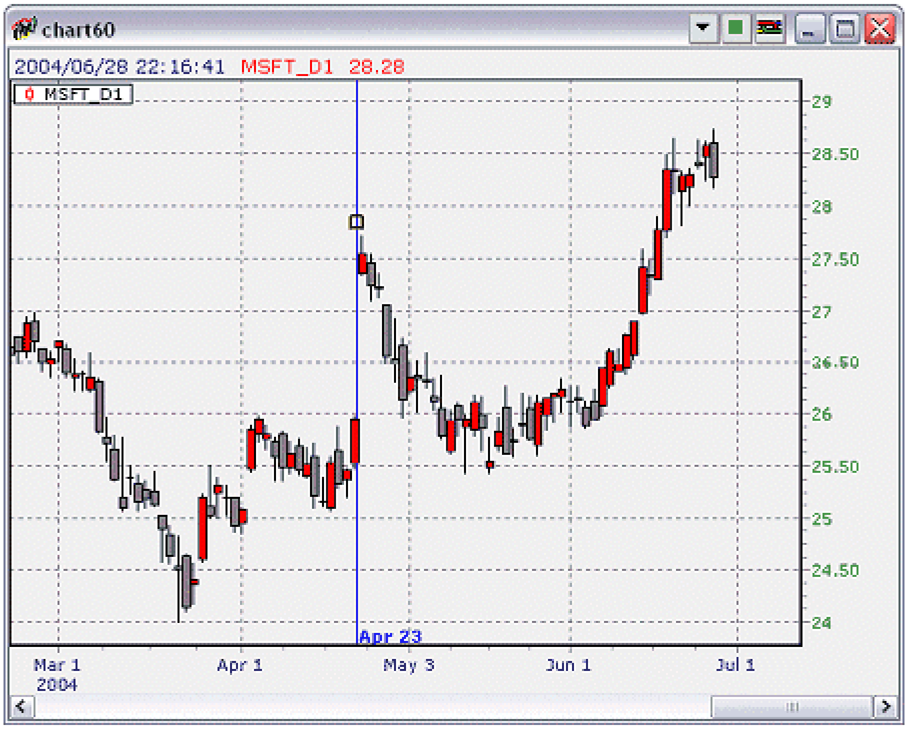
Horizontal Line Tool

The horizontal line tool lets you draw a horizontal line in the chart. To draw a horizontal line, press the horizontal line tool button on the tool bar and click on the position you want to place the horizontal line. You can drag and move the horizontal line later.



Vertical Line Tool

The vertical line tool lets you draw a vertical line in the chart. To draw a vertical line, press the vertical line tool button on the tool bar and click on the position you want to place the vertical line. You can drag and move the vertical line later.



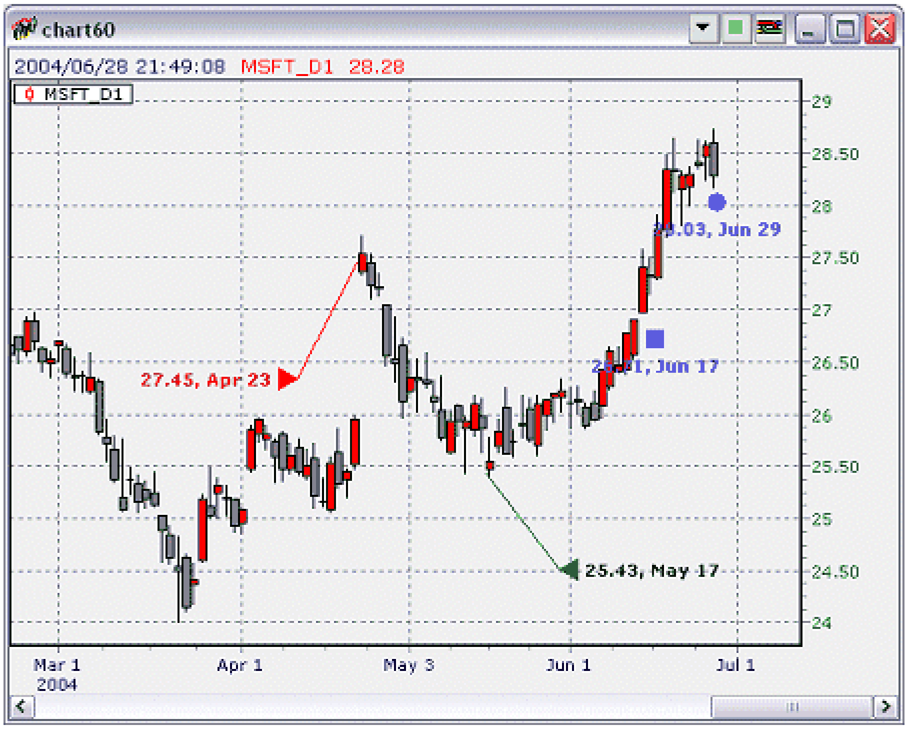
Marker Tool

The marker tool lets you mark a single in the chart. To draw a marker, press the marker tool button on the tool bar and click on the position you want to place the marker. You can drag and move the vertical line later.

Various marker styles available (square, circle, triangle, etc). Edit the marker (by double click on it) to change marker style.

If you SHIFT drag a marker, you will offset the marker. A line will be drawn between the point being marked and the marker itself. This allows you mark important points without overlapping on the point.

The figure below illustrate various marker styles.

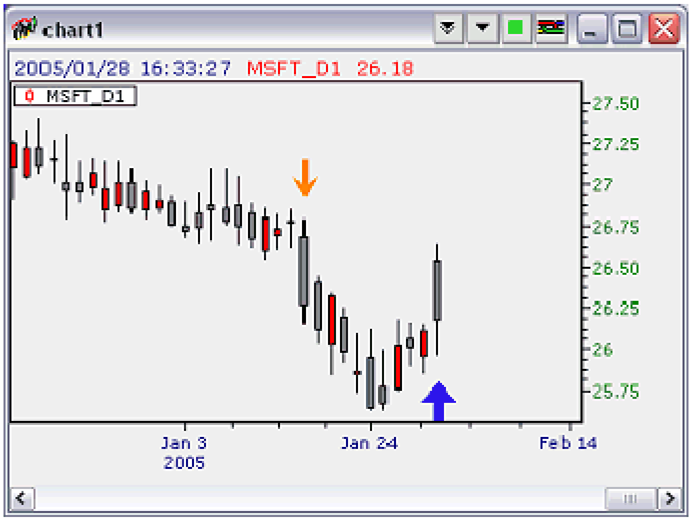


Up Arrow and Down Arrow Tools

The Up Arrow and Down Arrow tools are for drawing arrows.

NeoTicker® provides the following arrow options you can access in the set up window for arrows:

- Wide/Narrow arrows.
- Head only arrows.

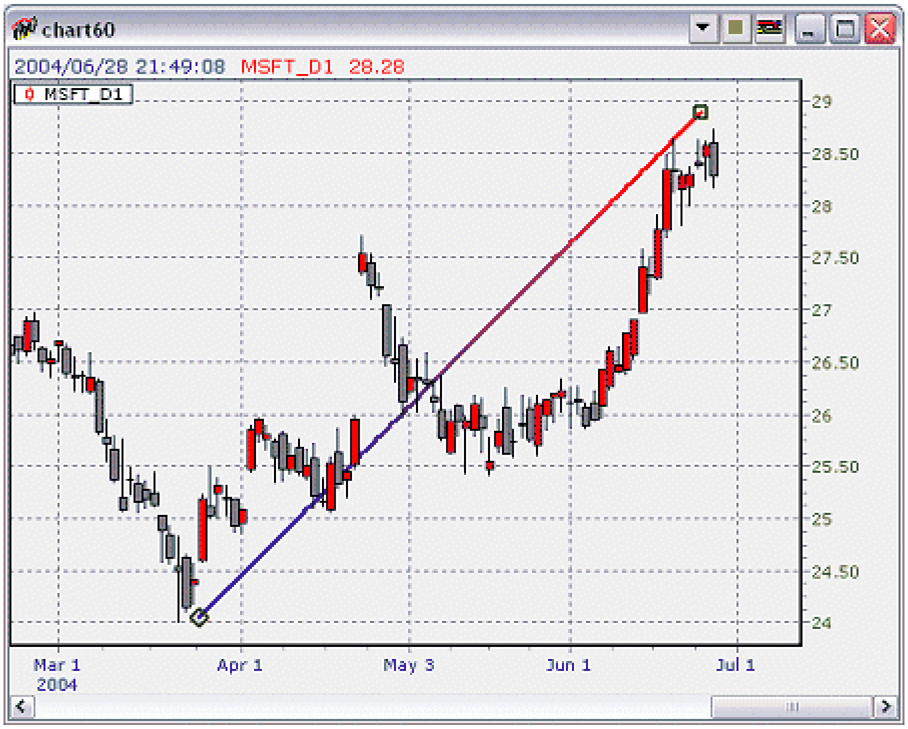


Trend Line Tool

The trend line tool lets you draw a trend line in the chart. To draw a trend line, press the trend line tool button on the tool bar. Click on the point you want the trend line to start, drag to the point you want the trend line to end and release the left mouse button.

The start of a trend line is marked by a square handle (square end). The end of a trend line is marked by a diamond handle (diamond end). The square and diamond shapes help identify the ends of the trend line when you want to extend the trend lines.

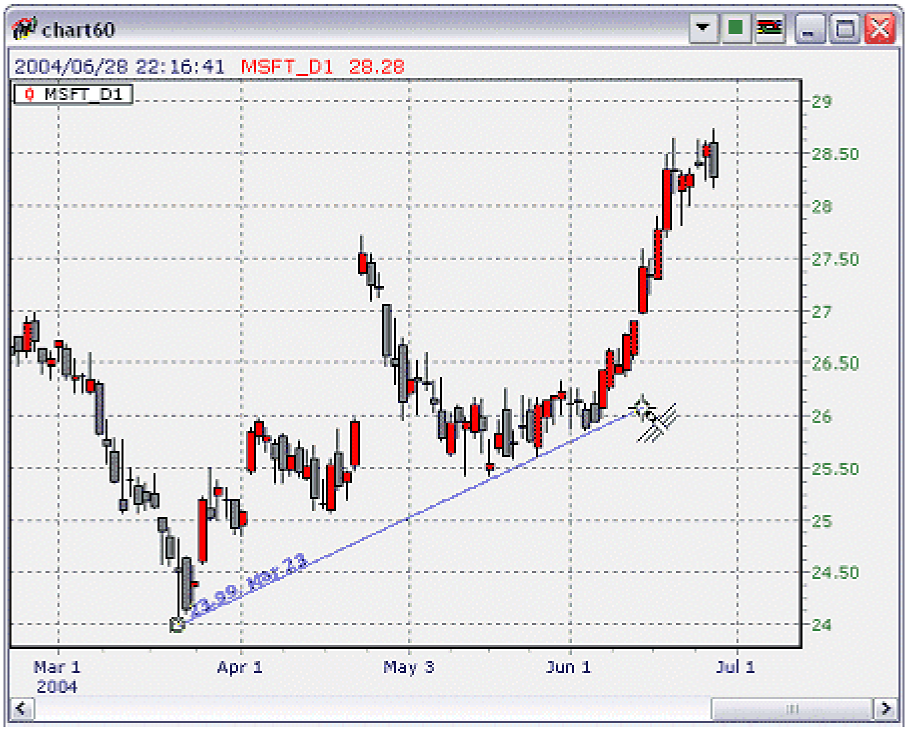
You can drag and move the trend line, and you can drag and move the square handle and the diamond handle.



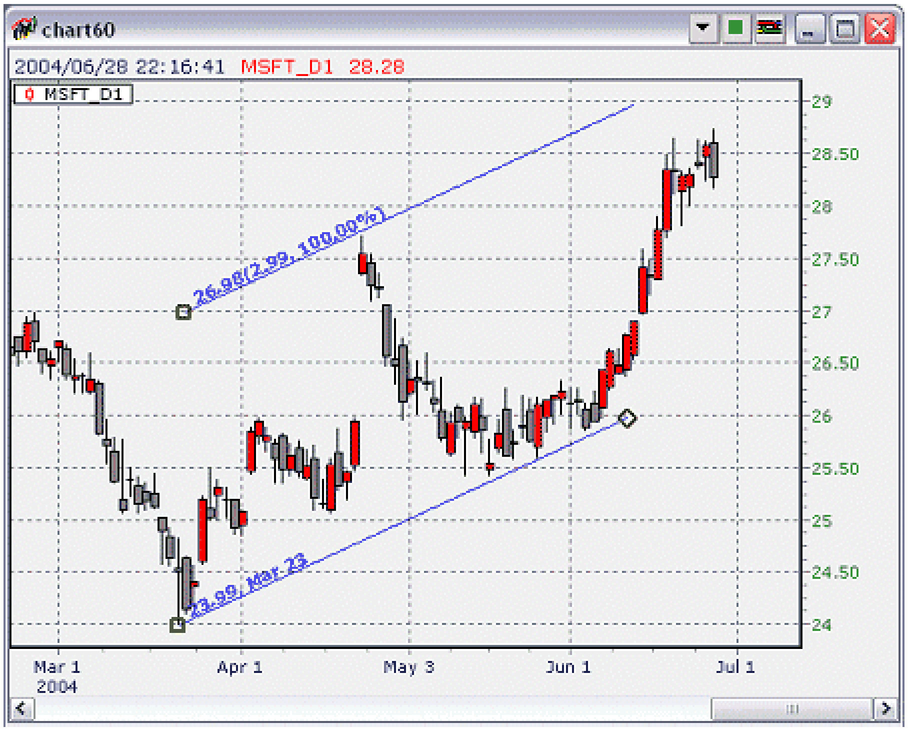
Trend Channels Tool

Trend channels are drawings of parallel lines to help you to read a chart.

To draw trend channels, press the trend channel tool button on the tool bar, left click on the point where the trend channel starts, and drag to a point where the trend channel ends. After you have released the mouse button, a trend line will be drawn. This trend line forms a base for the trend channels.



As you release the mouse button, the tool creates a line parallel to the trend line. Move the mouse to a position and press the left mouse button to anchor the parallel line.



The trend line and the parallel line are the reference lines for trend channels.

You can drag and move the trend line. You can either drag and move the square handle or the diamond handle on the trend line. To move the parallel line, you must drag the square handle on the parallel line.

Trend channels are identified by multiples. The distance between the trend line and the parallel line is considered to be 100%. If a channel has a multiple of 50%, it means that the distance of the channel from the trend line to the channel is 50% the distance of the trend line to the parallel line.

For more information about editing multiples, refer to ***Setting up Lines for Multi-line Drawing Tool*** (on page 1141).

Support/Resistance Tool

Support/resistance tool is a versatile tool that combines the best elements of traditional support/resistance, Gann Fan and time forecast tools.

To draw a support/resistance tool, press the support/resistance tool button on the tool bar window. The first time you draw a support/resistance object, you will need to specify a trend line. Then you will need to choose **Edit** from the popup menu or press the space bar to edit the properties of the support/resistance object.

Support/resistance is a complex object. The best way to draw it is to set up the tool, and save the settings to the template. The next time you draw a support/resistance object, you can reuse the settings stored in the template.

You can drag and move a support/resistance object by dragging the trend line. You can adjust the handles by dragging on them.

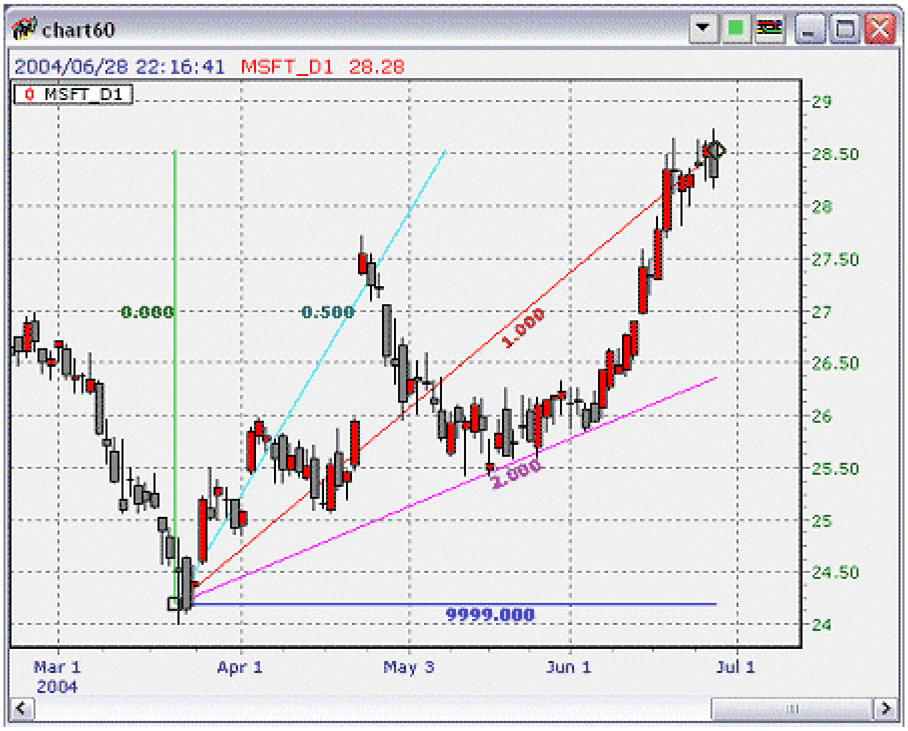
To edit support/resistance, select the support/resistance object. Choose **Edit** from the pop up menu or press the space bar. The support/resistance dialog contains five tabs, the **Visual** tab for editing visual properties; the **Price** tab for editing vertical partitions; the **Time** tab for editing horizontal partitions; the **Fan** tab for editing the fans; the **Arc** tab for editing arcs.

You can partition the support/resistance object vertically between the square handle and the diamond handle. By partitioning the object vertically, you can create horizontal lines. Each horizontal line is identified by a position. This position is relative to the square handle and the diamond handle. A horizontal line at the square handle is identified as position 0%. A horizontal line at the diamond handle is identified as position 100%.

You can partition the support/resistance object horizontally between the square handle and the diamond handle. By partitioning the object horizontally, you can create vertical lines. Each vertical line is identified by a position. This position is relative to the square handle and the diamond handle. A vertical line at the square handle is identified as position 0%. A vertical line at the diamond handle is identified as position 100%.

You can create fan lines in the support/resistance object. A fan line is identified by a multiple that determines the fan line's relationship to the trend line. For example, if a fan line has a multiple of 50%, the fan line is half as long as the trend line horizontally while the height remains the same; if a fan line has a multiple of 200%, the fan line is twice as long as the trend line horizontally while the height remains the same.

This is best illustrated with an example:



Notice that the horizontal line has a 999999% multiple. Theoretically, a horizontal fan line has an infinite multiple, but in practice, a large multiple will create a horizontal fan line.

The arcs centered around the square handle. A multiple of 1 will draw a arc at the diamond handle.

The support/resistance tool implements the trend line as a special case fan line at multiple 100%. It is suggested that you do not turn off the display of the fan line at multiple 100% because otherwise you will lose the trend line, and the handles to the support/resistance tool.

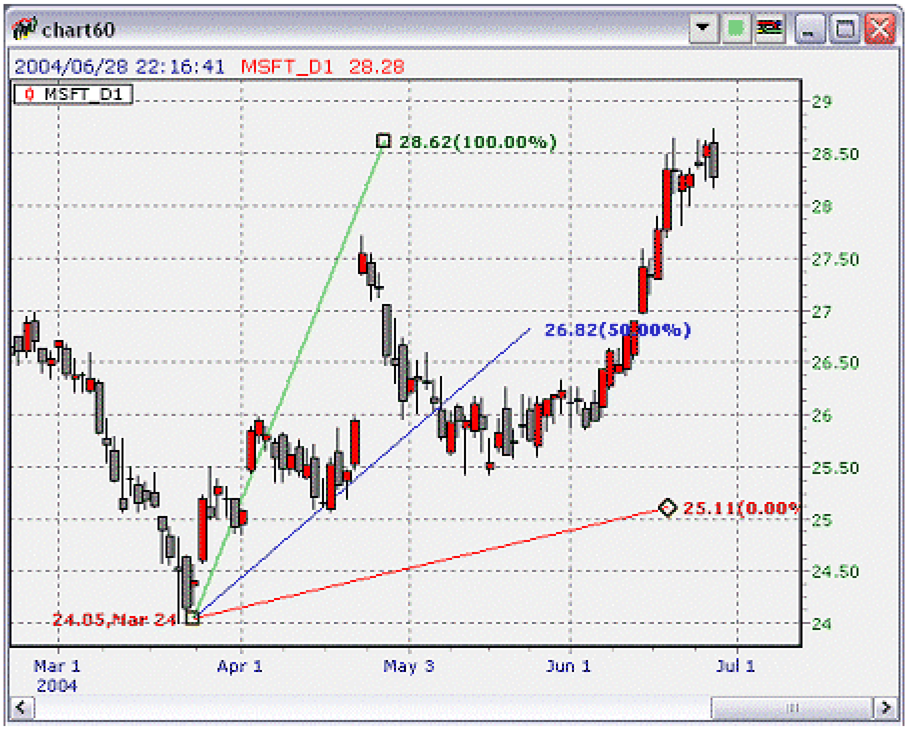
For more information about editing multiples, refer to ***Setting up Lines for Multi-line Drawing Tool*** (on page 1141).

Fan and Gann Fan Tool

To draw a fan, press the fan tool button on the tool bar window, click on a point in the chart and drag. After you release the mouse button, move the cursor to another point in the chart and click. This action specifies three points that define an angle. The fan tool considers this angle has a value 100%. All fan lines are defined with a value with respect to this angle.

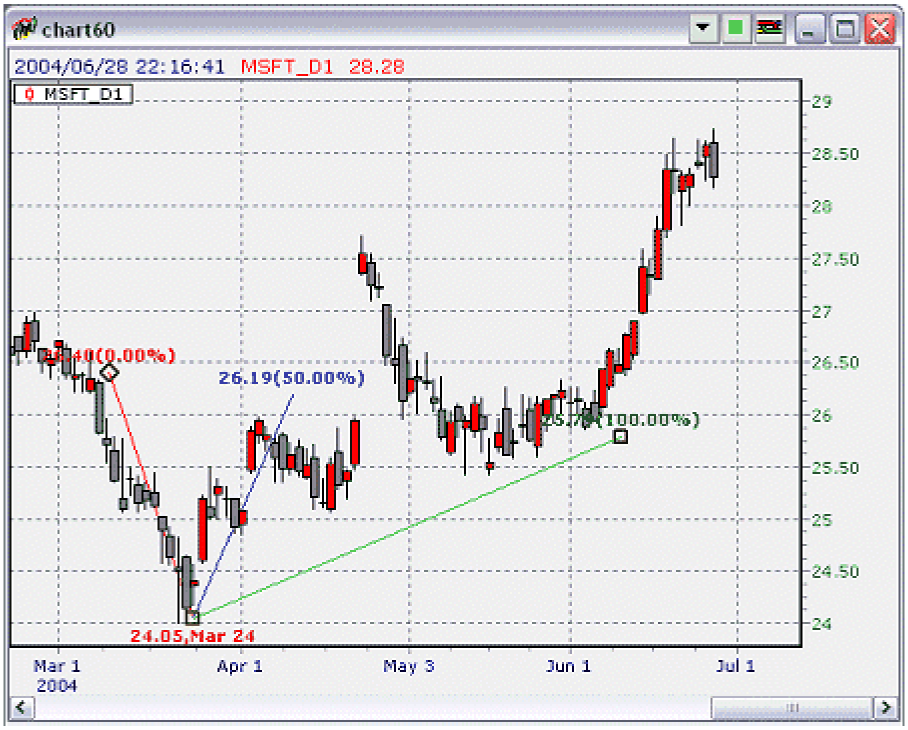
To draw a Gann fan, press the Gann fan tool button instead of the regular fan tool button.

The following is an example of fan lines, with fan lines at 0%, 50% and 100%. Refer to ***Setting up Lines for Multi-line Drawing Tool*** (on page 1141) for information on editing the fan lines.



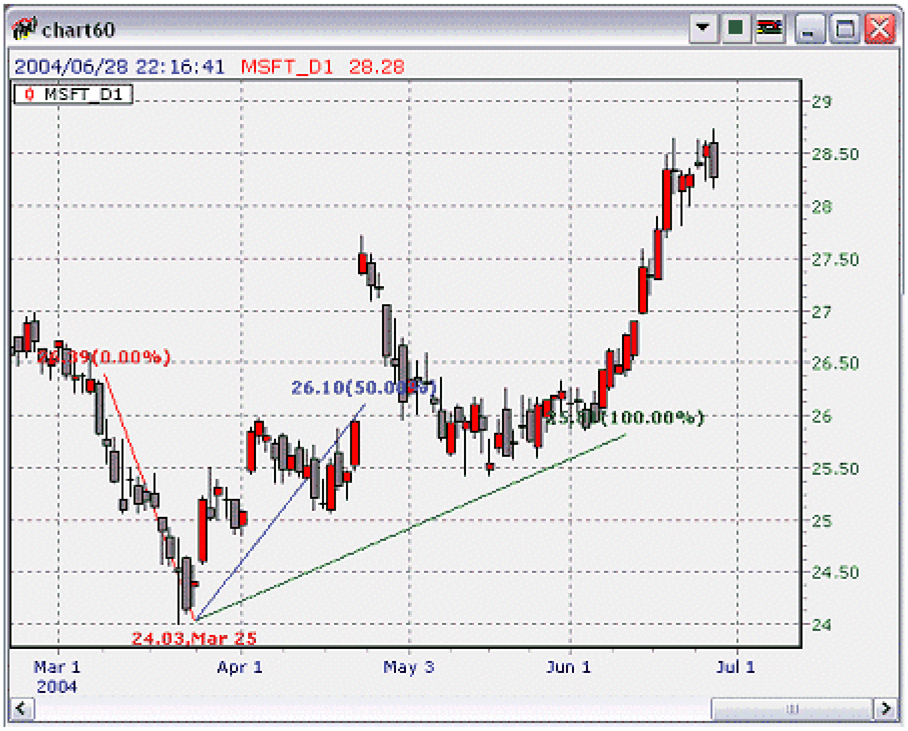
Fan vs. Gann Fan

The regular fan lines by divides the angle. For example, a fan line at 50% will be at half of the angle between the fan's two main lines. This is best illustrated in the figure below.



In the figure above, the 50% line is exactly divide the angle between the 0% and 100% line.

Some users prefer Gann fan: the 50% line should divide the distance between the end points of the 0% and 100% by exactly a half. The figure below illustrates Gann fan:



Andrew pitchfork

Andrew pitchfork is a drawing tool with three parallel trend lines base on three point you select.

To draw Andrew pitchfork press the Andrew pitchfork tool button on the tool bar . Left click on the chart you will start with a point as the end of the fork handle (Point 1), then you can drag the mouse to where you want to place the head of fork (Point 2). After you release the mouse button, a line is drawn.

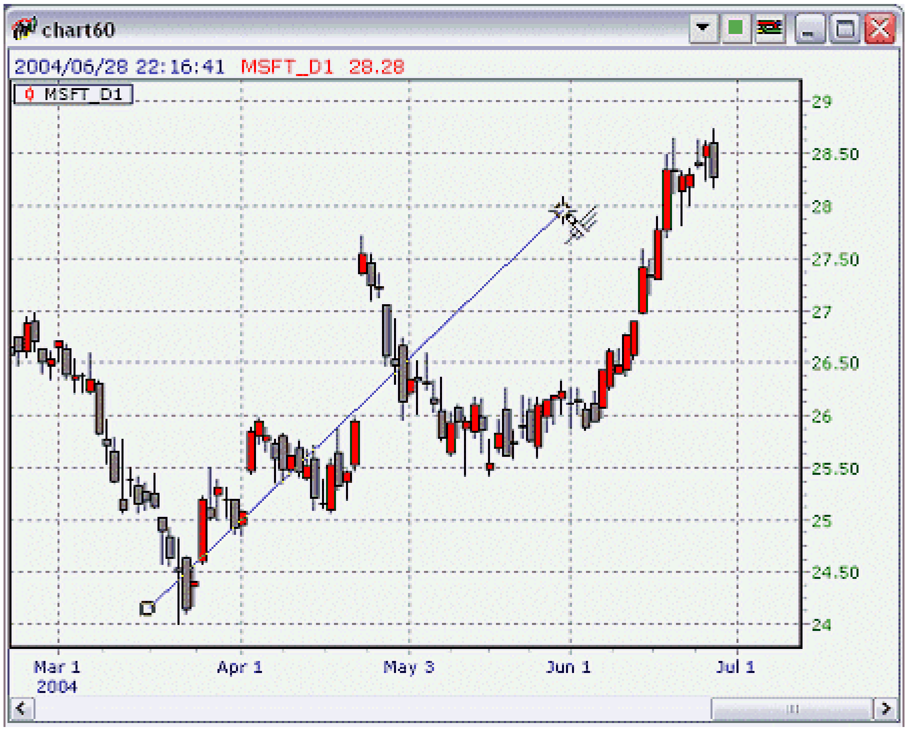
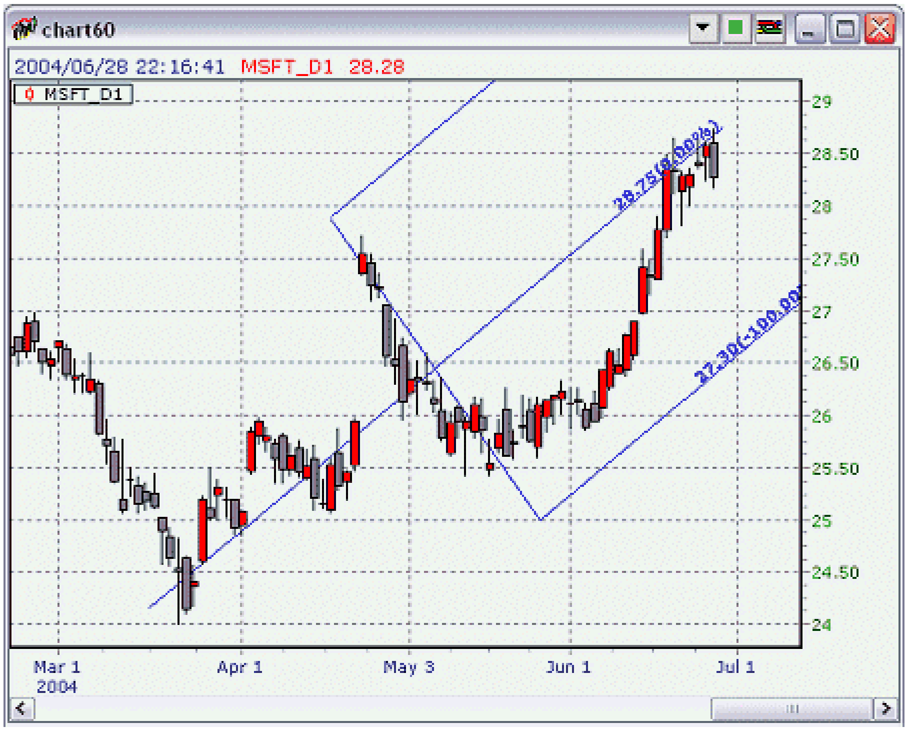


chart60

2004/06/28 22:16:41 MSFT_D1 28.28

When you move the mouse point to the side of this line, you will be able to place the third point of the fork. Click on the position where you want to place the third point.



The resulting Andrew pitchfork creates three trend lines. You can drag and move the whole fork or you can drag and move any one of the three points.

There are initially three trend lines in Andrew pitchfork the distance between the first line and second line is 100%, the second line to the third line is 100%. If you add a line has a multiple of 50% it will show up in between the top line and the middle line with the line drawing at the head of the fork.

When you want more parallel trend lines in the Andrew pitchfork on your chart simply add more multiples to Andrew pitchfork. For more information about adding multiples, refer to ***Setting up Lines for Multi-line Drawing Tool*** (on page 1141).

3-Point Projection

3-Point Projection is a collection of drawing tools that make projection based on 3 handles of the drawing tool.

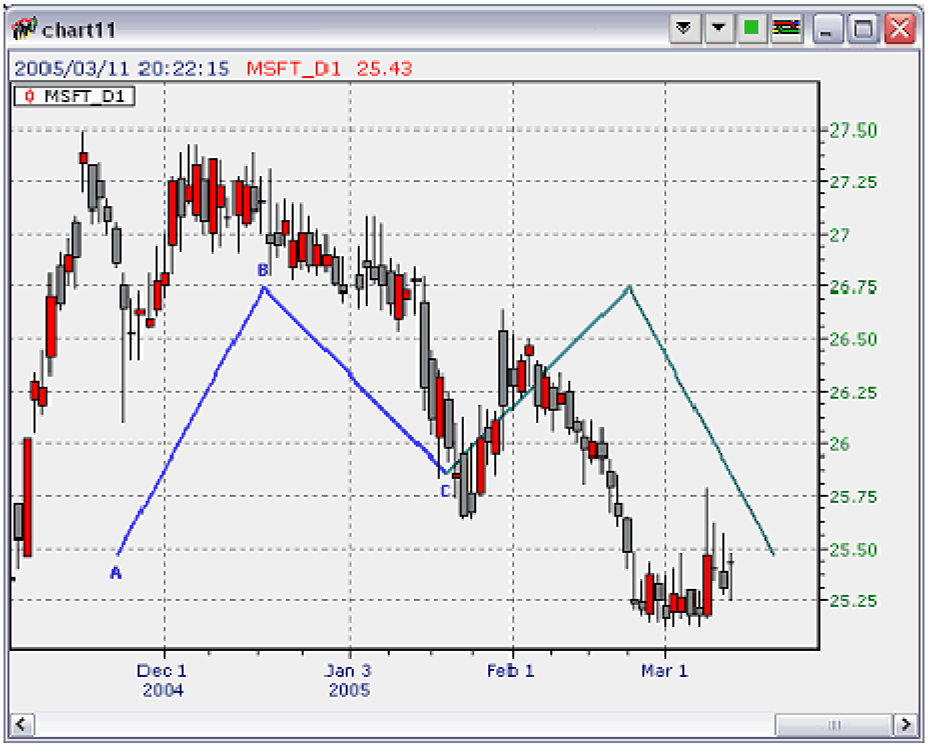
To draw 3-Point Projection, press the 3-Point Projection tool button on the tool bar.

1 Left click on the chart you will start with a point (Point A).

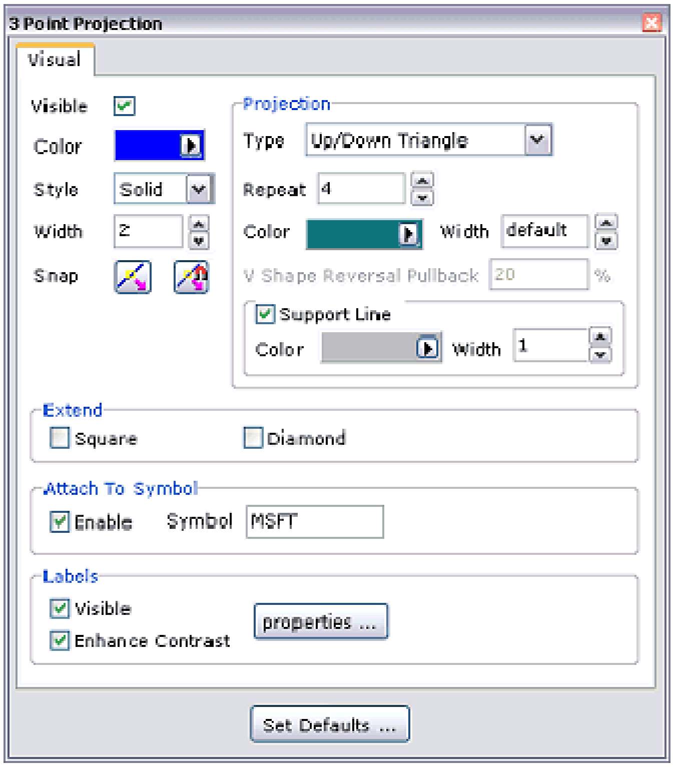
2 Do not release the mouse button. Drag the cursor and release mouse button. This is second point (Point B).

3 Move the cursor and press left mouse button. This the third point (Point C).

4 Once the three points are defined, projection will be drawn.



The default projection is mirror. You can change to other type of projection by editing the drawing tool (double click on it).



3 Point Projection

Visual

Visible ☑

Color

Style Solid

Width 2

Snap

Projection

Type Up/Down Triangle

Repeat 4

Color

Width default

V Shape Reversal Pullback 20 %

☑ Support Line

Color

Width 1

Extend

□ Square

□ Diamond

Attach To Symbol

☑ Enable

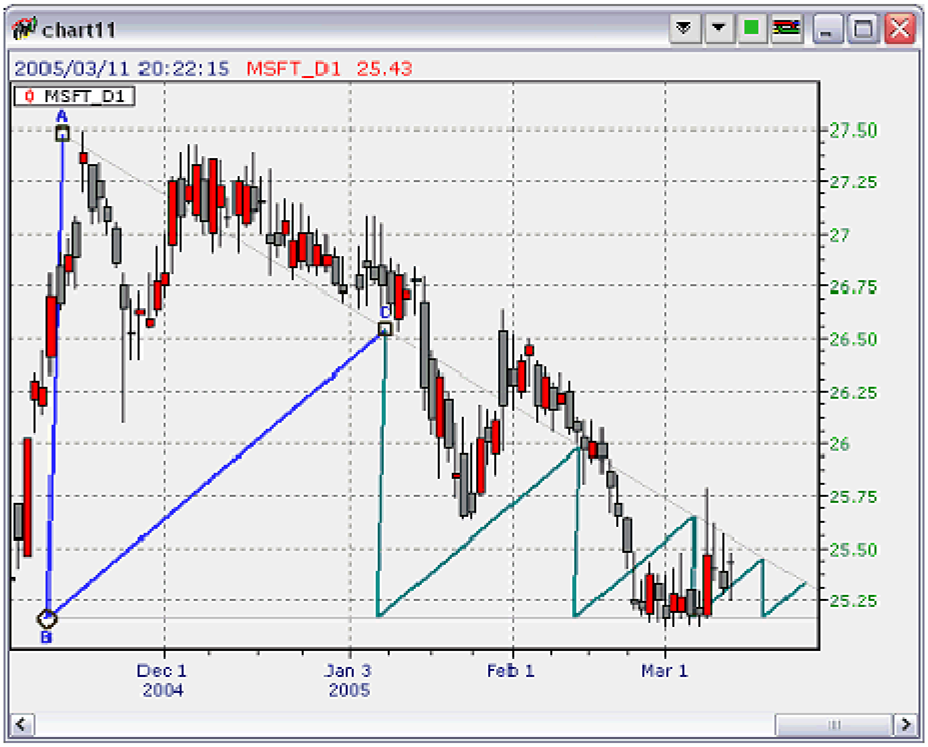
Symbol MSFT

Labels

☑ Visible

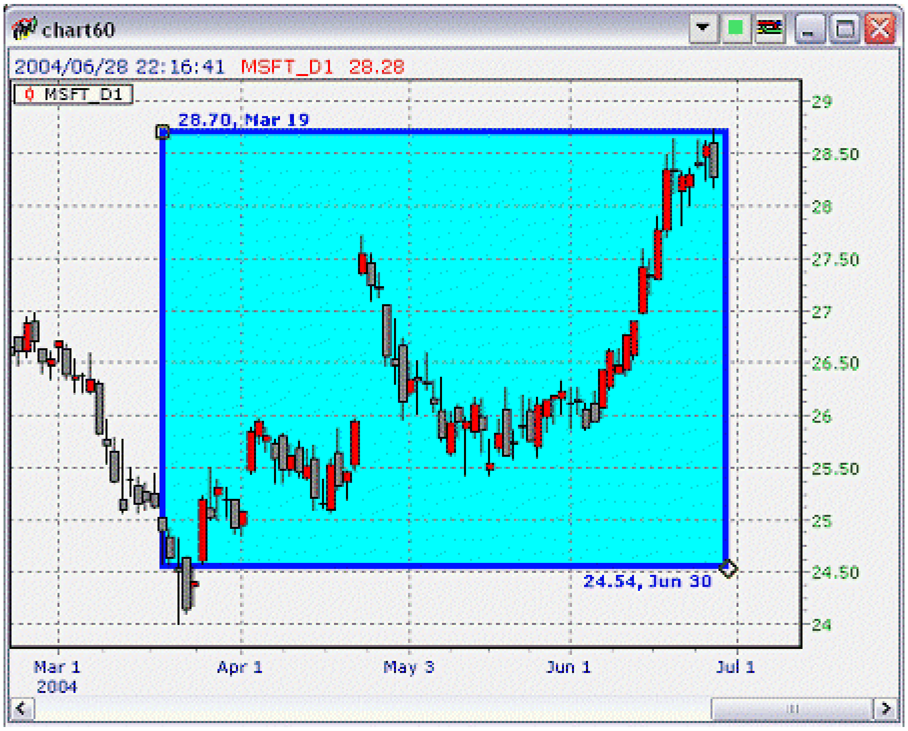
☑ Enhance Contrast

Below is an example of Up/Down Triangle projection.



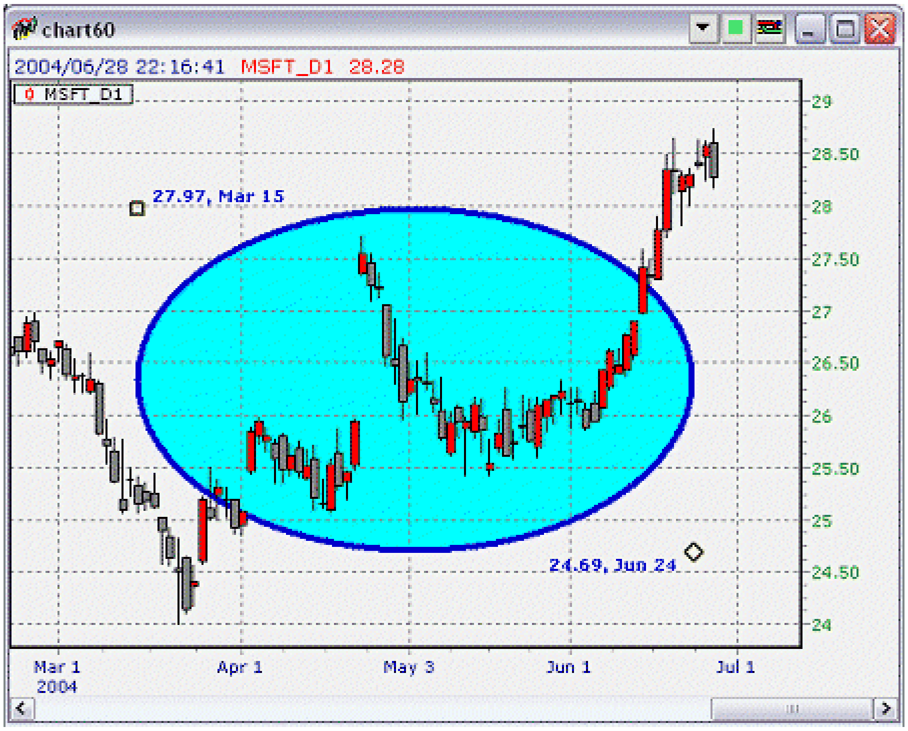
Rectangle Tool

To draw a rectangle, press the rectangle tool button on the tool bar, click on a point in the chart and drag. The point you click on and the point you release the mouse button form two opposing angles of the rectangle.



Ellipse Tool

To draw an ellipse, press the ellipse tool button on the tool bar, click on a point in the chart and drag. This window action defines the boundary box for the ellipse.



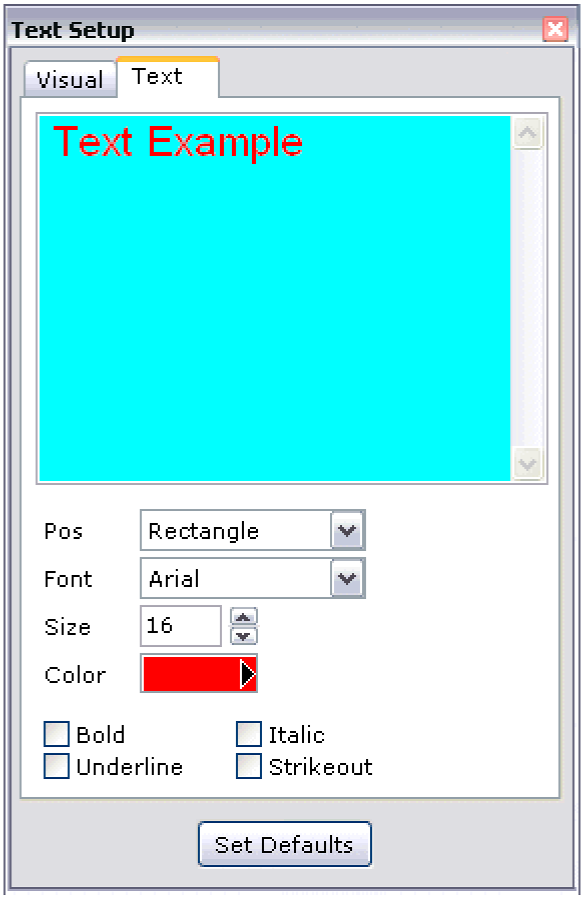
Text Tool

To put text on the chart:

1 Press the text tool button on the tool bar

2 Click on a point in the chart. This action defines the area where the text will be put in.

3 The Text Setup window will open, you can start entering text. Close the window once you are done.



Text Example

Pos Rectangle

Font Arial

Size 16

Color 🟥

□ Bold □ Italic

□ Underline □ Strikeout

To edit existing text in a chart:

1 Double click the text box to open Text Setup window.

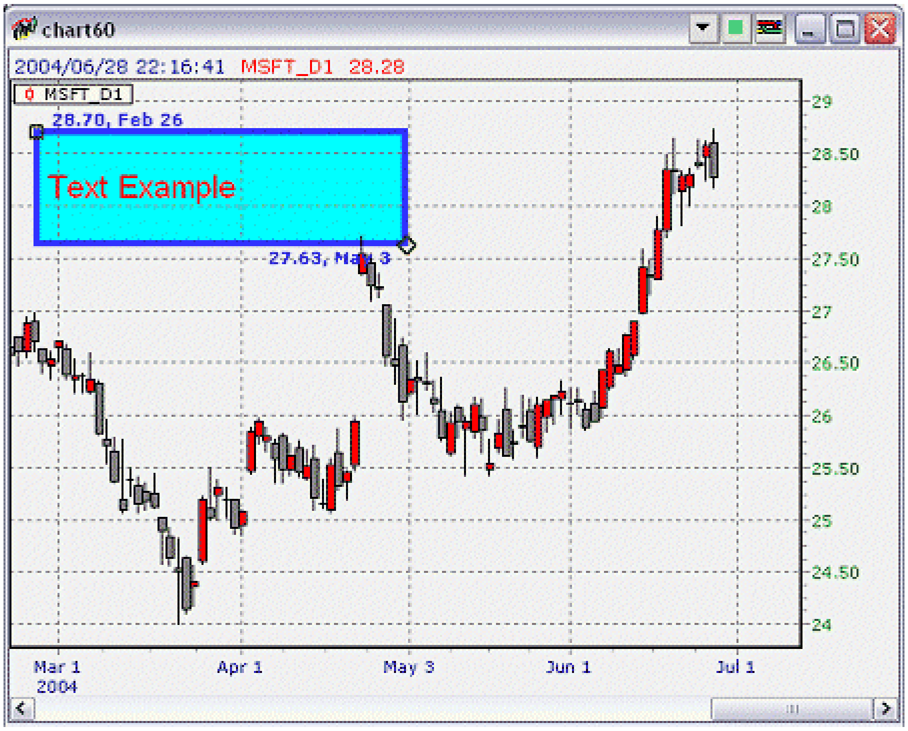
2 Press **Text** tab.

3 Edit the text.

4 Close Text Setup window.

You can change the following properties under the **Text** tab:

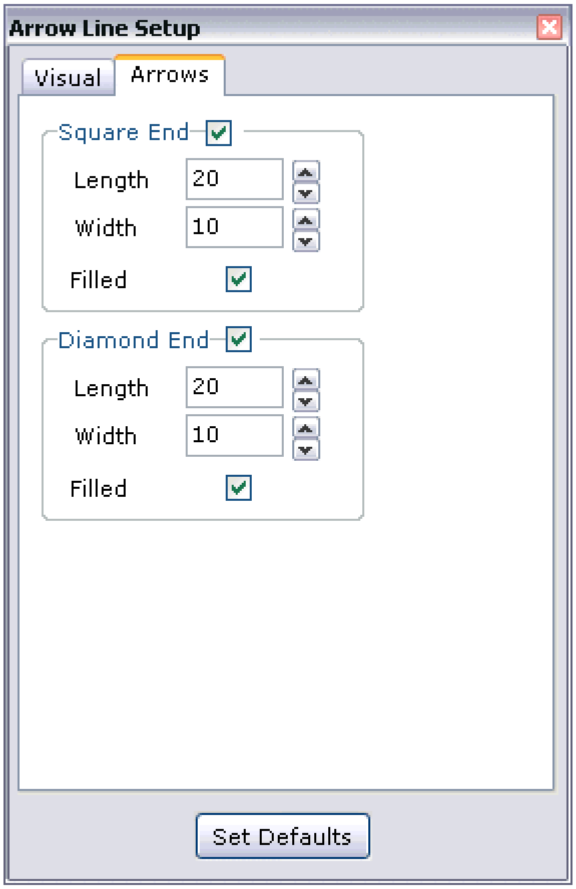
Setting	Function
Font	The text font
Size	Font Size
Color	Font Color
Bold	Bold text
Italic	Italic text
Underline	Underline text
Strikeout	Strike out text



Arrow Line Tool

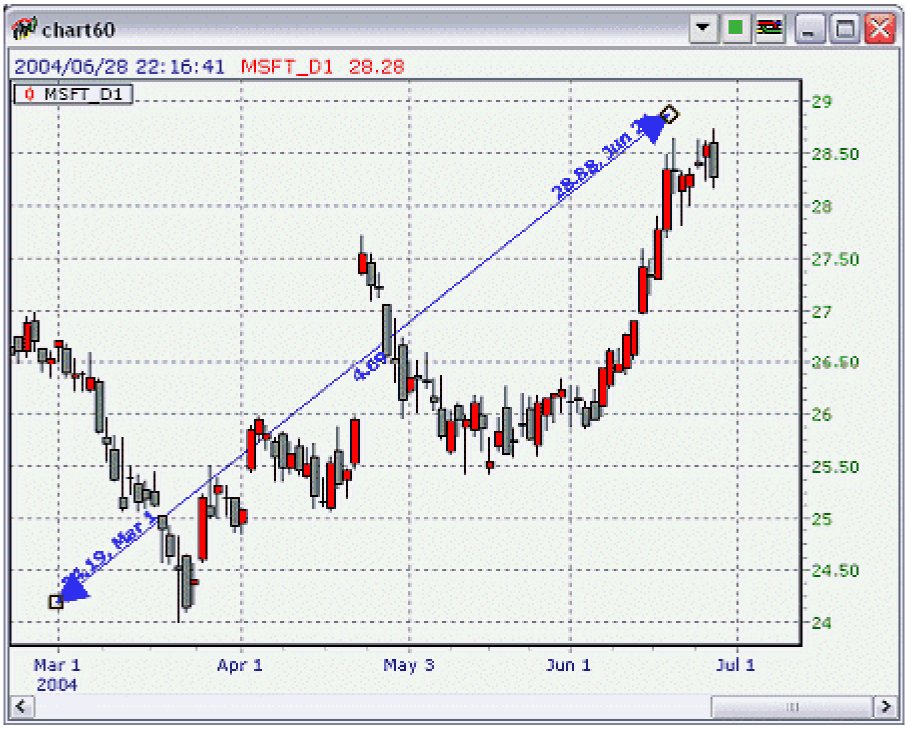
The arrow line tool lets you draw a trend line with arrow on one or both end in the chart.

To draw an arrow line, press the arrow line tool button . Click on the point you want the line to start, drag to the position you want the arrow line to end and release the left mouse button. Initially there are no arrows drawn on the line. You can hit space bar or right-click and select **Setup**. Click on the **Arrows** tab and enable the arrowhead of you choice.



Square End ☑
Length 20
Width 10
Filled ☑

Diamond End ☑
Length 20
Width 10
Filled ☑



The following table explains the settings under the **Arrows** tab.

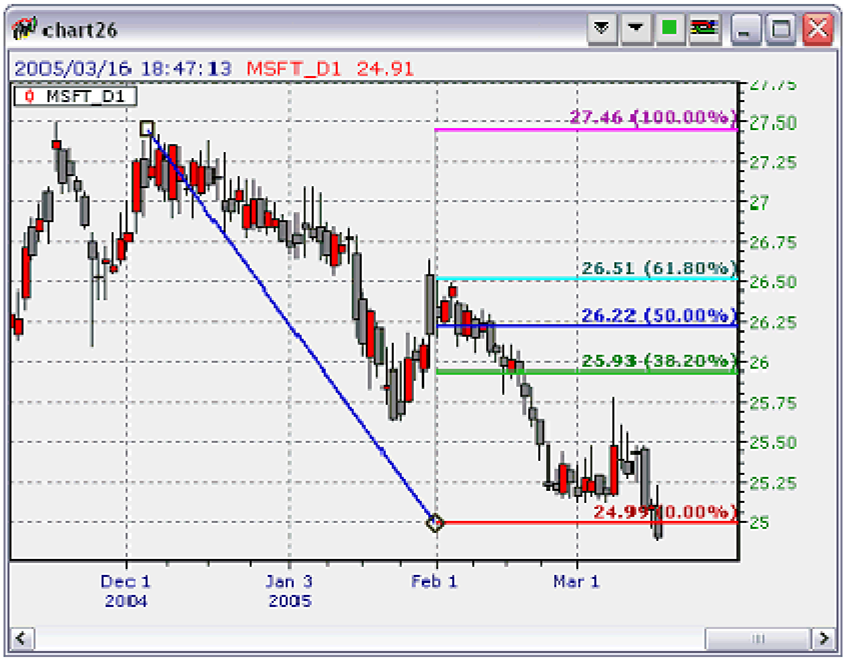
Setting	Meaning
Square End	Add arrowhead to the square end when checked
Length	Length of the square end arrowhead
Width	Width of the square end arrowhead
Filled	Whether the square end arrowhead is solidly filled or hollow
Diamond End	Add arrowhead to the diamond end when checked
Length	Length of the diamond end arrowhead
Width	Width of the diamond end arrowhead
Filled	Whether the diamond end arrowhead is solidly filled or hollow

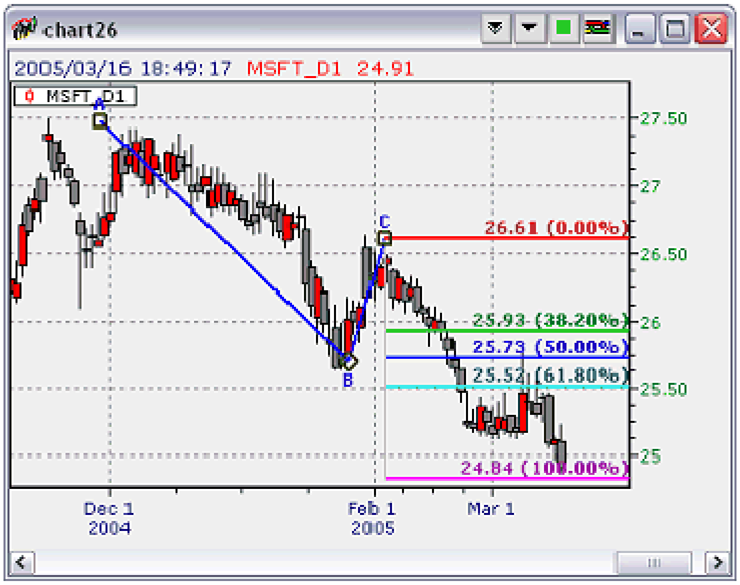
Fibonacci Level Tool

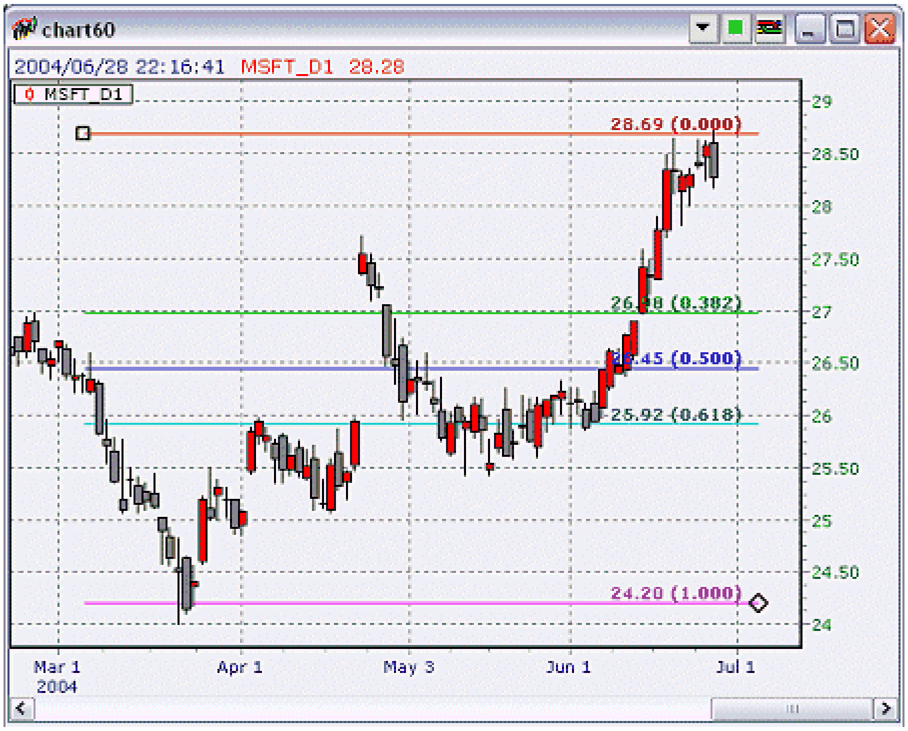
Fibonacci Level is a multiple color drawing tool with level lines based on fibonacci ratio.

To draw Fibonacci level press the Fibonacci Level Tool button or the 3-Point Fibonacci Level Tool button on the tool bar. Left click on the chart, you will have the Fibonacci level lines appear on the chart. You can left click on the body of any lines and drag to move the lines to a different location. You can click and drag the square box on the upper left hand side or the diamond box on the lower right hand side to adjust the levels.

Fibonacci tools support different drawing styles (For 2-point Fibonacci - Measurement, Retracement, Expansion. For 3-point Fibonacci - Measurement, Expansion). The different drawing styles determine the starting point and direction of the Fibonacci lines. You can use the drawing tool editor to select the drawing style.







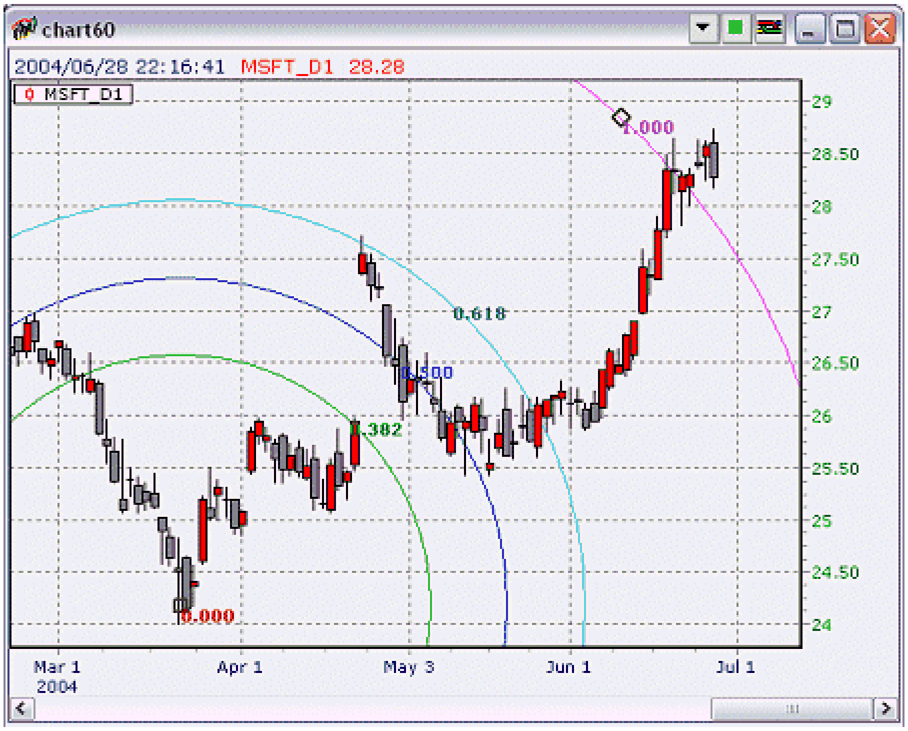
Refer to ***Setting up Lines for Multi-line Drawing Tool*** (on page 1141) for information on editing Fibonacci lines.

The make partition button ☰ in the edit window will create extra fibonacci lines.

Fibonacci Arc Tool

Fibonacci Arc Tool is a multiple color drawing tool with circles based on Fibonacci ratio.

To draw Fibonacci arc, press the Fibonacci arc tool button on the tool bar. Left click on the chart, you will have the arcs with Fibonacci ratio appear on the chart. You can left click on the body of the circle and drag to move the arcs around. You can click and drag the square and diamond end to adjust the distance between the arcs.



Refer to ***Setting up Lines for Multi-line Drawing Tool*** (on page 1141) for information on editing Fibonacci arcs.

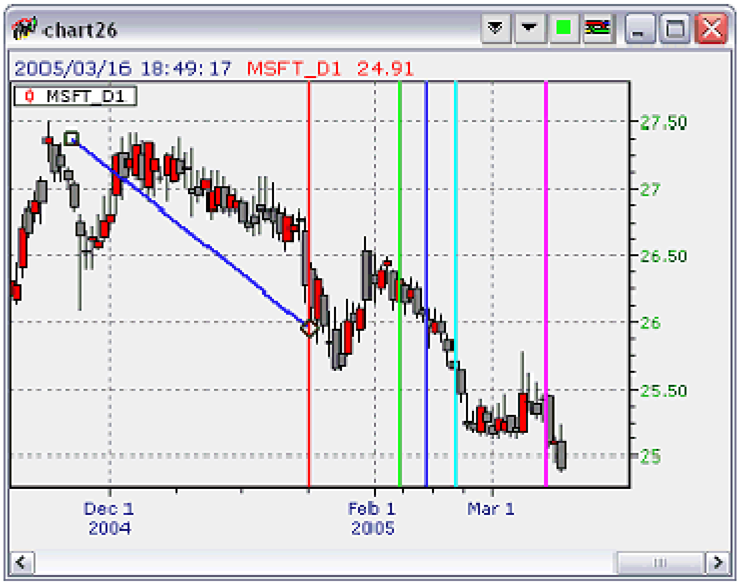
The make partition button in the edit window will create extra fibonacci arcs.

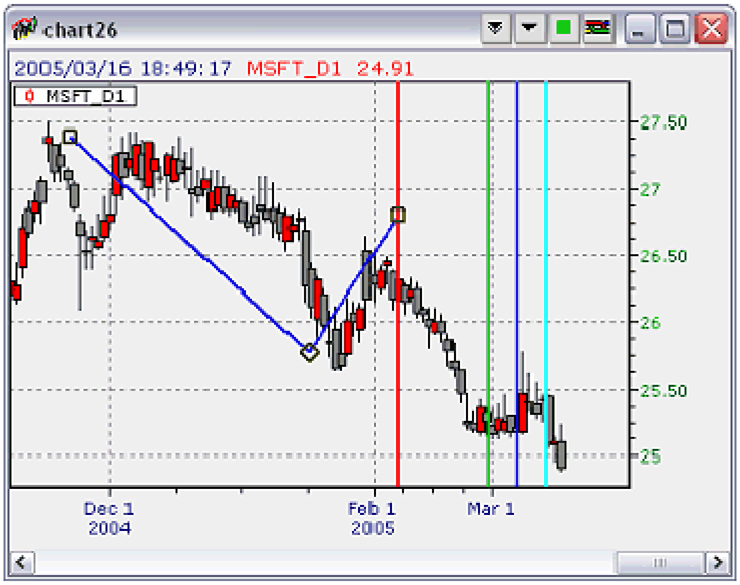
Fibonacci Time Tool

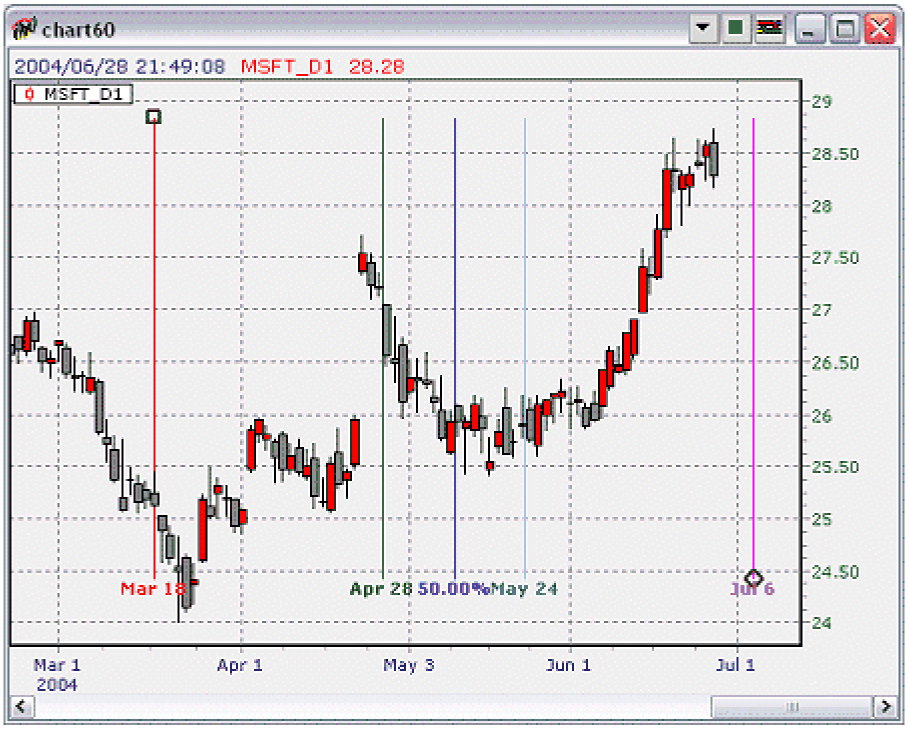
Fibonacci Time Tool is a multiple color drawing tool with lines perpendicular to the time axis. These time lines are drawn based on the Fibonacci ratio.

To draw Fibonacci time lines, press the Fibonacci Time Tool button or the 3-Point Fibonacci Time Tool button on the tool bar. Left click on the chart to place the fibonacci time tool, you will have the lines with Fibonacci ratio. You can left click on the body of the lines and drag to move the lines around. You can click and drag the square and diamond to adjust the distance between the lines.

Fibonacci tools support different drawing styles (Measurement and Expansion). The different drawing styles determine the starting point and direction of the Fibonacci lines. You can use the drawing tool editor to select the drawing style.







Refer to ***Setting up Lines for Multi-line Drawing Tool*** (on page 1141) for information on editing Fibonacci arcs.

The make partition button ☰ in the edit window will create extra fibonacci arcs.

Highlight Handles

When you select a drawing tool, you can choose to highlight the whole drawing tool, or just the handles only.

1 Choose **Program>User Preference** from main window.

2 Press the **General** tab.

3 The option is **When active, highlight handles only**.

Checking off this option will make NeoTicker® behaves like version 2.8 and before.

Drawing Tool Above or Below Data Series and Indicators

You can control whether drawing tools should be drawn above or below data series and indicators.

1 Choose **Program>User Preference** from main window.

2 Press the **General** tab.

3 Choose between **Below data and indicators** and **Above data and indicators**.

Auto Enhancing Highlight Contrast

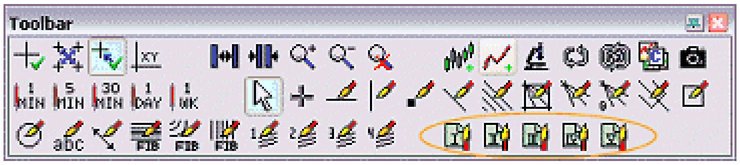
Time chart automatically enhance draw tool highlight color. This will make the highlight color easy to read if the chart background color is similar to the highlight color. You can toggle this option on/off:

1 Choose **Program>User Preference** from main window.

2 Press the **General** tab.

3 The option is **Auto Enhance Highlight Contrast**.

Custom Drawing Tools



You can associate these buttons with user defined custom drawing tools. To find out more, go to ***Installing Custom Drawing Object Script*** (see "Installing Custom Drawing Tool Scripts" on page 1033).

Point and Figure Chart

Point and Figure chart is a specializing charting style of data series. It requires two extra parameters to define the way it works. First parameter is the box size and the second parameter is the number of boxes to reverse direction.

To turn a regular data series into point and figure chart:

1 Select the data series and press space bar to edit it.

2 In the editor, under **Visual**, change the **Style** to **P+F**.

3 Enter the parameters under **Point and Figure**. The first parameter is box size. Use a small size (e.g. 0.05) of intraday charts and use a large size (e.g. 1) for EOD charts. The second parameter is number of boxes to reverse direction.

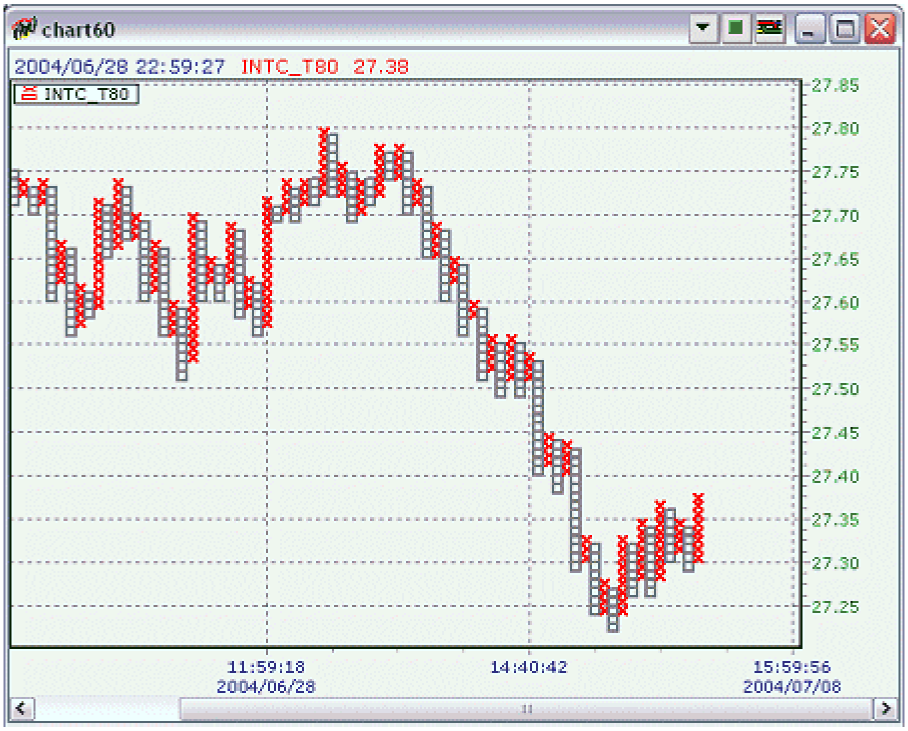
4 Press the **Apply** button.

5 (Optional) For best visual effect, use a bar width that is at least 6. This allows the crosses and squares to be clearly marked.

6 (Optional) Set **Bar Driven** under the **Data Settings** tab in Chart Manager to make the point and figure bars even space.

7 (Optional) For day trading focusing on very short time frame, you can enable **Break on Day Change**. When this option is enabled, Point and Figure chart will start as new on each trading day.

Here is an example of point and figure chart.



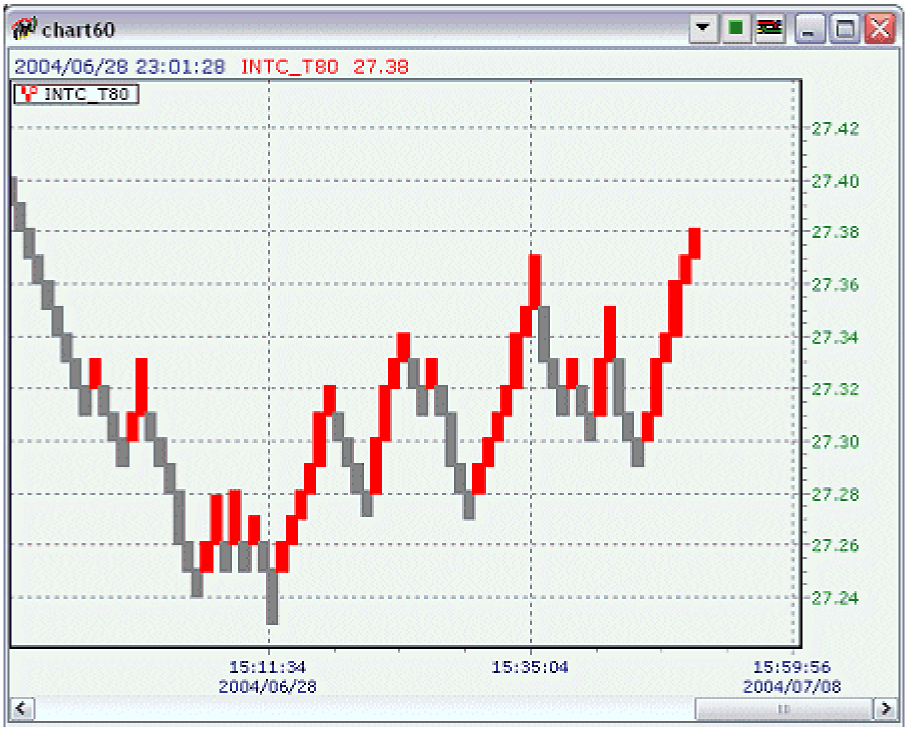
Point and Figure chart is a data series. You can apply indicators on it.

Three Line Break Chart

Three Line Break chart is a specializing charting style of data series. It requires an extra parameter to define the line. To turn a regular data series into Three Line Break chart:

1. Select the data series and press space bar to edit it.
2. In the editor, change the **Visual Style** to **3LB**.
3. Enter the parameters for **Lines**. Press the **Apply** button.
4. (Optional) Set **Bar Driven** under the **Data Settings** tab in chart manager to make the Three Line Break chart even space.
5. (Optional) For day trading focusing on very short time frame, you can enable **Break on Day Change**. When this option is enabled, Three Line Break chart will start as new on each trading day.

Here is an example of Three Line Break chart.



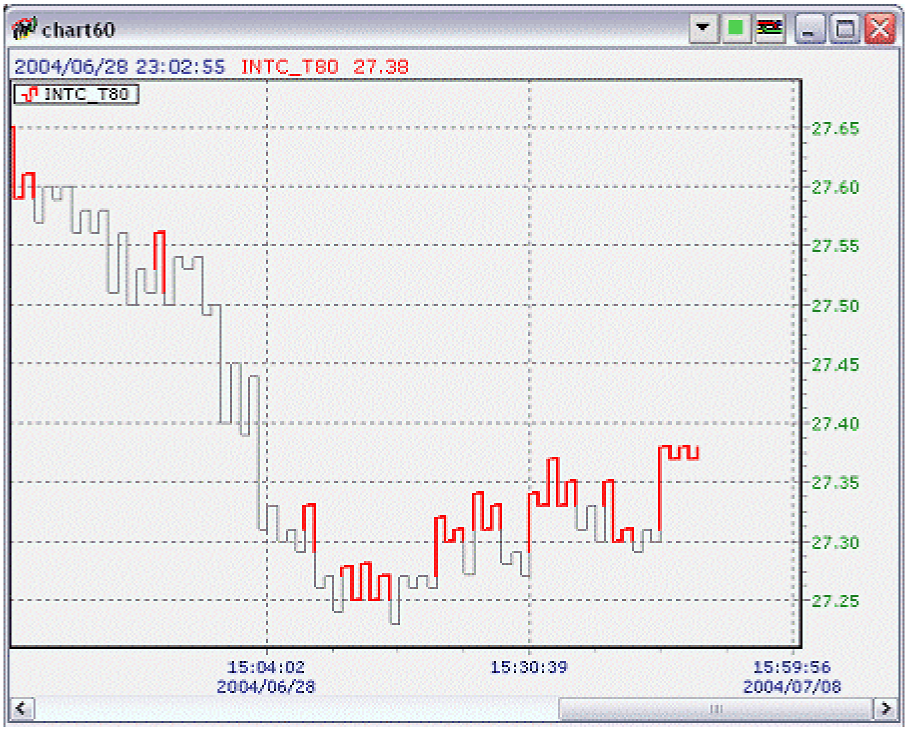
Three Line Break chart is a data series. You can apply indicators on it.

Kagi Chart

Kagi chart is a specializing charting style of data series. It requires an extra parameter to define the turn around. To turn a regular data series into Kagi chart:

1. Select the data series and press space bar to edit it.
2. In the editor, change the **Visual Style** to **Kagi**.
3. Enter the parameters for **Turn Around**. Press the **Apply** button.
4. (Optional) Set **Bar Driven** under the **Data Settings** tab in chart manager to make the Kagi chart even space.
5. (Optional) For day trading focusing on very short time frame, you can enable **Break on Day Change**. When this option is enabled, Kagi chart will start as new on each trading day.

Here is an example of Kagi chart.



Kagi chart is a data series. You can apply indicators on it.

Superposition Chart

Superposition is a new technology introduced by TickQuest. Superposition lets you summaries multiple bars into a single bar according to different criteria. What is unique is the transformation is data-based. You have the flexibility of visual displaying superposition data in different visual styles.

To enable Superposition:

1 Add a data series to chart.

2 (Optional) In chart manager, under **Data Settings** tab, in **Time Axis Style** box, change to **Bar Driven**. Press **Apply** button.

3 Select the data series, press space bar to open Edit Data window.

4 Make sure the **Visual style** is not **P+F**, **Kagi**, or **3LB**. These styles are already summarizing bars and you cannot apply superposition on them.

5 Check **Superposition** box.

6 Choose either **Momentum Bars**, **Price Range**, **Vol Range**, **% Range**, **Tick Range**, **Data Sync**, **3 Line Break**, **Kagi**, or **Point-n-Figure.**

7 Choose a visual style for display.

8 (Optional) For day trading focusing on very short time frame, you can enable **Break on Day Change**. If this option is enabled, Superposition start as new on each trading day.

9 Press **Apply** button.

Below are the explanation of the different Superposition options.

Momentum Bars

Data is accumulated in the current bar. Once the high-low range is larger than the specified **Min Range**, a new bar is created. With Momentum Bars, price movement of the old bar that is outside of **Min Range** will be transferred to the new bar. So in effect, all bars will have the same high/low range.

You need to specify the tick size of the underlying instrument. For stocks, use 0.01 (one cent movement). For contracts, the tick size is instrument specific, e.g. S&P 500 e-mini use a tick size of 0.25.

Momentum Bars is invented by Danton Long. Below is a list of published articles about Momentum Bars:

- MacRae, Desmond, "Paradigm Shift Lights the Way to Momentum Bars", SFO Magazine, Feb 2003.
- MacRae, Desmond, "Momentum Bars: The Sequel - The New World of Technical Analysis and Parallel Invention", SFO Magazine, Apr 2003.

Price Range

Data is accumulated in the current bar. Once the high-low range is larger than the specified **Min Range**, a new bar is created.

Vol Range

Data is accumulated in the current bar. Once the accumulated volume is larger than or equal to the specified **Min Volume**, a new bar is created.

% Range

Data is accumulated in the current bar. Once the high-low range is larger than the specified **Min Percent**, a new bar is created.

Tick Range

Data is accumulated in the current bar. Once the total number of ticks is larger than or equal to the specified **Min Tick**, a new bar is created.

Note: this option works properly only if the data contains tick volume. Tick volume is not supplied by all data vendors. To verify your data has tick volume, apply a tick indicator to the data. If your data has tick volume, you will see it, otherwise you will see a flat graph of 1 or 0.

Data Sync

This option is for charts with multiple data series. When you specified **Data Sync** for a data series, bars are generated in sync with the bars in the first data series in the chart, i.e. for each bar in the first data series, a corresponding bar in the **Data Sync** series is created.

3 Line Break

Data is accumulated in the current bar. Once line break criteria is true as specified by **Lines**, a new bar is created. Set Lines to 3 for classic behavior.

Kagi

Data is accumulated in the current bar. Once turnaround criteria is true as specified by **Turn Around**, a new bar is created.

Point+Figure

Data is accumulated in the current bar. New bar is created using P+F criteria as specified by **Point and Figure**.

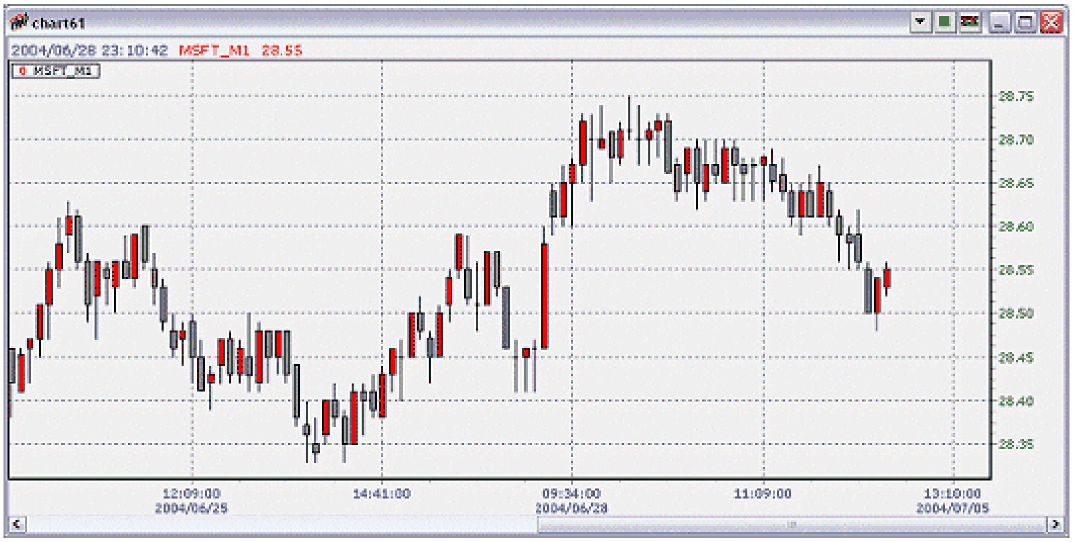
The following visual style makes Superposition behaves like class charts - **Flex Blocks**, **Flex Kagi**, **Flex P+F**.

Example - Price Range Candle

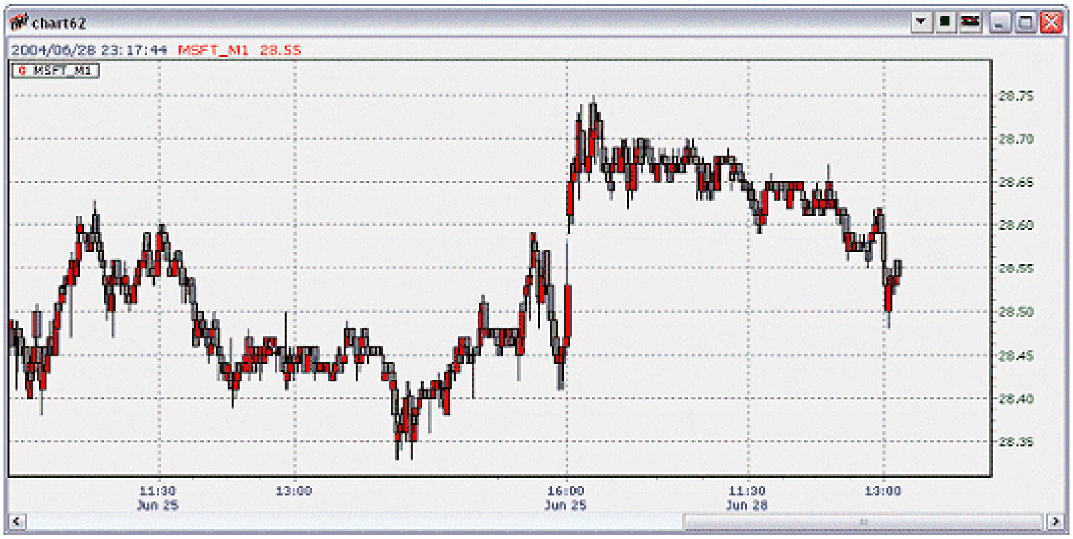
This example is MSFT Price Range Candle based on 1-min MSFT data. The settings are:

- Visual Style - Candle
- Superposition on, set to Price Range
- Min Range - 0.05

Therefore, each candle consist of $0.05 price movement. You can see in the chart what is the open, high, low, close within these $0.05 movements.



Compared to a regular 1-minute chart below that covers the same time period.



Price Range Candle has much less noise. In fact, the noise level is similar to a regular 5-minute chart. On the other hand, in fast market, Price Range Candle has the same fast response as a 1-minute chart. So Price Range Candle has the best features of the faster and slower charts.

How to use Price Range Candle:

- Price Range Candle filter out data when there is insufficient price movement. Because candles are not formed until there is sufficient price movement, Price Range Candle is particular suitable to help detecting breakouts.
- In time driven mode, spacing between bars provides information about how stable the price is. In the chart above, you can see that narrow spacing indicates the price is not very stable. In this case, breakout/breakdown often follows.

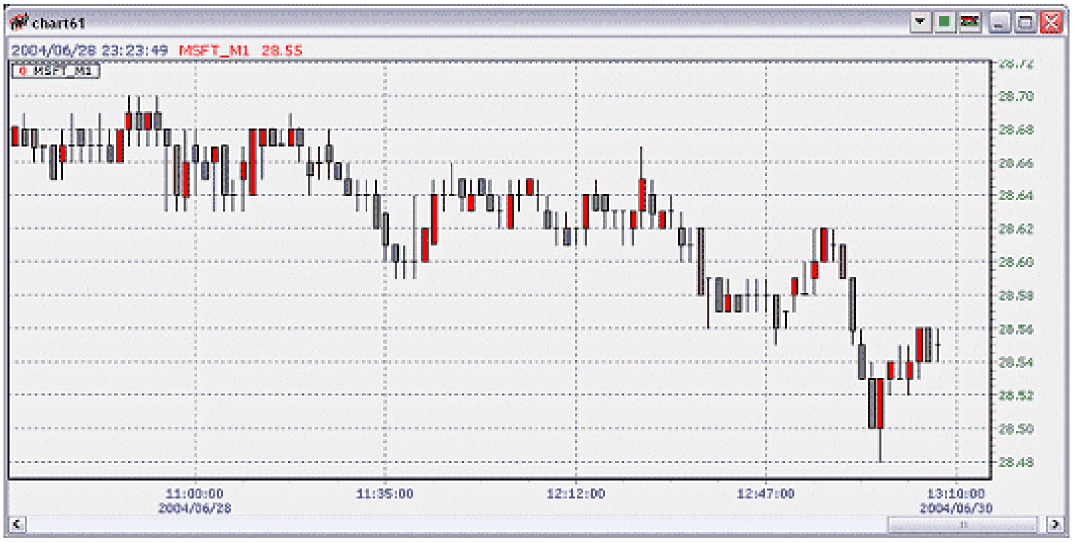
Example - Volume Range Candle

This example is MSFT Volume Range Candle based on 1-min MSFT data. The settings are:

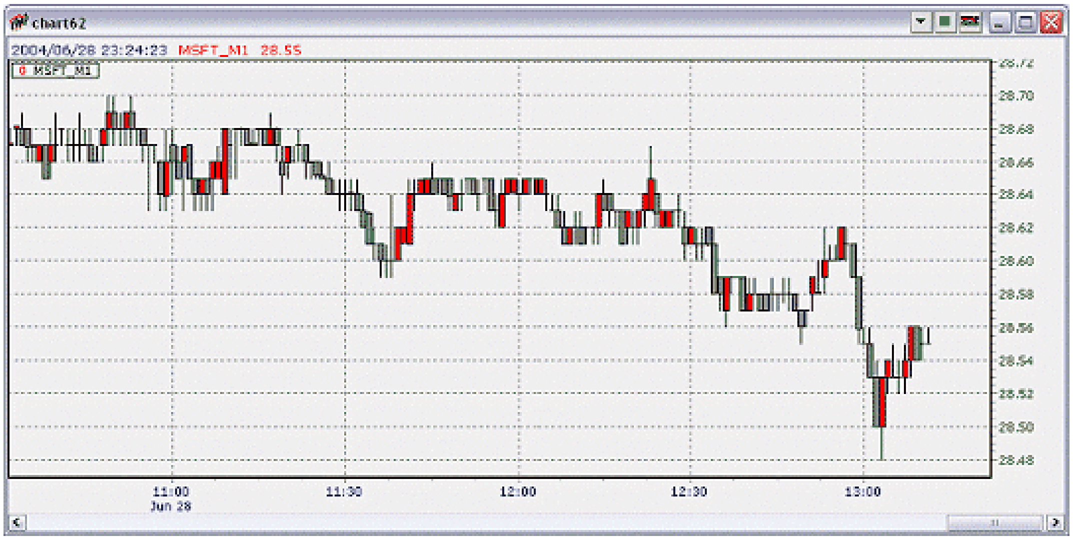
- Visual Style - Candle
- Superposition on, set to Volume Range
- Min Volume - 100000

Therefore, each candle consist of 100000 shares traded. You can see in the chart what is the open, high, low, close within these 100000 shares.

In the chart below, the top pane shows the Volume Range Candle.



Compared to a regular 1-minute chart below that covers the same time period.



Volume Range Candle has much less noise. In fact, the noise level is similar to a regular 2-minute chart. On the other hand, when trading volume is high, Volume Range Candle has the same fast response as a 1-minute chart. So Volume Range Candle has the best features of the faster and slower charts.

How to use Volume Range Candle:

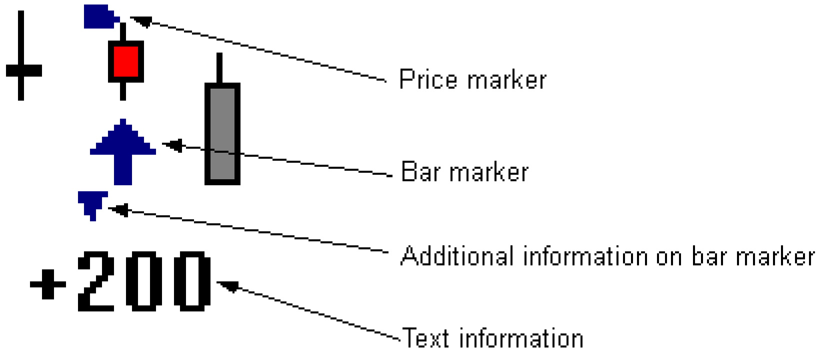
- Volume Range Candle filter out data when there is insufficient trading activities. Because the candles are form only with sufficient volume, Volume Range Candle is particular suitable for support resistance analysis.
- In time driven mode, spacing between bars provides information about how active the trading is. For instance, the bar spacing near closing is very close. This indicates 100000 shares is quickly traded within a short period of time. Because of the trading volume, it is probable that the closing price will form a support/resistance level in the next trading day.

Trading Systems in Charts

Trading System Markings

If a time chart contains one or more trading systems, the trades are marked in the chart.

Reading



The trade is marked on the bar where the trade happens. The figure above is a trading system marking.

The marking consists of various information about the trade. You can customize the markings easily.

Item	Meaning
Price marker	The entry price of the trade
Bar marker	Points at the bar where the trade happens. Up arrow is a buy. Down arrow is a sell.
Additional information on bar marker	This part provides additional information about how the trade affects the current position. Triangles indicate a reverse in position. The direction of the triangle indicates the original position. An up triangle marks a reverse from long to short. A down triangle marks a reverse from short to long. A + sign indicates increase holding. A - sign indicates decrease holding. If there is no previous position, this part is not marked.
Text information	Text information display information such as shares traded, etc.

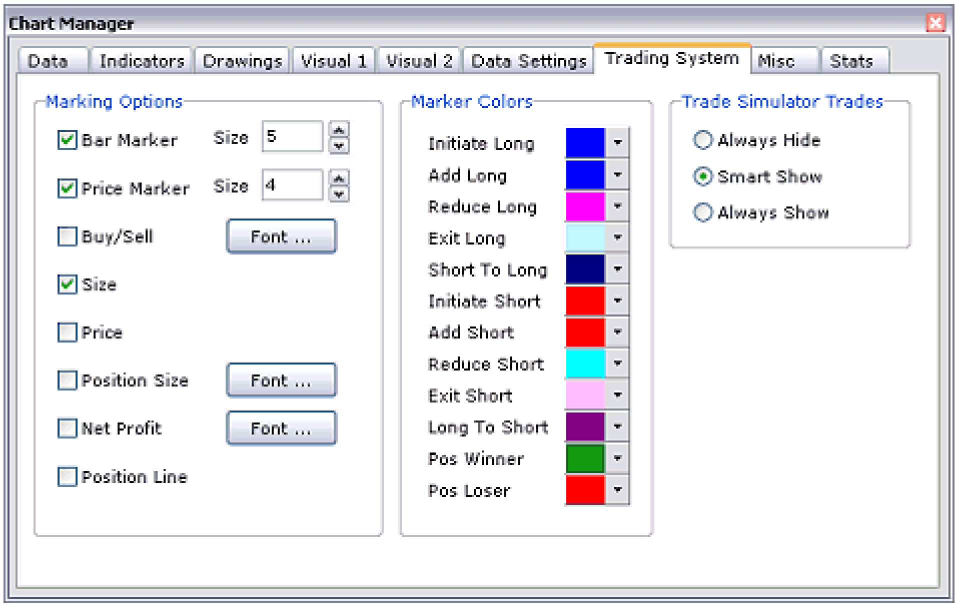
Customization

To customize the markings:

1 Open Chart Manager.

2 Press the **Trading System** tab.

3 Configure the settings.



Marking Options

☑ Bar Marker Size 5

☑ Price Marker Size 4

☐ Buy/Sell Font ...

☑ Size

☐ Price

☐ Position Size Font ...

☐ Net Profit Font ...

☐ Position Line

Marker Colors

Initiate Long	🟦
Add Long	🟦
Reduce Long	🟪
Exit Long	⬜
Short To Long	🟦
Initiate Short	🟥
Add Short	🟥
Reduce Short	🟦
Exit Short	⬜
Long To Short	🟪
Pos Winner	🟩
Pos Loser	🟥

Trade Simulator Trades

○ Always Hide

● Smart Show

○ Always Show

The following items are customizable for trading system markings:

Item	Meaning
Bar Marker	Toggles bar marker.
Price Marker	Toggles price marker.
Buy/Sell	Whether the word 'Buy' or 'Sell' appears in text information.
Size	Whether the trade size appears in text information.
Price	Whether the entry price appears in text information.
Position Size	Whether the position size appears in text information.
Net Profit	Whether the net profit of the trade appears in text information.
Position Line	A position line is drawn between first entry and last exit of position.
Marker Colors	The color of the marker is determined by how the trade affects the position. For example, by default, when a trade is entered to initiate a long position, the marker will be a blue up arrow. **Initiate Long**, **Add Long**, **Reduce Long**, **Exit Long**, **Short to Long** - colors to use for markings related to long positions. **Initiate Short**, **Add Short**, **Reduce Short**, **Exit Short**, **Long to Short** - colors to use for markings related to short positions. **Pos Winner**, **Pos Loser** - colors to use for position line.

Trade Simulator Trades

Whether trades from the trade simulator will appear in this chart.

Always Hide - Do not draw trades from Trade Simulator.

Smart Show - Draw Trade Simulator trades only when there are no trading system in the chart. This setting avoids mix up of trade markers drawn by Trade Simulator and trading systems.

Always Show - Always draw trade simulator trades, regardless of whether there are trading systems in the chart.

Trading System Reporting

In addition to report trading system in scripts, there are additional ways to launch the reporting. You can create trading system report when you select an indicator that has trading system UI turned on.

System Performance Viewer

This is the preferred method of reporting for most users.

To open system performance viewer, select the trading system indicator. Right click on the chart to open the pop up menu and choose **Trading System>Open Performance Viewer**.

For more information about system performance viewer, go to ***System Performance Viewer*** (on page 951).

Manual Reporting to Report Window

Select the trading system indicator. Right click on the chart to open the pop up menu and choose any of the following reports under **Trading System**:

Item	Meaning
Quick Status	A quick report on the current status of the trading system.
Current Status	A longer report on the current status of the trading system.
Open Positions	Current open positions.
Order List	Orders the system has placed.
Trade List	Trades the system has executed.
Equity List	Tracking down the equity on a bar-by-bar basis.
Trades Summary	Statistics of the trades.
System Summary	Statistics of the trading system.

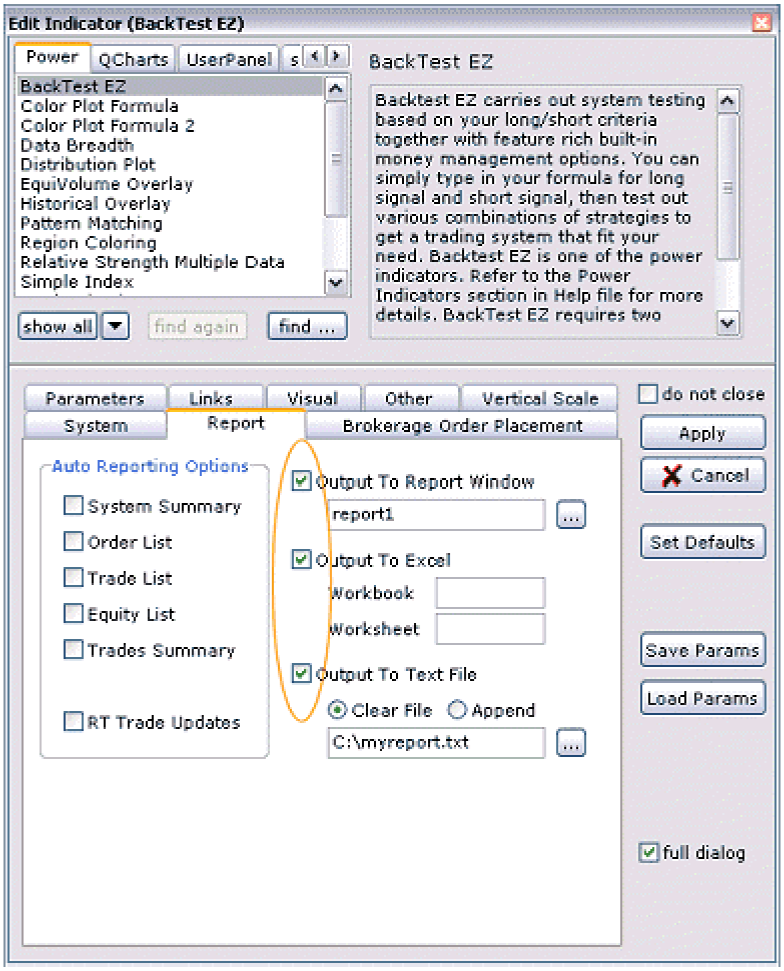
By default, these reports go to the first report window in the active group. You can change report destination by choosing **Trading System>Report Options** in the pop up menu.

Auto Reporting to Report Window, Text File and Excel

You can specify automatic trading system reporting.

1. Select the trading system indicator.
2. Press the space bar to edit it.
3. Press the **Report** tab.

4 You can set the automatic reporting options here. Reports is sent to the destination when the trading system is executed.



BackTest EZ
Color Plot Formula
Color Plot Formula 2
Data Breadth
Distribution Plot
EquiVolume Overlay
Historical Overlay
Pattern Matching
Region Coloring
Relative Strength Multiple Data
Simple Index

BackTest EZ

Backtest EZ carries out system testing based on your long/short criteria together with feature rich built-in money management options. You can simply type in your formula for long signal and short signal, then test out various combinations of strategies to get a trading system that fit your need. Backtest EZ is one of the power indicators. Refer to the Power Indicators section in Help file for more details. BackTest EZ requires two

Auto Reporting Options

□ System Summary
□ Order List
□ Trade List
□ Equity List
□ Trades Summary
□ RT Trade Updates

☑ Output To Report Window: report1

☑ Output To Excel
Workbook:
Worksheet:

☑ Output To Text File
● Clear File ○ Append
C:\myreport.txt

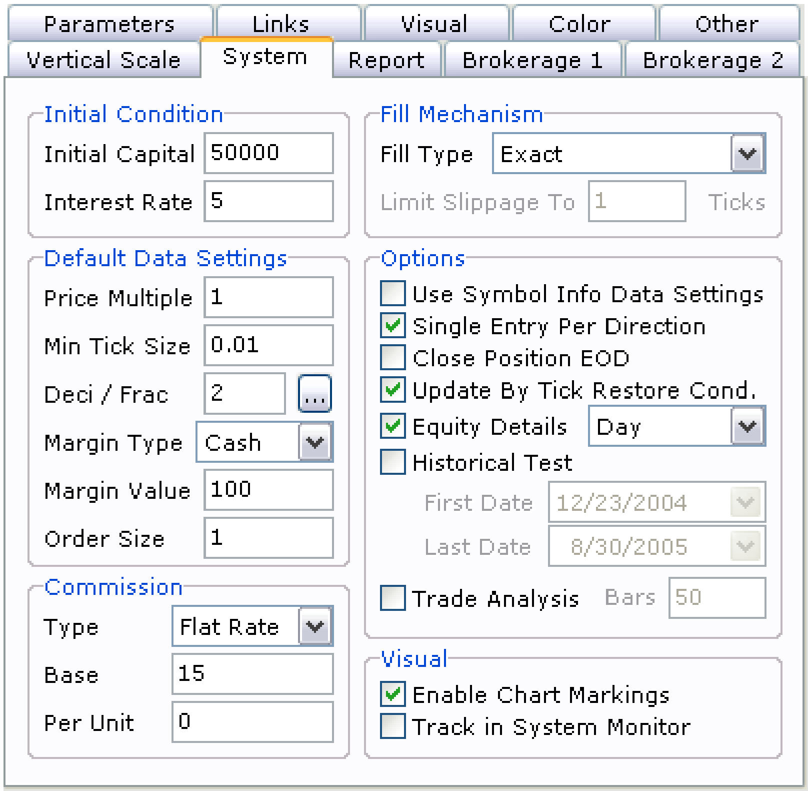
□ do not close
☑ full dialog

Trading System Settings

Trading system is a specialized type of indicator, for example ***Backtest EZ*** (on page 1275).

Because trading system is a specialized type of indicator, you can edit a trading system with just like an indicator, i.e. you can open editor by selecting the indicator and press the space bar.

Many trading systems provide an additional **System** tab in the indicator editor for setting up parameters that are specific to trading systems.



Initial Condition

Initial Capital 50000

Interest Rate 5

Default Data Settings

Price Multiple 1

Min Tick Size 0.01

Deci / Frac 2 ...

Margin Type Cash

Margin Value 100

Order Size 1

Commission

Type Flat Rate

Base 15

Per Unit 0

Fill Mechanism

Fill Type Exact

Limit Slippage To 1 Ticks

Options

□ Use Symbol Info Data Settings

☑ Single Entry Per Direction

□ Close Position EOD

☑ Update By Tick Restore Cond.

☑ Equity Details Day

□ Historical Test

First Date 12/23/2004

Last Date 8/30/2005

□ Trade Analysis Bars 50

Visual

☑ Enable Chart Markings

□ Track in System Monitor

Initial Condition

Initial conditions of the trading system.

Initial Capital

Initial capital of the trading system.

Interest Rate

Interest rate is used for Sharpe ratio calculation.

Default Data Settings

Settings under this group specifies the how the symbol trades in the real world. If the symbol is already specified in ***Symbol Info Manager*** (on page 921), you can skip these settings and turn on **Use Symbol Info Data Settings** instead.

Price Multiple

Price multiple for the instrument. Set to 1 for stocks ($1). Set to 50 for S&P e-mini ($50 per point).

Min Tick Size

Minimum movement of the instrument. Set to 0.01 for stocks ($0.01). Set to 0.25 for S&P e-mini (0.25 point).

Margin Type

Margin Type is set to either Cash, Percent or Amount. Margin is used to provide available cash information to trading systems that take margin into account.

Cash - no margin is taken. Available cash equals the actual cash.

Percent - percentage of the face value of the position. This is typically used for stock trading systems.

Amount - exact dollar amount. This is typically used for future contracts trading systems.

Margin Value

Margin Values provides the value for percent and amount margin types.

Order Size

Order Size specifies the default order size (e.g. 100 for stocks, 1 for futures). Some trading systems do not specify an absolute order size, but relying on this value when placing orders.

Commission

Commission Type

Commission Type is set to either Flat Rate, Per Unit or None. Commission is used to calculate the per order cost of trades.

When set to Flat Rate, commission equals **Base** value. This is suitable for stock trading systems.

When set to Per Unit, commission equals **Base** value plus the number of units (contracts/stocks) times the **Per Unit** value. This is suitable for some stock trading systems and futures trading systems.

When set to None, no commission is charged. This is suitable if you want to publish the trading system results to many people who may not share the same commission schedule.

Fill Mechanism

Fill mechanism controls the fill price of the trades.

Fill Type is set to either Exact, Average, Worst Case or Limited Worst Case.

Limit Slippage To value is use specify maximum damage when fill type is set to Limited Worst Case.

For the full explanation of how fill type works, refer to ***How Orders are Filled*** (on page 1197).

Options

This group contains additional options for the trading system.

Use Symbol Info Data Settings

If the symbol is entered into Symbol Info manager, you can check this option. In this case, information such as price multiple, tick size, will be retrieved from Symbol Info Manager.

Single Entry Per Direction

If checked, the system will allow only one entry (short or long) per direction. See ***Order Defaults and Completion Related Topics, Single Entry Per Direction*** (see "Single Entry Per Direction" on page 749).

Close Position EOD

If checked, he system will close all position at end of trading day. This option is useful for real-time trading system only.

Update By Tick Restore Cond.

The trading system will restore to the bar's initial state on a tick-by-tick basis until the bar is finished. This provides maximum consistency between historical testing and real-time running of the trading system in simulation.

Read the following paragraphs carefully regarding update by tick and wiring to real-life broker.

Update by tick restore condition is not possible with real-life brokers. If your trading system is wired to a real-life broker, you should turn off update by tick, and this option becomes irrelevant.

If you turn on update by tick, you can use this option to control whether the trading system will restore the orders. When off, your trading system is expected to manage all orders. DO NOT turn on update by tick unless you are absolutely certain that the trading system manages orders properly.

If you choose to turn on update by tick and turn off update by tick restore condition, you should always test the trading system first with Trading Simulator before wiring it to a real-life broker.

Equity Details

This setting affects the resolution of the equity curve stored in the trading system. This setting only affects the equity graph report in System Performance Viewer. It is safe to leave the setting to the default **Day**. It does not affect trading system calculation or current equity reporting.

Historical Test

Limits the trading system to trade between the specified dates.

Trade Analysis

For enabling system performance analysis' trade analysis feature to work.

Visual

The settings here controls how the trading system is displayed in the chart. These options do not affect the actual calculation of the trading system.

Enable Chart Markings

Trading systems entries will be marked on the chart.

Track in System Monitor

The trades are tracked in system monitor.

How Orders are Filled

System testing is a simulation of trading system using historical data. The actual trades are no longer present during system testing and certain rules are needed to determine how orders are filled. Therefore, order filling is a simulation and it is never 100% accurate. However you can set different order filling rules to analyze a trading system under different scenarios.

NeoTicker® uses the same order filling mechanism for trading systems implemented in different technologies (Backtest EZ, formulas, scripts).

Fill Type

Fill type controls how a trade is filled. By changing the fill behavior you can get a sense of sensitivity of the trading system related to entry conditions. Better trading systems usually are insensitive to fill results, even under the worst condition.

Fill Type	Meaning
Worst Case	use the worst possible price of the next bar as the filled price
Average	use the average of the next bar as the filled price
Exact	use the exact price based on the type of order as the filled price
Limited Worst Case	Similar to worst fill, except the worst case is limited by the value specified by the slippage value.
Bid Limited Worst Ask Limited Worst	These fill types are specific to testing Forex trading systems. Worst case is estimated using spread. If the Forex trade data comes from bid, use Bid Limited Worst. If the Forex trade data comes from ask, use Ask Limited Worst.

Orders Activated on Next Bar

After orders are placed at a specific bar, it is activated at the beginning of the next bar. Thus the first bar that an order can get filled is the bar right after the order is placed.

Transactions are Available on Filled Bar

If the order is filled on a particular bar, a transaction is generated on this bar.

If you use script/IDL to implement the trading system, you can access this transaction information in your script within the same bar. You can then have rules of money management to generate orders to protect the position based on the transaction just happened.

Minimum Tick Size

Orders are filled at prices rounded towards the nearest tick specified by minimum tick size. For example, if minimum tick size is 0.05, orders are filled at 1.00, 1.05, 1.10, etc.

No “On Close” Orders

There is no filling mechanism that tries to fill with the close of the current bar. It is against the principle of not allowing the act of peeking into the future. Remember you have access to all the data of the current bar, which includes the close of the bar. Close of a bar in principle is not known to the trading system until data for the next bar has arrived.

By having on close orders available, a trading system can check the close for profitability first, then place the on close order to close out a position with profit.

The way to make an order with close price is by using the next close orders . These ordering methods allow you to place orders that will be filled at next bar's close. Next bar close orders do not peek the close price of current bar, yet provide a mechanism for filling an order based on close price.

Rules of Order Filling

The following are the rules how the price for a fill is determined.

Limited Worst Case

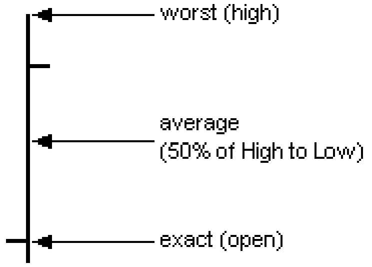
Limited worst case is similar to worst case, but provide a limit on the worse scenario.

In limited worst case, the fill price is either:

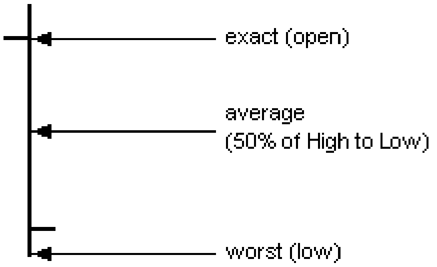
- the worst price, or
- exact plus/minus the slippage value (Order object property)

The better fill is used.

Buy at Market

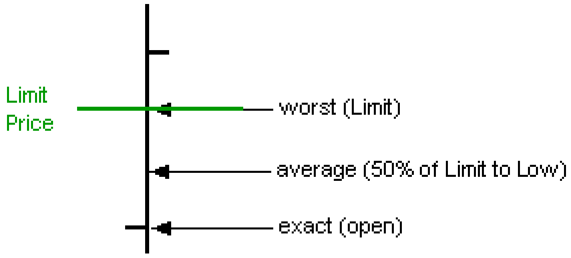


Sell at Market

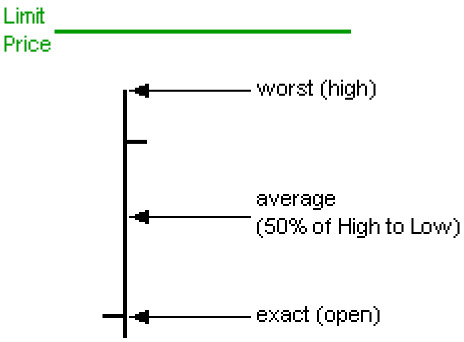


Buy Limit

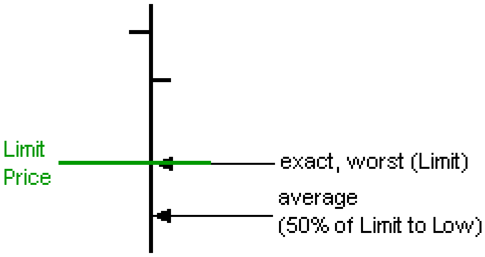
Case 1



Case 2

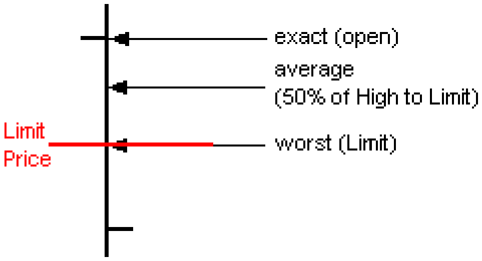


Case 3

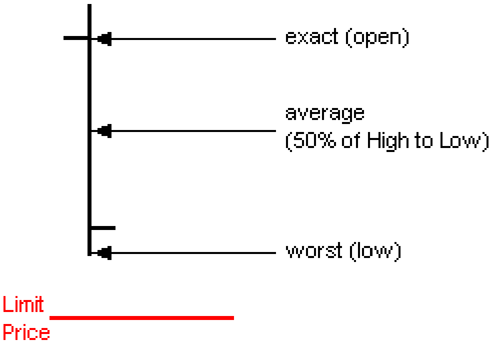


Sell Limit

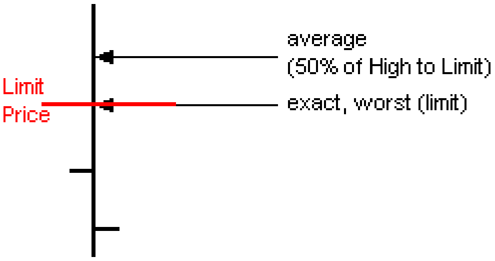
Case 1



Case 2

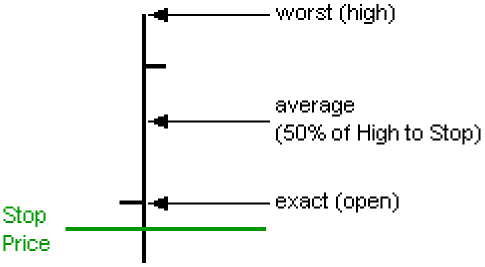


Case 3

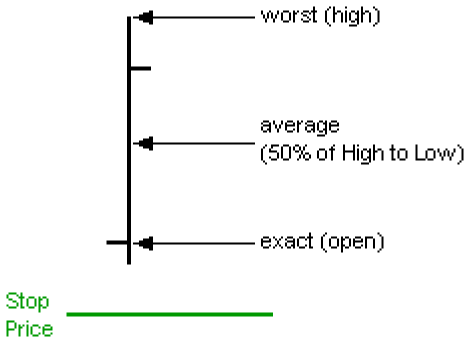


Buy Stop

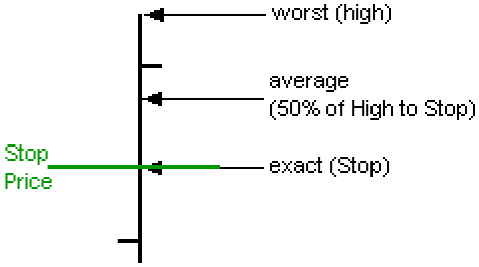
Case 1



Case 2

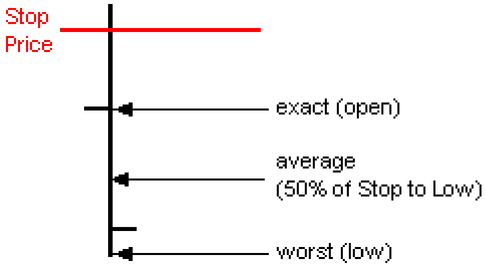


Case 3

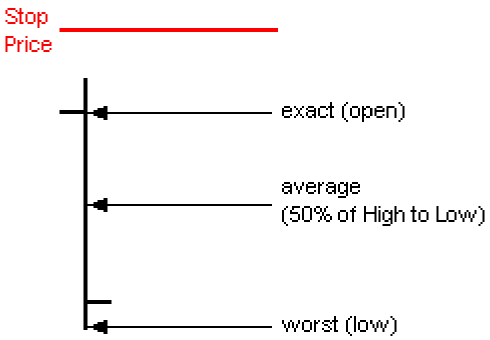


Sell Stop

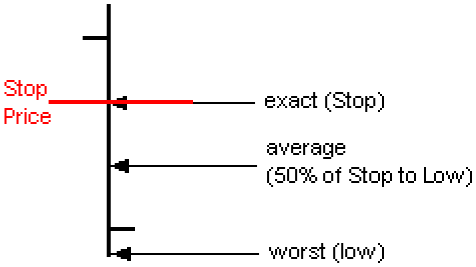
Case 1



Case 2



Case 3



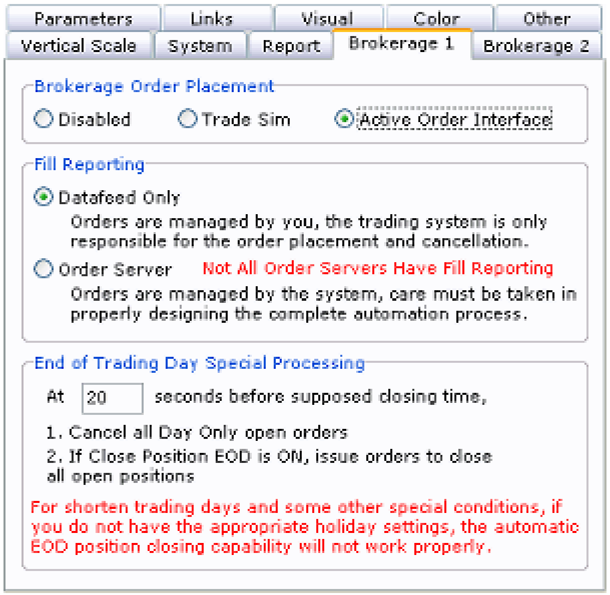
Sending Orders to Your Broker

Placing orders to a broker is a high risk operation. Make sure you read and completely understand the material in ***Trading System Life Deployment Guide*** (see "Trading System Live Deployment Guide" on page 785) before you proceed.

If you intended the orders to be placed automatically to a broker, you will need to:

1. Make sure ***Order Interface*** (on page 735) routes the orders to Trade Simulator. This way, if something is wrong with the trading system, the orders are sent to Trade Simulator, not a real-life broker.
2. Open the editor for the indicator (trading system).
3. Press the **Brokerage 1** tab.
4. Under **Brokerage Order Placement**, check **Active Order Interface**. By turning on this option, you are switching from system testing to deployment.
5. Review the options under **Brokerage 1** and **Brokerage 2** tab.

6 By default, Broker Interface is connected to Trade Simulator for deployment testing. Observe the trading system. Only after you are completely satisfy with the trading system's behavior, you can configure Order Interface to route the orders to a real-life broker. Refer to ***Connecting to Your Broker*** (on page 49) for instruction.



Brokerage Order Placement

○ Disabled ○ Trade Sim ● Active Order Interface

Fill Reporting

● Datafeed Only

Orders are managed by you, the trading system is only responsible for the order placement and cancellation.

○ Order Server Not All Order Servers Have Fill Reporting

Orders are managed by the system, care must be taken in properly designing the complete automation process.

End of Trading Day Special Processing

At 20 seconds before supposed closing time,

1. Cancel all Day Only open orders
2. If Close Position EOD is ON, issue orders to close all open positions

For shorten trading days and some other special conditions, if you do not have the appropriate holiday settings, the automatic EOD position closing capability will not work properly.

While trading system is being evaluated using historical data and in real-time, only orders generate in real-time is sent to your broker. Orders generated using historical data are never sent to your broker. Historical orders are "filled" just like when you are doing system testing, i.e. using trading system's own filling mechanism. They are there as a background for your trading system to continue on working in real-time.

You can manage orders sent from trading systems in Account Manager. See ***Account Manager*** (on page 767), under ***Account Manager, Live Systems*** (see "Live Systems" on page 776) section.

Fill Reporting

If you choose **Datafeed Only**, in the chart, the trading system orders are filled using fill mechanism, i.e. the fill is a simulation.

If you choose **Order Server**, the fill success and fill price is feed backed from the broker, i.e. the fill is realistic. However, not all Broker Interface support this feature. Consult TickQuest if you have questions.

The following Broker Interface support back fill, i.e. you can choose **Order Server** for realistic fill price:

- MB Trading
- Interactive Brokers, except futures and Forex
- Ninja Trader
- Trade Simulator

You can consult TickQuest support if you are not sure which option to set.

End of Trading Day Special Processing

At end of trading day, some special processing needed to be done for a trading system:

- Cancelling Day Only orders - orders that are specified to be filled or killed by end of trading day.
- Closing all open positions if **Close Position EOD** is specified under **System** tab.

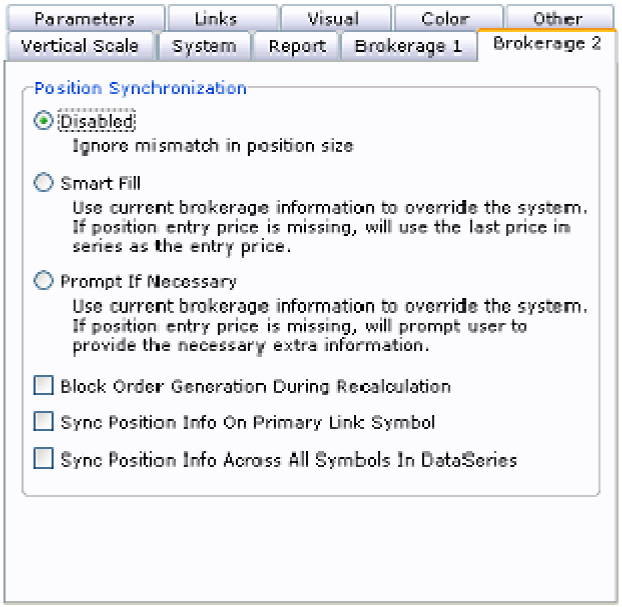
You will need to specify a time when these actions are carried out when the trading system is connected to a real-life broker. The exact time required is broker dependent. You will need to allocate enough time for:

- Your broker to carry out these actions.
- You to verify these actions and manually correct any mistakes. This is especially important for order cancelling because in practice, there is no guarantee orders can be successfully cancelled.

If set to 0, end of day special processing will not be activated.

End of Trading Day Special Processing is not applicable to 24-hour charts.

Position Synchronization



Position Synchronization

● Disabled
Ignore mismatch in position size

○ Smart Fill
Use current brokerage information to override the system. If position entry price is missing, will use the last price in series as the entry price.

○ Prompt If Necessary
Use current brokerage information to override the system. If position entry price is missing, will prompt user to provide the necessary extra information.

□ Block Order Generation During Recalculation

□ Sync Position Info On Primary Link Symbol

□ Sync Position Info Across All Symbols In DataSeries

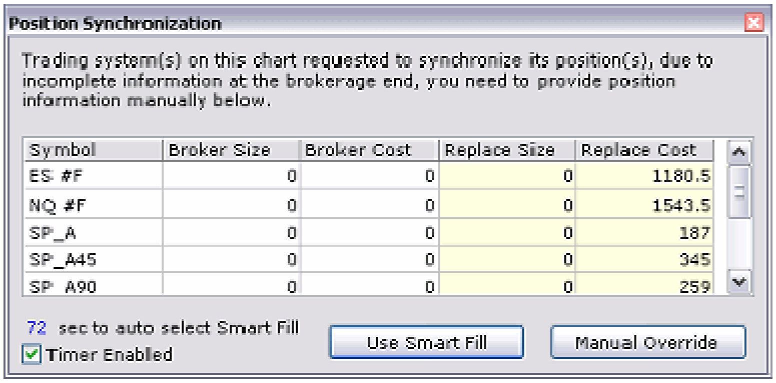
When deploying a trading system, ideally the trading system position should match the broker position (position at your brokerage account). This matching is especially important for trading systems that perform position sizing.

When the trading system is being evaluated, right at the transition between historical calculation and real-time calculation, an attempt is made to synchronize the trading system position (position resulted from historical orders from the trading system) and broker position. This synchronization action is controlled by the options under **Brokerage 2** tab.

Disabled - ignore mismatch position size. All orders will be sent. This is the default.

Smart Fill - trading system position is adjusted automatically to match that of the broker. This option requires the Broker Interface to supply position entry price. Not all brokers provide position entry price. Therefore, Smart Fill can fail to match the positions. In this case, no synchronization is performed.

Prompt If Necessary - Similar to **Smart Fill**, except when smart synchronization fails, a dialog is open to let you fill in the size and cost of position.



Trading system(s) on this chart requested to synchronize its position(s), due to incomplete information at the brokerage end, you need to provide position information manually below.

Symbol	Broker Size	Broker Cost	Replace Size	Replace Cost
ES #F	0	0	0	1180.5
NQ #F	0	0	0	1543.5
SP_A	0	0	0	187
SP_A45	0	0	0	345
SP_A90	0	0	0	259

72 sec to auto select Smart Fill
☑ Timer Enabled

Block Order Generation During Recalculation - When evaluating historical data, no order is generated, i.e. trading system will start flat in real-time. This option is useful for short term systems (e.g. scalping systems) where historical position is unimportant. This option is meaningful only when synchronization is enabled.

Sync Position Info On Primary Link Symbol - Synchronize the position for the symbol that is the primary link of the trading system. This option is for single instrument trading systems. This option is meaningful only when synchronization is enabled.

Sync Position Info Across All Symbols In DataSeries - Synchronize the positions for all data series in the chart, regardless whether the data series is linked to the trading system. This option is for portfolio trading systems.This option is meaningful only when synchronization is enabled.

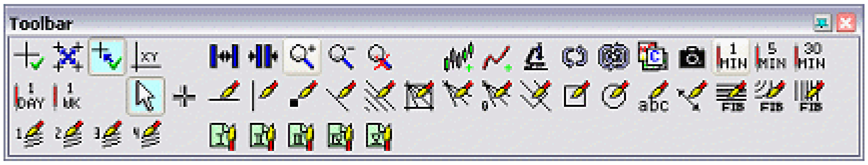
Trading system position and broker position can become mismatch when a system running in real-time. These mismatches are not handled by options here. You can correct these mismatch in Account Manager. See ***Account Manager, Live Systems*** (see "Live Systems" on page 776).

For a big picture discussion on position mismatch, see ***Position Size Matching*** (on page 745).

Order Confirmation

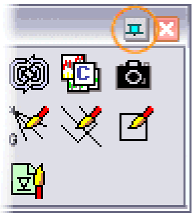
By default, all orders sent from trading systems require confirmation in Account Manager. See ***Account Manager, Confirming Orders*** (see "Confirming Orders" on page 768).

Tool Bar



Tool bar provides many time chart functions. If the tool bar is hidden, you can use the F4 hot key to open it. You can also use the tool bar button on the user panel to show the tool bar.

Pin Button



Tool bar has a pin button besides the close button on the upper right corner.

If the pin is down (default), the tool bar will not close after you select an operation. This is true for most operations except for the zooming buttons.

If the pin is up , the tool bar will close after you select an operation.

Pin button up is useful for user who wants to open up the tool bar quickly using F4 to select an operation and wants the tool bar to go away afterwards.

Customizing the Tool Bar

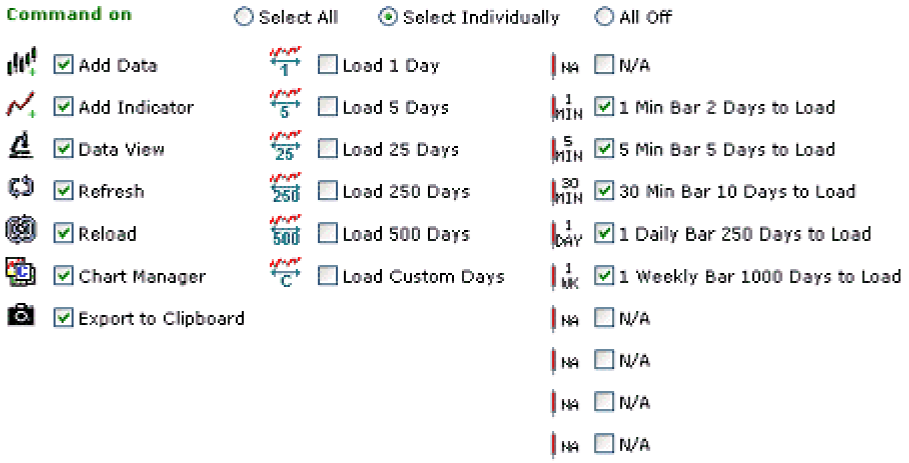
To customize the buttons on the tool bar:

1 Right click on the tool bar.

2 Choose **Setup.** This will open the tool bar set up.

The buttons are organized into button groups. The first tab let you decide whether you want to show all button of all groups on the tool bar. Usually you want to select individual buttons, so leave **Select Button Groups Individually** checked.

You can turn individual buttons on and off in a button. The following figure shows time chart related buttons. In the **Command** button group, you can see that some buttons are configured on and some are configured off.



Command on

○ Select All ● Select Individually ○ All Off

☑ Add Data
☑ Add Indicator
☑ Data View
☑ Refresh
☑ Reload
☑ Chart Manager
☑ Export to Clipboard

□ Load 1 Day
□ Load 5 Days
□ Load 25 Days
□ Load 250 Days
□ Load 500 Days
□ Load Custom Days

□ N/A
☑ 1 Min Bar 2 Days to Load
☑ 5 Min Bar 5 Days to Load
☑ 30 Min Bar 10 Days to Load
☑ 1 Daily Bar 250 Days to Load
☑ 1 Weekly Bar 1000 Days to Load
□ N/A
□ N/A
□ N/A
□ N/A

To quickly setup the tool bar, you can use the pop up menu rather than the tool bar setup window.

Customizing F4 Behavior

Open the tool bar manager by choosing **Manager>Toolbar** in the main window.

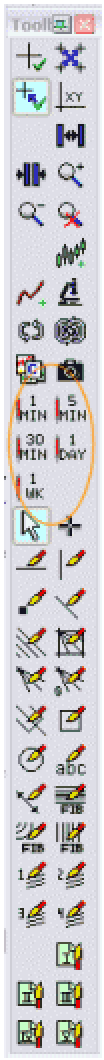
You can change which tool bar F4 brings up under **F4 Shortcut**.

Under **F4 Options**, you can force the tool bar F4 brings up to move to the active window. This feature is especially useful for users with multiple monitors.

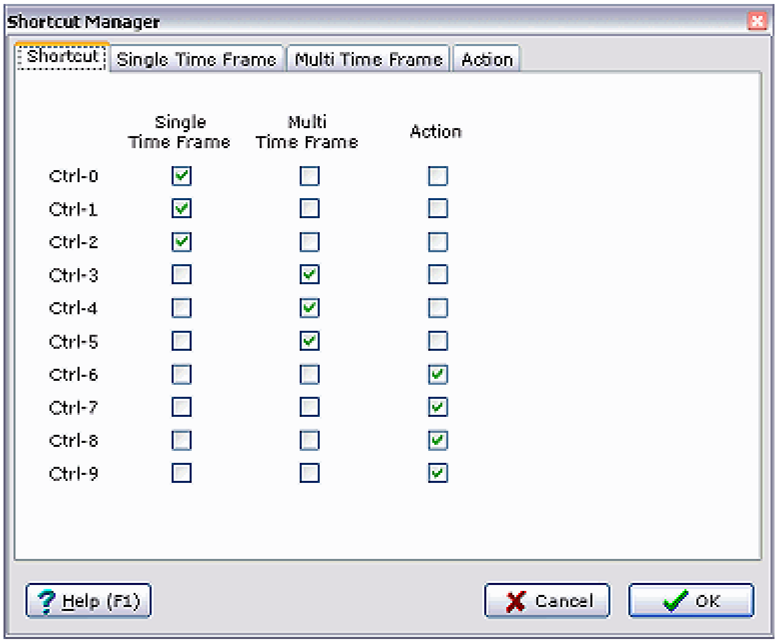
How the Tool Bars Work with the Shortcut Manager

Certain tool bar buttons are related to the chart shortcuts. The shortcuts are configured by the shortcut manager. These shortcuts are used to quickly change the time frame and loading days of a chart. For a complete discussion of the shortcut manager, go to ***Shortcut Manager Operation Guide*** (on page 911).

The tool bar buttons that are related to the shortcut manager are shown below:



For these buttons, because the meaning and consequently the look changes with the shortcut manager, the shortcut manager provides a preview of these buttons.

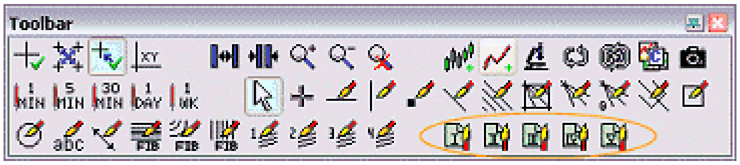


Shortcut Manager

	Single Time Frame	Multi Time Frame	Action
Ctrl-0	☑	□	□
Ctrl-1	☑	□	□
Ctrl-2	☑	□	□
Ctrl-3	□	☑	□
Ctrl-4	□	☑	□
Ctrl-5	□	☑	□
Ctrl-6	□	□	☑
Ctrl-7	□	□	☑
Ctrl-8	□	□	☑
Ctrl-9	□	□	☑

Custom Drawing Tool Buttons

Custom drawing tools can be loaded into the custom drawing tools buttons. Custom drawing tools are user defined drawing tools (for example, a script that defines your version of Fibonacci lines).



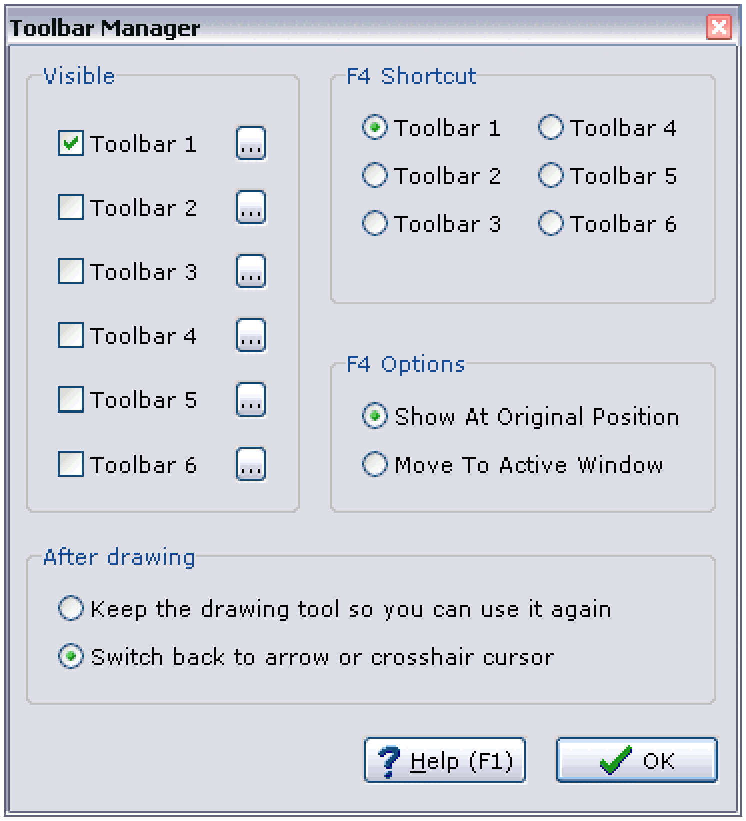
Press the define button lets you load a custom drawing tool into the button. For more information about custom tools, go to ***Custom Drawing Tools Button*** (see "Custom Drawing Tools" on page 1173).

Tool Bar Manager

Tool Bar Manager provides a centralized view of all the available tool bars.

To launch the Tool Bar Manager choose **Manager>Toolbar** in the main window.

The Tool Bar Manager will be shown with your current settings.



Toolbar Manager

Visible

☑ Toolbar 1
□ Toolbar 2
□ Toolbar 3
□ Toolbar 4
□ Toolbar 5
□ Toolbar 6

F4 Shortcut

● Toolbar 1 ○ Toolbar 4
○ Toolbar 2 ○ Toolbar 5
○ Toolbar 3 ○ Toolbar 6

F4 Options

● Show At Original Position
○ Move To Active Window

After drawing

○ Keep the drawing tool so you can use it again
● Switch back to arrow or crosshair cursor

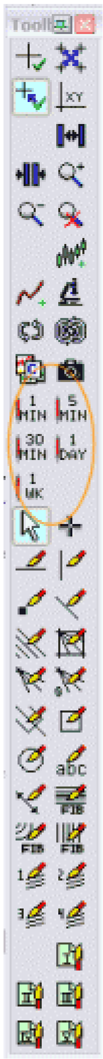
How to Add More Tool Bars

Only the first tool bar is visible by default. You can add more tool bars by turning on the visibility of the other tool bars.

1 Open the tool bar manager by choosing **Manager>Toolbar** in the main window.

2 Check the option for the tool bar you want to turn visible under **Visible.**

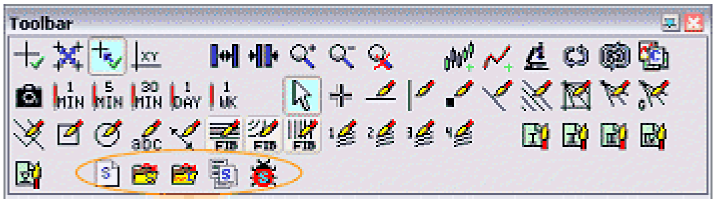
Shortcut Buttons



These buttons provide the same functionality as those short cuts defined in the shortcut manager. To find out more, go to ***How the Tool Bars Work with the Shortcut Manager*** (on page 1211).

Script Buttons

These buttons let you quickly work with scripts.



Creates a new script

Opens an existing script

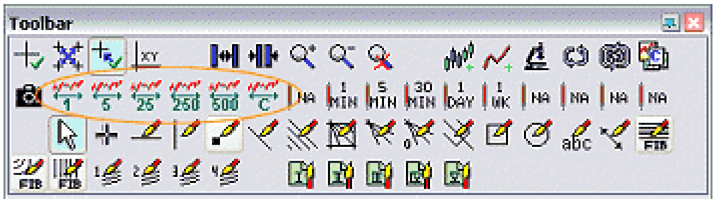
Opens indicator with Script Editor

Opens Indicator Manager

Opens Script Error Log

Days to Load

These buttons let you quickly control the number of days to load.

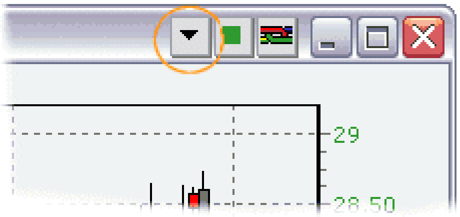


Pop up Menu Reference

The time chart window displays a pop up menu when you right click on the chart. Items on the pop up menu depends on:

- If right clicking not on a data series, indicator or drawing tool, the general pop up menu is opened. See ***Pop up Menu when Nothing is Selected (General Pop Up Menu)*** (on page 1218).
- If right clicking on a data series/indicator, or when a data series/indicator is currently selected, the pop up menu for data series/indicator is opened. See ***Pop up Menu when a Data Series or an Indicator is Selected*** (on page 1220).
- If right clicking on a drawing tool, or when a drawing tool is currently selected, the pop up menu for drawing tool is opened. See ***Pop up Menu when a Drawing Tool is Selected*** (on page 1222).

You can also press the menu button on the caption bar. In this case, only the general menu (i.e. the menu when nothing is selected) is open, regardless of the chart selection.



Some pop up menu services such as send symbol, save window are common to all function windows (quote, time chart, etc) are not discussed in this section.

Pop up Menu when Nothing is Selected (General Pop Up Menu)

Pop up menu item	Usage
Add Data	Adds a data series to the current pane.
Add Indicator	Adds an indicator to the current pane.
User Panel	Turns user panel on/off.
Dynamic Grid	Turns dynamic grid on/off.
Add Alert	Adds an alert to the chart.
Alert List	Shows the current list of alerts.
Window Style	Opens the chart manager and switches to **Visual 1** tab, which is related to window style.
Bar Spacing	Opens the chart manager and switches to **Visual 2** tab, which is related to bar spacing.
Chart Manager	Opens the chart manager window.
Object Ordering	Lets you re-order the display order of data series and indicators.
Days of Data to Load	Changes days to of data load the selection include Custom, 1 Day, 5 Days, 25 Days, 250 Days and 500 Days.
Visual > Grid	Shows/hides chart grid.
Visual > Legend / Label	Shows/hides legend for data series and indicators.
Visual > Wide Price Axis	Adds more space on the right of the price axis to accommodate a large price axis values (e.g. volume)
Visual >Floating Marker	Shows/hides floating marker which display the price of the final bar of a data series or indicator (Note: When nothing is selected Floating Marker of the whole chart will be enabled, hence float marker of all data series and indicators will be enabled)
Visual > Time Scroll Bar	Shows/hides the time axis scroll bar
Pane / Price Axis > Add	Add a pane below the current pane

Pane / Price Axis > Delete Deletes the current pane

Pane / Price Axis > Edit Edit the current pane style, this include price axis scale and price label setting.

Pane / Price Axis > Arithmetic Changes the value axis of the current pane to arithmetic (linear) scaling.

Pane / Price Axis > Logarithmic Changes the value axis of the current pane to logarithmic scaling.

Pane / Price Axis > Auto price scale When chosen, value axis of the pane is scaled to fit all visible data.

Pane / Price Axis > Fixed price scale Lets you specify a fixed range of values for the value axis.

Font Font settings.

Global Crosshair Global crosshair settings.

More Bar Space Zooms in to the chart (less effect than regular zoom).

Less Bar Space Zooms out from the chart (less effect than regular zoom).

Zoom In Zooms in to the chart.

Zoom Out Zooms out from the chart.

UnZoom Resets the zooming to default zoom level.

Chart Tool Window Opens/Closes the tool bar window.

Data View Window Opens/Closes the data view window.

Refresh Chart Refreshes the chart by recalculating all the indicators.

Export Exports screen shot to file or clipboard

Save Pattern Save the first data series in a format that is compatible to TickQuest's pattern scanning technology. This pattern can be used in the pattern scanner window

Pop up Menu when a Data Series or an Indicator is Selected

When a data series or indicator is selected, the following menu items are available.

Pop up menu item	Usage
Add Indicator	Adds an indicator to the current pane. The selected series is the first link to the selected data series or indicator.
Add Recent Indicators	Add a recently used indicator to the current pane.
Edit	Edits the selected data series series or indicator.
Window Style	Opens the chart manager and switches to **Visual 1** tab, which is related to window style.
Bar Spacing	Opens the chart manager and switches to **Visual 2** tab, which is related to bar spacing.
Chart Manager	Opens the chart manager window.
Object Ordering	Lets you re-order the display order of data series and indicators.
Hide	Hides the selected data series or indicator.
Delete	Deletes the selected data series or indicator.
Style	Changes the plotting style of the selected data series or indicator.
Part of Pane Scale	Pane / Price Axis will automatically scale to fit the selected data series or indicator.
Overlay	Makes selected data series or indicator an overlay, which means the selected data series is price axis independent.
Auto Float Marker	Show / hide price marker for selected data series or indicator. Note: **Floating Marker** must be enable at chart level.
Export Series	Exports the data series or indicator to a file or another application.

Additional Items Available When Data Series is Selected

When a data series is selected, the following additional menu item is available:

Pop up menu item	Usage

Repair Data	Repairs data by retrieving the data from data feed. This feature should only be used with Internet data feed with historical database.

Additional Items Available When Indicator is Selected

When an indicator is selected, the following additional menu items are available:

Pop up menu item	Usage
Open with Script Editor	If the indicator is written with script or formula, selecting this item will open the source code of the indicator in a script editor.
Trading System	If the indicator is a trading system (e.g. Backtest EZ), a wide range of trading system related tools are available under this item.

Pop up Menu when a Drawing Tool is Selected

Pop up menu item	Usage
Edit	Edits the selected drawing tool
Snap	Snaps drawing tool once
Snapping Properties	Allows you to set up the snapping properties of the drawing tool
Label Properties	Opens window for editing label properties. See ***Configuring Drawing Object Labels*** (see "Configuring Drawing Tool Labels" on page 1133).
Load From Template	Applying a template to current drawing tool. See ***Drawing Tool Templates*** (on page 1143).
Window Style	Opens the chart manager and switches to **Visual 1** tab, which is related to window style
Bar Spacing	Opens the chart manager and switches to **Visual 2** tab, which is related to bar spacing
Chart Manager	Opens the chart manager window
Flip Horizontally	Flips the selected drawing object horizontally
Flip Vertically	Flips the selected drawing object vertically
Copy	Duplicates the selected drawing tool. Snapping properties are not copied.
Copy with Snapping Properties	Duplicates the selected drawing tool. Snapping properties are copied.
Delete	Deletes the selected drawing tool
Hide	Hides the selected drawing tool

Keyboard Shortcuts

Time chart supports the following keyboard shortcuts:

Key	Function
Numeric +	Zoom in key
Numeric -	Zoom out key
SHIFT Numeric +	Zoom in (1/20 of normal power)
SHIFT Numeric -	Zoom out (1/20 of normal power)
CTRL Z	Unzoom
Left/Right arrows	Scrolling, one bar at a time
HOME	Moves the chart to the earliest position in time axis
END	Moves the chart to the latest position in time axis
Space bar	Opens an editor to edit the selected object
ESC	Deselects all objects
F4	Opens/Closes the tool bar window
F6	Opens data view window
F11	Repairs data series
TAB	Changes to different cursor and crosshair combinations. If a drawing tool is still active, the pointer cursor will be selected

Time Frame Manager

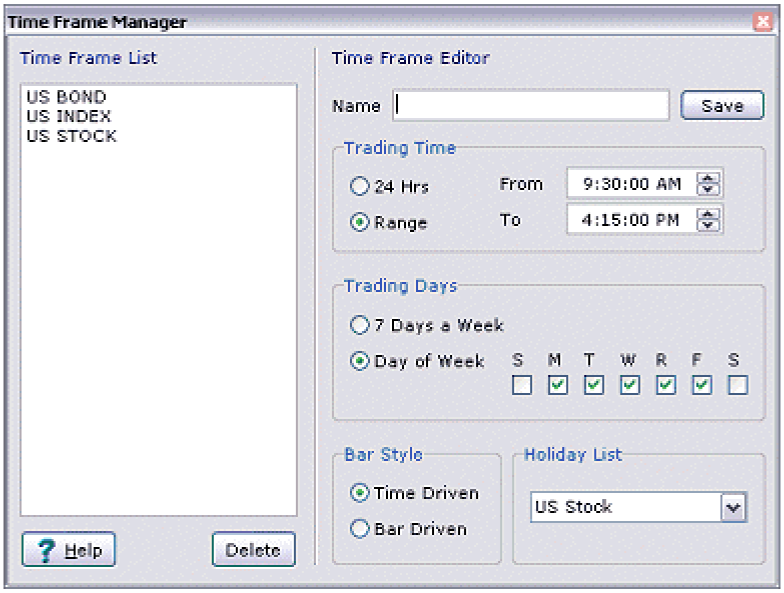
To define the trading time and holiday information for use in Time Chart, Quote Window, Volume Distribution, or DDE Link, usually you will need to specify a long list of parameters. By defining time frame beforehand with an abstract name, and refer to this name when you need to fill in these time frame parameters will save you time and avoid unnecessary mistakes.

The Time Frame Manager allows you to define your set of time frame settings and save them with a name that you can remember. For example, a pre-defined time frame can be used in a DDE link without the messy issue of providing all the different settings into the link formula.

Access the Time Frame Manager

To open the Time Frame Manager, choose from the main menu **Manager>Time Frame.**

Time Frame Manager will show up.



Time Frame Manager

Time Frame List

US BOND
US INDEX
US STOCK

Time Frame Editor

Name

Trading Time

○ 24 Hrs   From 9:30:00 AM

● Range   To 4:15:00 PM

Trading Days

○ 7 Days a Week

● Day of Week

S	M	T	W	R	F	S
□	☑	☑	☑	☑	☑	□

Bar Style

● Time Driven

○ Bar Driven

Holiday List: US Stock

The left side of the window is a list of currently defined time frames. When you add or remove a time frame setting that will be reflected in this list immediately.

The right side of the window is the time frame editor. You can change or assign the settings here until you are satisfied and save the changes using the Save Button.

Add a New Time Frame

To add a new time frame, simply put your time frame choices into the editor area. Then give a name to the new settings. Press the **Save** button to save it. The new time frame will have its name showing in the list.

To add a new time frame based on an existing one, select an existing time frame from the list first. The settings of this time frame will be showing in the editor area. Make your modification now and assign a new name to the time frame. Press the **Save** button to save this new time frame.

Edit an Existing Time Frame

To edit an existing time frame, select the time frame from the list. Its settings will be showing in the editor area. Make your changes now and press the **Save** button. The time frame will have its settings updated.

Remove an Existing Time Frame

To remove an existing time frame, select the time frame item from the list on left side. Press the **Delete** button. The list will be updated to reflect the removal of the time frame you have chosen.

Using a Time Frame Reference

You can use a time frame reference in various areas of NeoTicker®.

You can refer to ***DDE Link Manager*** (see "RTD / DDE Link Manager" on page 889) section for usage examples.

Top List Operation Guide

Many Internet data vendor provides pre-scanned lists of symbols for their customers. Examples are the most active stocks, highest volume, etc. These lists are referred to as top list (Quote.com) and hot list (myTrack) by data vendors.

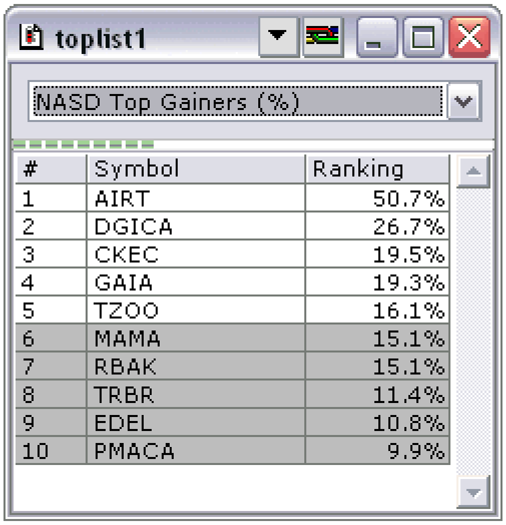
NeoTicker® provides the top list window to handle these pre-scanned lists from data vendors.

Creating a Top List Window

To create a top list, either:

- Press the top list button , or
- Choose **Window>New>Top List** in the main window.

In the top list window, choose the desired list from the drop down. The top list is updated around every 10 seconds. So it may take a few seconds before the list shows up. When updating, the top list window shows a blue status line.



NASD Top Gainers (%)

#	Symbol	Ranking
1	AIRT	50.7%
2	DGICA	26.7%
3	CKEC	19.5%
4	GAIA	19.3%
5	TZOO	16.1%
6	MAMA	15.1%
7	RBAK	15.1%
8	TRBR	11.4%
9	EDEL	10.8%
10	PMACA	9.9%

You can can send the symbols to other function window for analysis, scanning and charting, etc. For example, you can send the list to a quote window for monitoring.

1. Right click on Top List to open pop up menu.
2. Choose **Send List To** and select the window to send the symbols to.

After Hours Issues

Sometimes data vendors update their pre-scanned list much slower after hours than during trading hours. If you use the top list after hours, it may take more than a few seconds before the symbols show up.

Tool Menu

In main window, you can access the auxiliary tools of NeoTicker®. For example, you can choose **Tool>Scan Workshop** to open Scan Workshop, a scanning tool.

Customizing Tool Menu

To modify entries in **Tool** menu:

1. In main window, choose **Tool>Option** to open Tool Menu Option dialog.
2. In the dialog, check the items you want to display under **Tool** menu.
3. Some tools are installed separately from NeoTicker®, you may need to press the ... button to specify the location of the tool.

Defining Your Own Tool

To create an entry under **Tool** menu for your own programs.

1. In main window, choose **Tool>Option** to open Tool Menu Option dialog.
2. Choose a Name.
3. Enter the location of the program under App.
4. If the program requires any addition parameters, enter it under Param.
5. Press Add button.

Use **Replace** button and **Delete** button to replace and delete entries.

Trade Simulator Operation Guide

This section has been moved to ***Trade Simulator*** (on page 755).

Trading Time and Holidays

Trading Time

You can specify trading time for all windows that performs indicator calculation. Trading time is important for proper charting and indicator calculations for different types of instruments.

Another use for trading time is for excluding certain trading time. For instance, you can exclude Monday and Friday in a chart if you deem on Monday and Friday, an instrument behaves differently than on Tuesday, Wednesday and Thursday.

Holidays

Holidays pose a challenge for many traders because when there is a holiday, a gap is left on the charts. Traders who use trading styles that rely on a continuous chart cannot analyze properly because of the gaps.

The gap problem is not just visual. For indicator writers and system writers, the gaps contain invalid bars. This makes writing scripts more complex as the invalid bars must be handled.

NeoTicker® has an advanced holiday list management system that supports:

- Partial holidays such as early closing
- Multiple holiday lists for different markets such as stocks and commodities
- Locale specific holiday list. For example, a German trader can specify US holidays when he/she trades US securities

A default holiday list is used for all windows that support indicator calculation (chart, quote window, dynamic grid). Individual window can override the default with its own holiday list suitable for a particular purpose.

Setting a Default Holiday List

You can specify a default holiday list. This is the holiday list used by default during indicator calculation. Default holiday list affect a wide range of windows such as quote window, charts, time and sales window, cluster, etc.

1. Choose **Program>User Preference** from main window.
2. Change the default holiday list under the **General** tab.
3. Close the user preference window.

Saved windows and templates have a holiday list (or no holiday list) associated with it. This holiday list will not change even if you have changed the default holiday list. Examples are charts saved before version 3. Even if such chart does not have a holiday list assigned to it, the default holiday list is not automatically assigned to this chart.

Specifying Trading Time and Holiday List in Time Chart

See ***Trading Time and Holiday List Settings*** (on page 1086).

Specifying Trading Time and Holiday List in Pattern Scanner

In pattern scanners, the holiday list and trading time affects indicator calculations. The holiday list and trading time are applied to all symbols being scanned.

To specify a holiday list that is used only in a particular pattern scanner:

1. Open the set up window for the pattern scanner.
2. Press the **Data** tab.
3. Specify the holiday list and trading time.
4. Use one of the apply buttons to make the changes.

Specifying Trading Time and Holiday List in Quote Window

In quote windows, holiday list and trading time affect formula calculations that uses indicators.

1. Open the setup window of the quote window.
2. Press the **Data Settings for Formulas** tab.
3. Set the trading time and holiday list.
4. Press the **Apply** button.

You can use **Quick Fill** to fill in the trading time and holiday. The settings for **Quick Fill** is defined by time frame manager. Refer to ***Time Frame Manager*** (on page 1225) for more information.

Specifying Trading Time and Holiday in Dynamic Grid

In dynamic grid (such as the summary panel in time and sales window), holiday list and trading time are not set directly. Instead, trading time and holidays are implicit in time frames.

Time frames are defined using ***Time Frame Manager*** (on page 1235).

Setting Time Frame in Dynamic Grid

1. Right click on the dynamic grid and choose **Grid Setup** to open grid setup. To set up individual cell, choose **Cell Setup**.
2. Press **Data Setup**.
3. Choose a time frame under **Time Frame**.
4. Press the **OK** button.

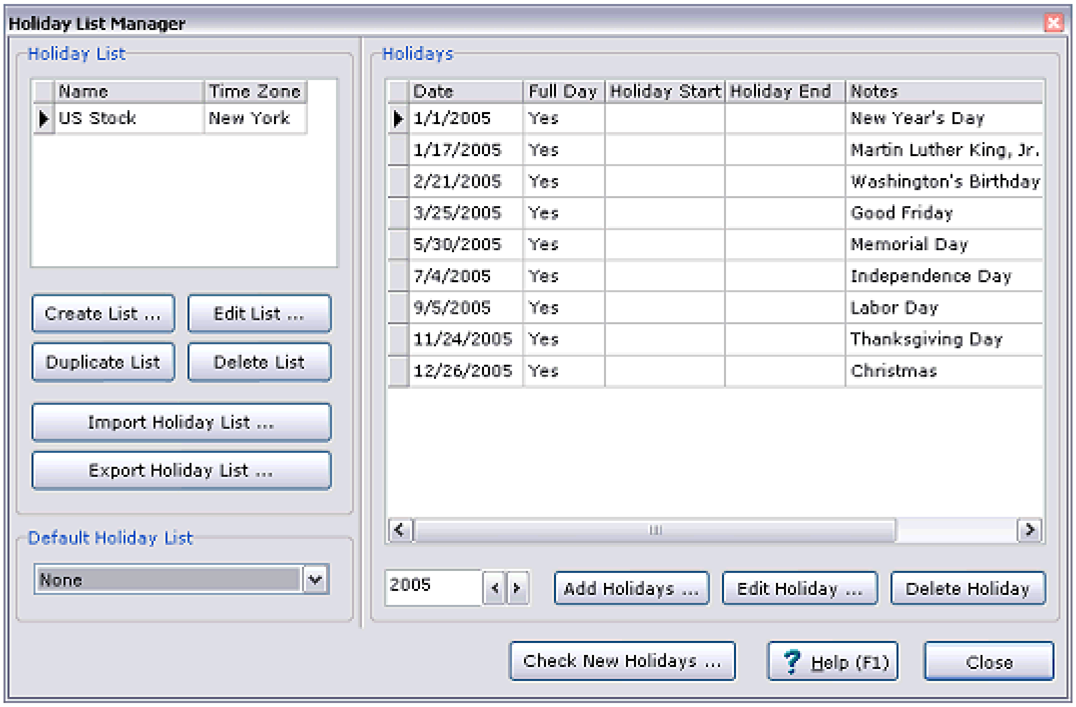
Time Frame Manager

Time frame manager provides a way to store pre-defined trading time and holidays and lets you recall them quickly. For example, you can use a pre-defined time frame to quickly fill the trading time and holidays in a quote window.

For more information about time frame manager, go to ***Time Frame Manager*** (on page 1225).

Holiday List Manager

You can edit holidays yourself using the holiday list manager. To open the holiday list manager, choose **Manager>Holiday List** from the main window.



Name	Time Zone
US Stock	New York

Default Holiday List: None

2005

Date	Full Day	Holiday Start	Holiday End	Notes
1/1/2005	Yes			New Year's Day
1/17/2005	Yes			Martin Luther King, Jr.
2/21/2005	Yes			Washington's Birthday
3/25/2005	Yes			Good Friday
5/30/2005	Yes			Memorial Day
7/4/2005	Yes			Independence Day
9/5/2005	Yes			Labor Day
11/24/2005	Yes			Thanksgiving Day
12/26/2005	Yes			Christmas

On the left hand side is the holiday lists. When you choose a specific holiday list, the holidays are displayed on the right, organized by year.

You can think of holiday list as the holidays for particular exchange. For example, the US Stock holiday list is the list of holidays when US stocks are not trading. Similarly, you can have a US Bond holiday list for days when the bond market is not trading.

Holiday List

To create a new holiday list,

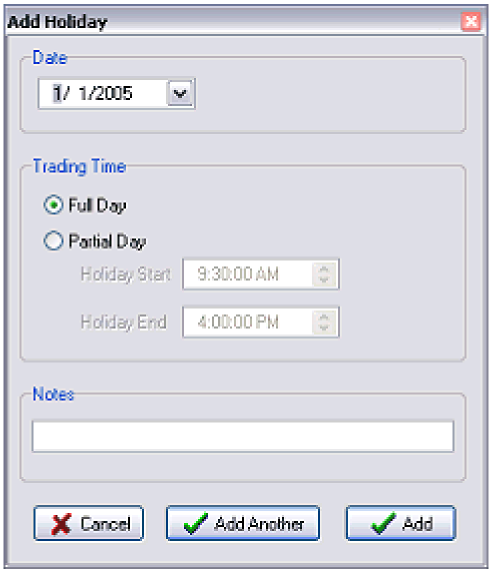
1. Press the **Create List** button. A dialog will open.
2. You need to provide a unique name and provide a locale for the holiday list.
3. If you choose **No Locale**, the holiday list will assume the locale of the target chart.
4. Press **OK** button.

If you need to edit the settings of a holiday list, click on list name to select it, then press the **Edit List** button.

To delete a holiday list, press the **Delete List** button.

Holidays

To create a new holiday day, first press the holiday list where you want the holiday to belong to. Then press the **Add Holidays** button on the holiday list manager to create a new holiday. A dialog is open to let you add holiday.


Add Holiday

Date: 1/ 1/2005

Trading Time

● Full Day

○ Partial Day

Holiday Start 9:30:00 AM

Holiday End 4:00:00 PM

Notes

A holiday is defined by the following parameters:

Parameter	Meaning
Date	The date of the holiday
Full Day / Partial Day	Whether the holiday is a full day holiday or a partial trading day.
Holiday Start	For partial trading day, time when holiday kicks in (i.e. when trading stop).
Holiday End	For partial trading day, time when holiday ends (i.e. time when trading resumes)
notes	Optional parameter for you to take notes

For example, 2003/11/28 is a partial trading day in the US. The trading time for futures market is 9:30am to 1:15pm. It has only one session, trading does not resume after 1:15pm. To specify holiday for this day:

- **Full Day** is off
- **Holiday Start** set to 1:15pm
- **Holiday End** set to end of trading day, you can use 11:59pm

To change settings of a holiday, click on the holiday to select it and press the **Edit Holiday** button.

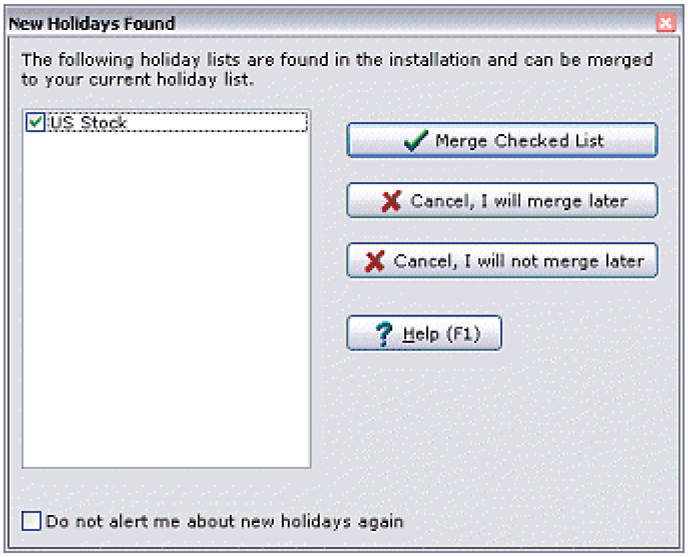
To delete a holiday from a holiday list, click on the holiday to select it and press the **Delete Holiday** button.

Importing/Exporting Holiday List

You can export holiday list to a text file by pressing the **Export Holiday List** button. Later, you can import the holidays from the text file by pressing the **Import Holiday List** button.

Discovering New Holidays

A new release of NeoTicker® may contain holiday list updates. If NeoTicker® found new holidays during startup, it will prompt you a dialog, asking you how to merge the new holidays with you current holiday list.



New Holidays Found

The following holiday lists are found in the installation and can be merged to your current holiday list.

☑ US Stock

□ Do not alert me about new holidays again

Merge Checked List

This option merges the new holidays with your current holiday list. If there is a collision, you will be prompted on how the merge should proceed.

Cancel, I will merge later

This option will cancel the merge. The merge dialog will be open again when you restart NeoTicker®, or if you press the **Check New Holidays** button in Holiday List Manager.

Cancel, I will not merge later

This option will cancel the merge. The new holidays will be removed from the installation and the merge dialog will not show up when you restart.

Trading System Live Deployment Guide

This section has been moved. See ***Trading System Life Deployment Guide*** (see "Trading System Live Deployment Guide" on page 785).

Trend Explorer ES Indicator

Introduction

Trend Explorer ES (TEES) indicator is a TickQuest proprietary indicator that shows statistical bias in long term testing. TEES is designed specifically for S&P 500 e-mini contract at 1-minute time frame.

TEES is not available in the demo version. TEES is available only in lease or permanent versions of NeoTicker®.

For information regarding rigorous long term testing of TEES, please refer to TEES's own website: ***www.trendexplorer.com*** http://www.trendexplorer.com.

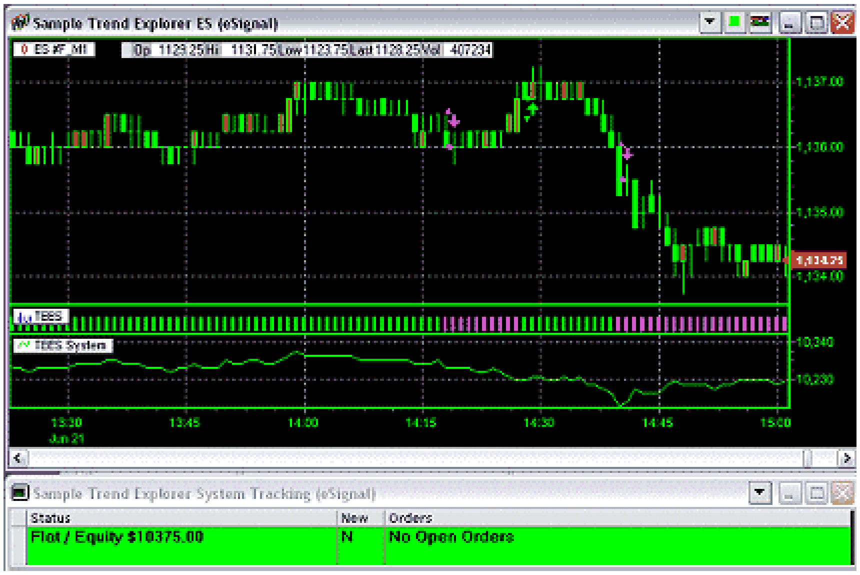
This chapter tells you how to use TEES in NeoTicker®.

Using a Prebuilt Group

To watch TEES in action, you can simply open one of the TEES groups. Choose from main window, **Group>Open Prebuilt Group**. The following groups are available for different data feeds:

- Demo Trend Explorer ES (NY Time, eSignal)
- Demo Trend Explorer ES (NY Time, IDS-DTN)
- Demo Trend Explorer ES (NY Time, myTrack)
- Demo Trend Explorer ES (NY Time, NAQ)
- Demo Trend Explorer ES (NY Time, QCharts)

These group will open a chart that has a trading system built around TEES and a system monitor window to alert you about the trades.


Sample Trend Explorer ES (eSignal)
0 ES #F_MI
Op 1123.25 Hi 1131.75 Low 1123.75 Last 1128.25 Vol 407234
1,137.00
1,136.00
1,135.00
1,134.25
1,134.00
TEES
TEES System
10,340
10,230
13:30
Jun 21
13:45
14:00
14:15
14:30
14:45
15:00
Sample Trend Explorer System Tracking (eSignal)

Status	New	Orders
Flat / Equity $10375.00	N	No Open Orders

The chart is based on NY Time, i.e. the trading time is from 9:30am to 4:15pm. If your computer or NeoTicker® is set to other time zone, you will need to adjust the chart's time to S&P 500 e-mini's trading time.

Contract Rollover

If your data vendor does not support continuous contract, you must manually roll over to the new contract in the TEES group on first notice day. Because TEES requires at least 20 days of minute data to work, you must also copy the data files from one contract to another. You can use the following procedure:

1 Open the TEES group, change the symbol of the chart to the next contract. Suppose the the June contract of 2003 is about to expire, you will want to change the symbol to the September contract. For example, you will want to change the symbol from `ES3M` to `ES3U`. The symbols are data vendor dependent.

2 Save the group.

3 Exit NeoTicker®.

4 In Windows, open the disk cache directory, e.g. `C:\Program Files\TickQuest\NeoTicker3\cache`.

5 Go into the minute data directory of the e-mini contract that just expired, e.g. `ES3M\min`

6 Select all the data files (e.g. `20030515`, etc). Choose CTRL-C to copy the files. Note that if you do this after the first notice day, make sure you select and copy only the data files before the first notice day.

7 Go into the minute data directory of the new e-mini contract, e.g. `ES3U\min`

8 Choose CTRL-V to paste the data files.

9 Re-open NeoTicker®.

10 Open the TEES group.

Using TEES Yourself

TEES is an indicator that returns -1, 0, 1 for bearish trend, neutral and bullish trend. Therefore, you can use TEES just like any other indicator. To properly use TEES, please keep the following points in mind:

- TEES is designed solely for 1-minute S&P 500 e-mini contract. While you can apply TEES to other instruments, the result can be very poor.
- TEES requires at least 20 days of 1-minute data to work. If you use TEES in a chart, make sure the chart and the e-mini data series has enough data. If you use TEES in quote window formula, make sure you allocate enough bars in RAM Cache in Cache Manager. In both cases, make sure the trading time is for e-mini trading.
- If you feed TEES to a trading system, make sure you set the trading system parameters for trading e-mini contracts. Also if you plan to follow the trading system with system monitor, you should enable system monitor option in the trading system.

User Define Symbol Manager

You can define special symbols that perform customized calculations based on your criteria. These symbols are known as user define symbols. To create these symbols, use the User Define Symbol Manager.

Common usages of user define symbols include customized portfolio indices, spreads, and special data resulted from computation by another program.

How does User Define Symbol Work

A user define symbol has a symbol name and a evaluation type associated with it. Evaluation type is the way the user define symbol is evaluated. Examples include formula, weight sum, etc.

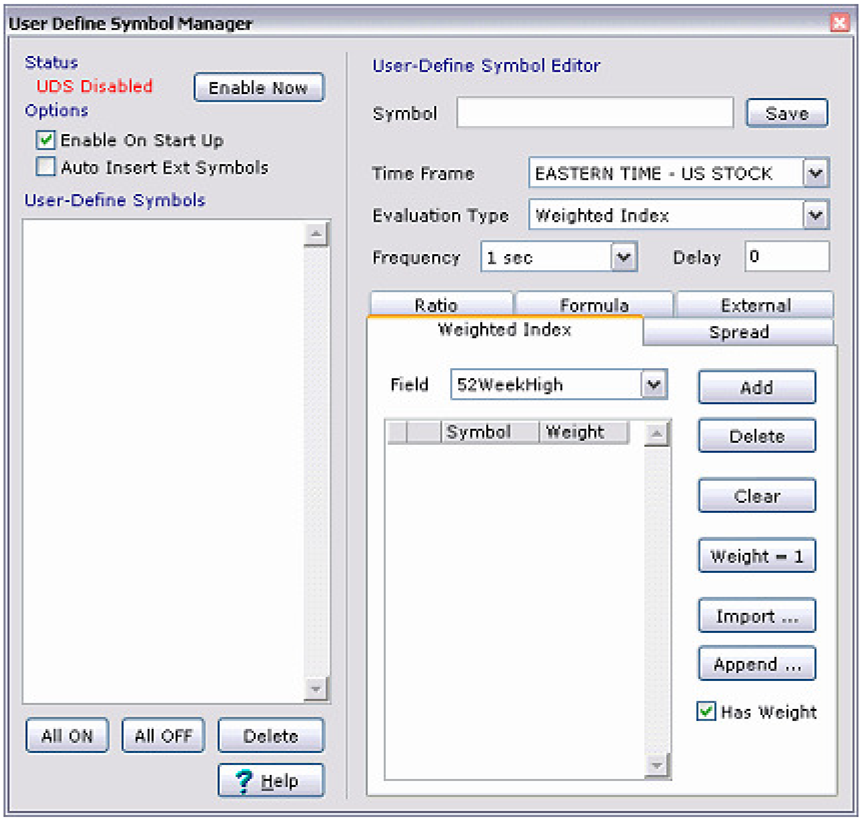
At a specific interval, the symbol is evaluated using the evaluation type. The value becomes the symbol's current value.

User define symbols can be used anywhere in NeoTicker®. For example, you can quote, chart the symbol or even set up a DDE link to Excel.

Opening User Define Symbol Manager

To open User Define Symbol Manager, choose from the main program window, **Manager>User Define Symbol**.

The manager will show up.



User Define Symbol Manager

Status

UDS Disabled

Options

☑ Enable On Start Up

□ Auto Insert Ext Symbols

User-Define Symbol Editor

Symbol

Time Frame: EASTERN TIME - US STOCK

Evaluation Type: Weighted Index

Frequency: 1 sec

Delay: 0

Field: 52WeekHigh

☑ Has Weight

The upper left hand corner of the window is the global options for controlling the behaviour of the user-defined symbols.

Left side of the window is a list of the user-defined symbols currently saved.

Right side of the window is the editor area for editing the definition of a user define symbol.

Adding a New User Define Symbol

To add a new user define symbol, enter a new symbol name at the **Symbol** edit area and fill in the details below. Once you are done entering the information, press the **Save** button to add the new symbol.

After pressing the **Save** button, user define symbol manager will ask if you want to start tracking the symbol now. If you want to start the tracking at once, press the **Yes** button. If you do not want to start the tracking now, say, you would like to add more user define symbols first, then you can press the **No** button to save the settings without tracking enabled. If you want to give up the saving completely, press the **Cancel** button.

Editing an User Define Symbol

When you click on a symbol in the list, that particular user define symbol will have its settings loaded into the editor area. You can make your changes and then press the **Save** button to modify the existing definition of the user define symbol.

Removing an User Define Symbol

To remove an existing user define symbol, you can select the symbol from the list, then press the **Delete** button located at the bottom of the list.

Enabling/Disabling an User Define Symbol

A user define symbol is enabled if it has a check mark next to its name in the list area. You can click on this check mark to toggle the enable / disable state of any user define symbol.

You can also use the **All ON** button to turn on all the user define symbols for data collection. Or you can use the **All OFF** button to turn off the entire list of user define symbols at the same time.

Enabling/Disabling User Define Manager

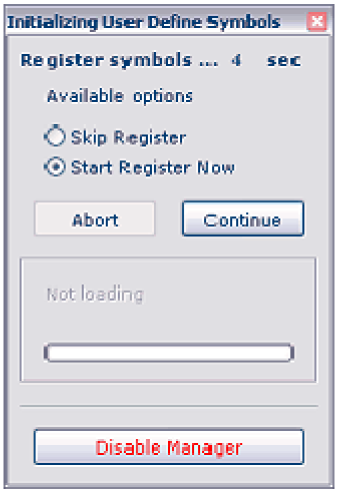
You can completely disable the User Define Manager by turning off the option **Enable Manager** located at the upper left corner of the window. Doing so will stop all real-time updates of any user define symbol at once.

To reactivate the User Define Symbol Manager just check the **Enable Manager** option.

This option is useful when you have defined many user define symbols but you do not want them to collect data in real-time for whatever reason.

Interrupting UDS on Startup

When starting up with a real-time feed, you have a choice to interrupt User Define Symbol initialization. The following dialog is opened automatically:



Register symbols ... 4 sec

Available options

○ Skip Register

● Start Register Now

Not loading

Actions you can take:

- Choose **Skip Register**, then press **Continue** button - Symbols required to generate User Define Symbols will not be registered (i.e. requested from your data vendor). User Define Symbols will still be generated, but their values will take longer to stabilize.
- Choose **Start Register Now**, then press **Continue** button - Symbols required to generate User Define Symbols will be registered (i.e. requested from your data vendor). This is the default.
- Press **Disable Manager** button - User Define Symbol Manager will be disabled and symbols will not be registered.

Auto Insert External Symbols Option

Another program can insert user define symbols into NeoTicker® as an external type user define symbol. The insert mechanism is done through NeoTicker®'s ActiveX automation interface.

The ability to insert from another program is off by default. To activate the ability to auto insert new external type symbols, just check the **Auto Insert Ext Symbols** option.

If you do not want to allow auto insert of new symbols of external type into NeoTicker® by other applications, you can turn off the **Auto Insert Ext Symbols** option.

When you find that the application program that sends burst of data to NeoTicker® in an erratic way, you can stop NeoTicker® from receiving data from this external program immediately by turning off this option.

Settings

Common Settings

Symbol

The unique identification of the user define symbol you have created. Make sure this symbol does not share its symbol with any regular symbols that you collect data from your data vendor. For example, a user define symbol of `MSFT` will likely to collide with most data vendors' symbol for Microsoft. Try using something like `MY_MSFT` will ensure that it is unique.

Time Frame

You can choose a pre-defined time frame from the list. Based on the time frame, the user define symbol will retain ticks within the given time range and weekdays specified by the time frame you have chosen. To create customized Time Frame settings, you can use the Time Frame Manager. See ***Time Frame Manager*** (on page 1225) for more information.

Evaluation Type

You can choose from one of the evaluation types in the list. After you have decided on your selection, remember to complete the setting in the corresponding tab sheet.

Frequency

This parameter determines how frequently the user define symbol will generate a tick. If you find your computer cannot handle the CPU load caused by frequent evaluation of a user define symbol, you can lower the frequency.

Delay

If the data is from a delayed source, enter the delay parameter here (in number of minutes). The tick generated will have a delay time stamp, matching that of the source data.

Weighted Index Type

The Weighted Index type user define symbol calculates its value based on three parameters - a list of symbols, the corresponding weights for each symbol, and the quote field that you want to take total of.

For example, you are interested in calculating the S&P 500 Advance Issues, then you can create a user define symbol named `SP500ADV`. Then you can import the list of 500 symbols using the **Import** button. By default, the weight is 1 for all symbols, which is sufficient in this case. Finally, we can specify the quote field to **UpFromClose**.

Since **UpFromClose** returns 1 for symbols that are trading above their previous close and 0 otherwise, by taking the sum of **UpFromClose** for all 500 symbols of S&P 500, we can obtain the number of advance issues.

The newly created `SP500ADV` symbol can be used in real-time just like other regular symbols. You can use it in quote window, in chart, and even apply indicators to it.

Weighted Index Type, Symbol List

By checking **Symbol List**, you can specified a symbol list file for weighted index. This is useful if you have a symbol list that is frequently changing. Symbol list is reset when the user defined symbol is disabled and then enabled.

Weight index symbol list recognizes a weight parameter in the list. For example, for the symbol list:

```
MSFT, 2
CSCO, 1
INTC
```

The symbol MSFT has a weight of 2, CSCO has a weight of 1, INTC has a weight of 1 (by default).

Weighted Index Type, Customized

By checking **Customized**, you can manually add symbols to the symbol listing area and directly adjust the weights by manual entry. If you are only interested in even weights, you can do so by pressing the **Weight = 1** button which will then reset all the weights to 1.

To add a new symbol to the symbol listing manually, use the **Add** button.

To remove a symbol from the symbol listing manually, use the **Delete** button.

You can clear all the symbols and their weights by pressing the **Clear** button.

The **Import** button let you import an existing symbol list and completely replace the existing symbols in the symbol listing area.

The **Append** button allows you to import a symbol list and merge that with the existing ones in the symbol listing area.

Has Weight. If this check box is checked, when importing/appending symbols, a comma separated weight parameter is recognized. For example, for the symbol list:

```
MSFT, 2
CSCO, 1
INTC
```

The symbol MSFT has a weight of 2, CSCO has a weight of 1, INTC has a weight of 1 (by default).

Spread Type

The Spread type user define symbol calculates the spread between 2 symbols with user specified weighting for each.

The default evaluation of spreads is to take the difference between 2 symbols. Thus when you enter the weights for both symbols with positive values, user define symbol manager will generate the spread like the formula below:

```
Latest Price of Symbol A x Weight of Symbol A – Latest Price of
Symbol B x Weight of Symbol B
```

When you enter a negative value for the weight of `Symbol B`, you will be effectively adding the weighted value of `Symbol B` to the weighted value of `Symbol A`.

If you select **Update by Tick**, Spread will be calculated when a tick arrive from either symbol and update timer will be ignored. Also, when updating by tick, each tick of the user define symbol will have a volume that is the sum of volume of the two symbols.

Ratio Type

The Ratio type user define symbol calculates the ratio between 2 symbols with user specified weighting for each.

The default evaluation of ratio is to divide the weighted value of `Symbol A` by the weighted value of `Symbol B`. The ratio can be expressed as a formula like the one below:

```
(Latest Price of Symbol A x Weight of Symbol A)  / (Latest
Price of Symbol B x Weight of Symbol B)
```

If you select Update by Tick**, Ratio will be calculated when a tick arrive from either symbol and update timer will be ignored. Also, when updating by tick, each tick of the user define symbol will have a volume that is the sum of volume of the two symbols.**

Formula Type

The Formula type user define symbol calculates the update value based on the user specified formula and base symbol.

The formula that you enter to the formula entry area is exactly the same type of formula you use in a quote window formula column. You can specify quote fields, indicators, etc.

To reference to other symbols within the same formula, use the square bracket syntax, e.g. if base symbol is `MSFT`, you can use the formula `[IBM]Last` to get IBM's last price.

External Type

The External type user define symbol is updated by another application that has access to NeoTicker® 's ActiveX interface.

If you specify **Update By Tick**, user define symbol manager will generate a tick for each update received from other application. Otherwise, symbol is updated on a fixed time interval basis similar to other type of user define symbols.

For more details on how to update an external type symbol, please refer to ***OLE Automation*** (on page 653).

Usage Considerations

When an user define symbol depends on other actual symbols in its computation, those actual symbols will have to be connected for real-time update with your data vendor. If the number of symbols being used has exceeded the symbol limit imposed by your data vendor, NeoTicker® will no longer be able to collect data in real-time consistently.

When you exceeded your symbol limit imposed by your data vendor, the basic way to re-establish connection is to first go into off-line mode. Then turn off the offending user define symbols. Finally, choose from the main program window, **Program>Server Connect/Disconnect** to connect to your data vendor again.

For some data vendor, you will have to exit both NeoTicker® and their server software first. If that is the case, when you start NeoTicker® again, press the SHIFT key to force NeoTicker® into off-line mode. Then disable the offending user define symbols, before connecting to the data vendor.

Volume Distribution Charts Operation Guide

Volume distribution chart utilizes NeoTicker®'s advance tick-by-tick processing technology to give a visual representation of the trading volume distribution according to price.

For example if MSFT is traded near $63 for a total of 500,000 shares, a 500,000 shares horizontal bar will be plotted on the volume distribution chart of MSFT. By looking at all the horizontal bars, you can gain insights on how much volume MSFT is traded at each price.

Creating Volume Distribution Chart

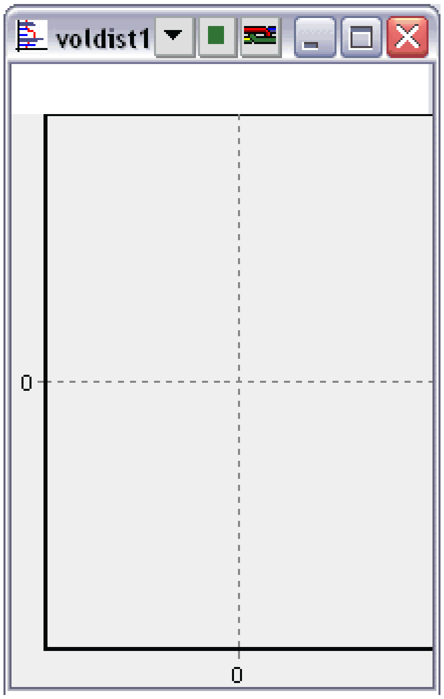
To create a volume distribution chart, either:

- Press the volume distribution chart button , or
- Choose **Window>New>Vol Dist Chart** from the main window.

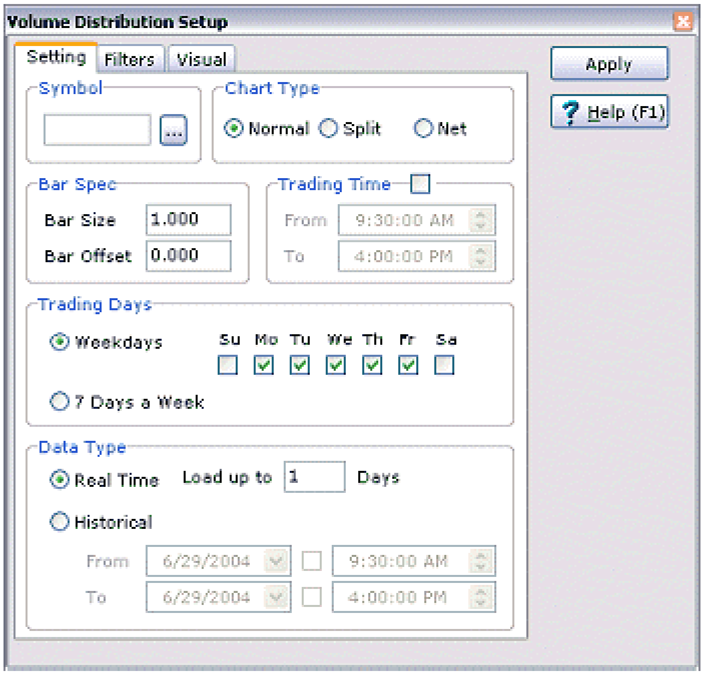
A volume distribution chart will be created. Initially, a volume distribution chart does not have any symbol in it. So you need to provide it a symbol.

Setting up Volume Distribution Chart

A new volume distribution chart is blank.



To set up the volume distribution chart, right click on the chart to bring out the pop up menu, and choose **Setup** to open the setup window.



Volume Distribution Setup

Setting | Filters | Visual

Apply

Help (F1)

Symbol

Chart Type: ● Normal ○ Split ○ Net

Bar Spec
Bar Size 1.000
Bar Offset 0.000

Trading Time □
From 9:30:00 AM
To 4:00:00 PM

Trading Days
● Weekdays
Su □ Mo ☑ Tu ☑ We ☑ Th ☑ Fr ☑ Sa □
○ 7 Days a Week

Data Type
● Real Time Load up to 1 Days
○ Historical
From 6/29/2004 □ 9:30:00 AM
To 6/29/2004 □ 4:00:00 PM

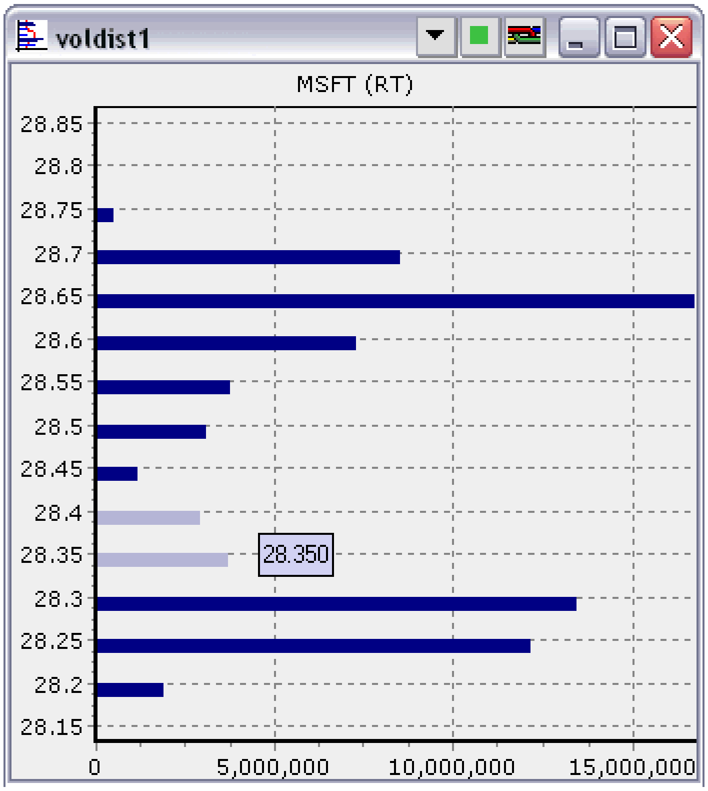
Here are the items you need to fill in:

- A symbol (e.g. MSFT) to the symbol field.
- A bar size. This is the range of value that makes up a bar in the volume distribution chart. For low price security such as stocks, you will probably enter something like 0.1. For index, you will probably use 1 or even 10.

Press the **Apply** button. Volume distribution chart will start calculating the volume distribution for the symbol. Volume distribution calculation is quite time consuming (because data is analyze on tick level), so volume distribution chart will show you a 'Loading' message on the caption of the volume distribution chart window until the chart is ready.

Loading data can be slow for data feeds that do not provide a fast database of tick data. Utilizing the pre-load symbol list can help the speed a lot for these feed as the ticks would be available in disk cache.

Below is an example of a volume distribution chart.



Settings

Here is an explanation of the settings:

Setting	Meaning

Symbol Enter the symbol here

Chart Type There are 3 types of volume distribution. Normal refers to the total tick-by-tick volume. Split refers to the splitting of the volume based on up/down tick. Net refers to the net difference between up volume and down volume.

Bar Size Controls the range of values that lump into a single horizontal bar. For stocks, a bar size of 1 indicates $1, so each dollar will become a horizontal bar in the chart

Bar Offset Controls how much offset in the bar to consider a trading price within the bar. For example, a trading price P is in the $70 bar when ($69.5 + offset) < = P < ($70.5 + offset), assuming a bar size of 1

Trading Time You can restrict the trading time. Ticks that have a time stamp marked outside of the trading time are not included in the volume distribution chart

Trading Days You can restrict the trading days. Ticks that have a time stamp marked outside of the trading days are not included in the volume distribution chart

Real Time Data Type Specifies a real-time chart. real-time chart accumulates the trading volume for a specified number of days and updates tick by tick in real-time

Historical Data Type Specifies a historical chart. Historical chart plots the volume distribution for a specific date range and does not update in real-time

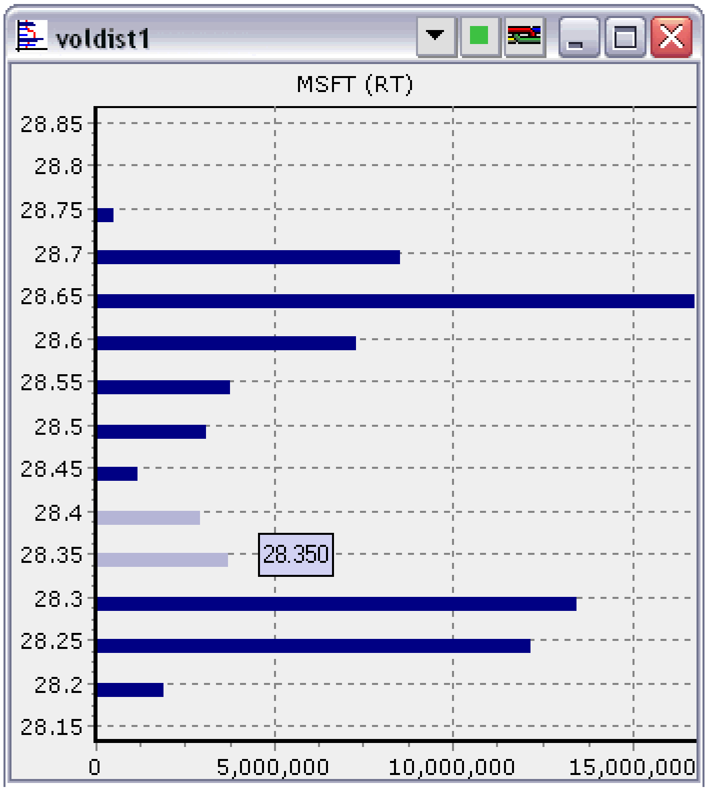
Use Last Time Specify a cut off time for historical chart on the last day. This feature is useful to replay how the volume distribution changes in time. For example, you can set Use Last Time to 3 PM to display the volume distribution up to 3 PM. Then change Use Last Time to 4 PM and observe how the volume distribution changes in an hour

Apply button Press the **Apply** button to finish Setup. Volume distribution chart calculation is time consuming. So after you have pressed the **Apply** button, NeoTicker® will display a Not Ready message in the caption of the volume distribution window until the calculation is finished

Reading Volume Distribution Chart

Volume Distribution Chart Window - Normal Type

Consider the following volume distribution chart example:



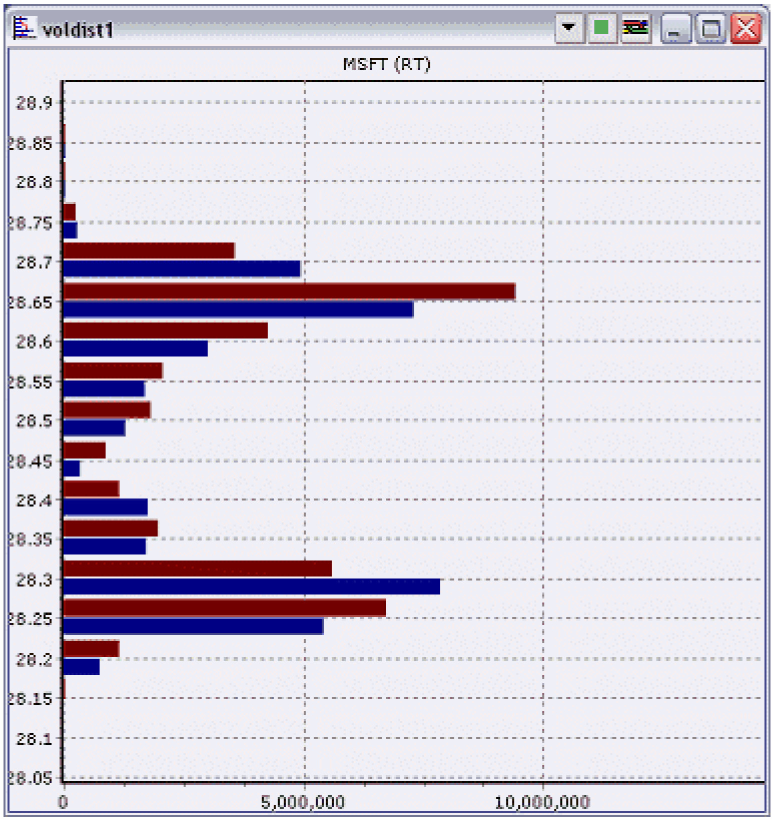
In the example chart, you will see that the traded volume for MSFT is plotted for different prices. From the chart, you can see that MSFT is traded mostly at around $28.65.

You may notice that the $28.35 and $28.4 bars are in bright blue color while other bars are in dark blue. Volume distribution implements a black body-style coloring scheme. When a trade is happening at a price, the bar will become a little bit brighter. So when a lot of trades are happening at a price range, the bar for that price will become very bright. On the other hand, bars 'lose heat' and cool down to darker color over time. So price bars that haven't been traded for a long time are in dark blue. This color scheme can give you a sense how the trades are happening and how they contribute to the formation of the shape of the volume distribution chart.

If you open the volume distribution chart during trading hours, the chart will show a small marker displaying the last trade price. The marker is drawn at the end of the volume bar.

In the example, the marker is in light blue indicating an up tick (or side tick after an up tick). If the marker is in pink, it is indicating a down tick (or side tick after a down tick).

Volume Distribution Chart Window - Split Type



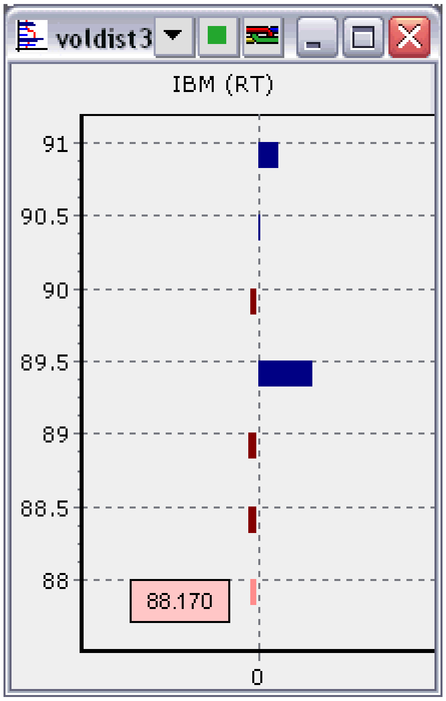
The above chart is a split type volume distribution chart. It has a setting of **Bar Size** equals to 0.1 and **Bar Offset** equals to 0.

The red color bars are the ones that cumulate volume over down ticks while the blue bars are the ones that cumulate volume over up ticks.

The red and blue bars are organized to show side by side for the same price slot. All the up tick trades within this group are cumulated in the blue bar below while the down tick trades within this group are cumulated in the bright red bar above. Similar to a normal chart, a bar that has been traded recently will be brighter in red or blue.

The reason for the red bars to be placed above blue bars is that the red bars represent a type of resistance. And for the same reason, blue bars are placed below the red bars because the blue bars represent a form of support.

Volume Distribution Chart Window - Net Type

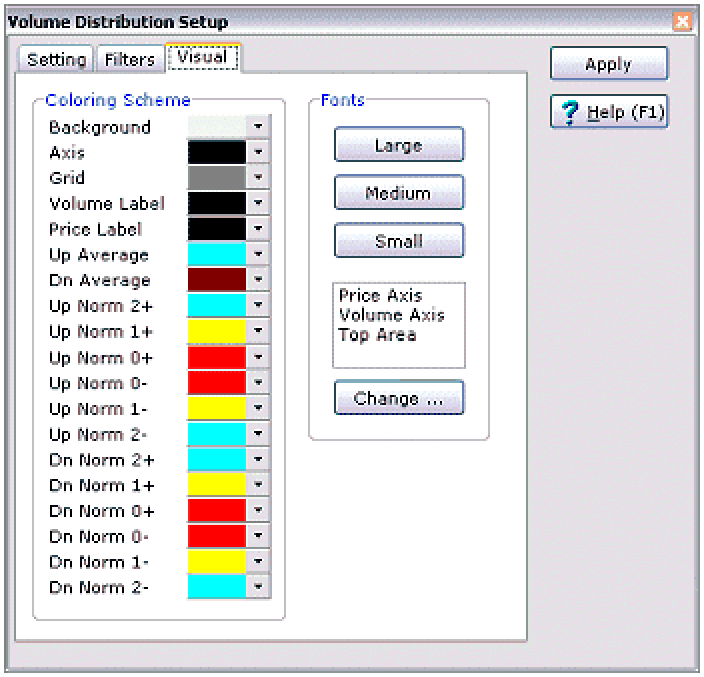


The above chart is a net type volume distribution chart. **Bar Size** is 0.5 and **Bar Offset** is 0.

The red color bars reflect a negative net trading volume at the specific price slot while the blue color bars reflect a positive net trading volume.

Changing Volume Distribution Chart Color and Font

You can change the color and font used by a volume distribution chart. The settings are under the **Visual** tab in the volume distribution setup window.



Coloring Scheme

Background	⬜
Axis	⬛
Grid	▨
Volume Label	⬛
Price Label	⬛
Up Average	🟦
Dn Average	🟫
Up Norm 2+	🟦
Up Norm 1+	🟨
Up Norm 0+	🟥
Up Norm 0-	🟥
Up Norm 1-	🟨
Up Norm 2-	🟦
Dn Norm 2+	🟦
Dn Norm 1+	🟨
Dn Norm 0+	🟥
Dn Norm 0-	🟥
Dn Norm 1-	🟨
Dn Norm 2-	🟦

Price Axis
Volume Axis
Top Area

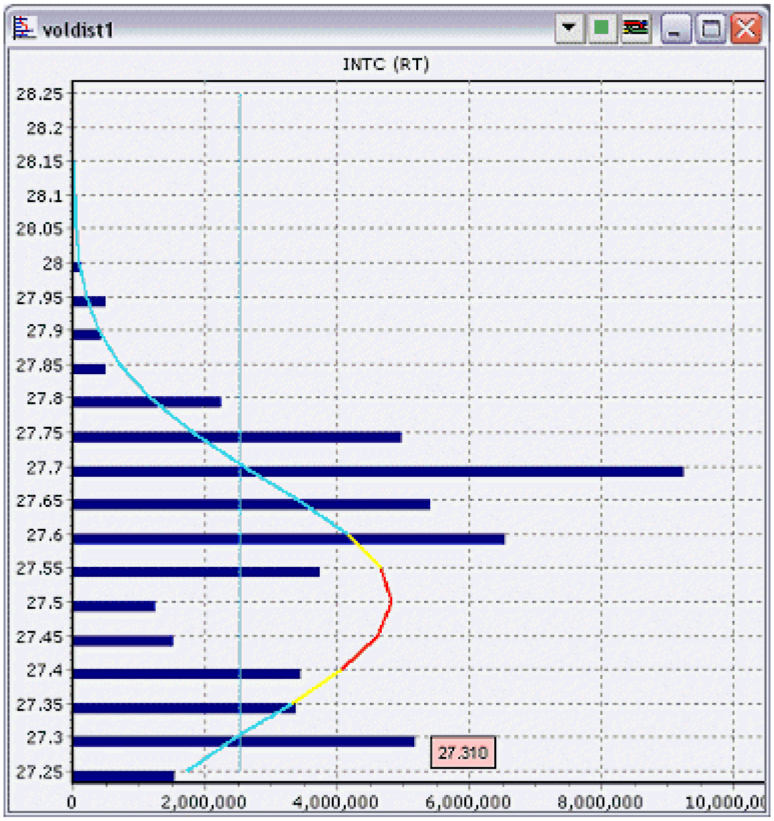
The **Large**, **Medium** and **Small** buttons let you quickly change the font size of the whole chart. You can also individually change the fonts of price axis labels, volume axis labels and the top area.

Displaying Average Volume and Normal Distribution

You can choose **Normal Distribution** from the pop up menu to fit the volume data into a normal distribution.

You can choose **Average Volume** from the pop up menu to display the average volume.

Average volume is displayed as a vertical line. The normal distribution is drawn on different colors for different standard deviations.



Filters and Uniform Weighting on Ticks

You can filter out ticks that you do not deem useful in your analysis. For example, you can filter out ticks with trade volume less than 1000 to analyze only the large trades.

To set up filtering:

1 Open the setup window for volume distribution chart.

2 Press the Filters tab.

3 Add conditions by choosing **Add** in the pop up menu inside **Volume Filter**.

4. Enter conditions. When a tick does not match the condition, it is filtered and not included in volume distribution.
5. Press **Apply** button.

Filter Setup

To set up volume filtering, choose the **Filters** tab in the volume distribution setup window.

You can add a filter by choosing **Add** from the pop up menu. Each filter specifies one or two conditions. Trades that satisfy the specified condition(s) pass through the filter and contribute to the volume distribution. For example, if you set **C1** to >=, **V1** to 1000, **C2** <= and **V2** to 2000, then trades that have a volume between 1000 to 2000 will contribute to the volume distribution chart.

Multiple lines of filters act as 'or' conditions. By specifying multiple lines of filters, you allow multiple types of trades to contribute to the volume distribution. For example, if you have two lines of filters, you set the first filter's **C1** to >= and **V1** to 1000 and the second filter's **C1** to <= and **V1** to 200, then trades with volume above 1000 or trades with volume below 200 will contribute to the volume distribution chart.

If you do not specify any filters, all trades will contribute to the volume distribution.

Uniform Weighting on Ticks

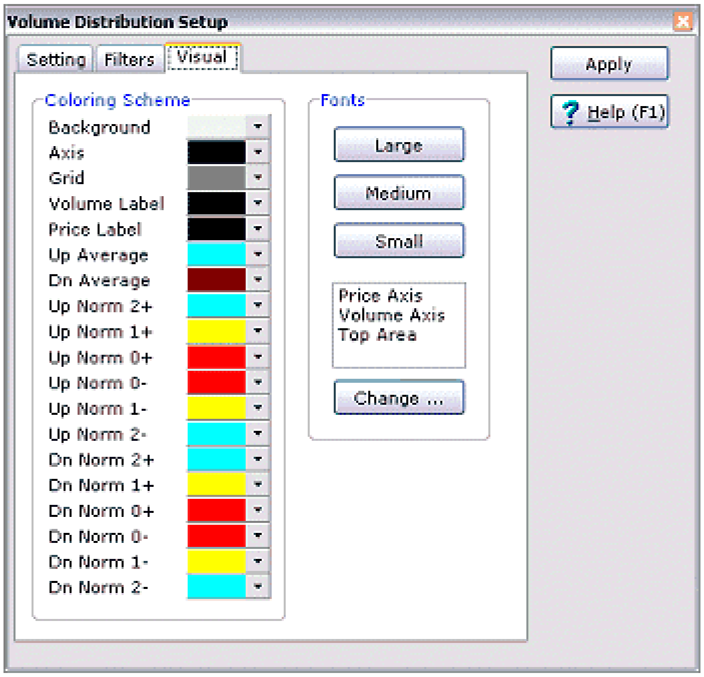
If you enable the **Uniform weighting on all ticks** option, then the volume distribution will be constructed such that each tick will be considered having a volume of 1. This provides an alternative view on volume by visualizing the number of ticks, rather than the actual volume.

Color and Font Settings

To adjust color and font settings:

1. Open the setup window of volume distribution chart.
2. Press the **Visual** tab.
3. Make changes.

4 Color and font changes are interactive. You do not need to press the **Apply** button for the changes to happen.



Here is the explanation of the visual tab items in the volume distribution setup:

Setting	Meaning
Background	Background color
Axis	Axis color
Grid	Grid color
Volume Label	Volume label color
Price Label	Price label color
Up Average	The color to use for the average line in normal chart or for the up average line in net chart and split chart
Dn Average	The color to use for the down average in net chart and split chart
Up Norm 2+	The color to use for the up normal distribution beyond 2 standard deviation
Up Norm 1+	The color to use for the up normal distribution from 1 to 2 standard deviation
Up Norm 0+	The color to use for the up normal distribution from 0 to 1 standard deviation
Up Norm 0-	The color to use for the up normal distribution from 0 to -1 standard deviation
Up Norm 1-	The color to use for the up normal distribution from -1 to -2 standard deviation
Dn Norm 2-	The color to use for the down normal distribution from -2 to -3 standard deviation
Dn Norm 2+	The color to use for the down normal distribution beyond 2 standard deviation
Dn Norm 1+	The color to use for the down normal distribution from 1 to 2 standard deviation
Dn Norm 0+	The color to use for the down normal distribution from 0 to 1 standard deviation
Dn Norm 0-	The color to use for the down normal distribution from 0 to -1 standard deviation

Dn Norm 1- The color to use for the down normal distribution from -1 to -2 standard deviation

Dn Norm 2- The color to use for the down normal distribution from -2 to -3 standard deviation

Fonts The **Large**, **Medium** and **Small** buttons change the font size of the whole chart. You can also individually change the fonts of price axis labels, volume axis labels and the top area

Crosshair

To turn on volume distribution crosshair, choose **Crosshair Mode** from the pop up menu.

Real-Time Price Markers

When volume distribution chart is working on real-time data, a real-time price marker is shown on the chart.

To turn on/off marker and change marker style, choose **Marker Style** from pop up menu.

Price Axis Scaling

By default, volume distribution chart automatically adjusts the price axis scale.

You can change the price axis scale by choosing **Fixed Scale** from the pop up menu. A dialog will be opened to let you enter the minimum and maximum for the price axis.

To turn automatic scaling back on, choose **Auto Scale** from the pop up menu.

Reloading Ticks

If you suspect there is a problem with the tick data in volume distribution chart, you can reload the tick data by choosing **Refresh** from the pop up menu.

Exporting the Image

You can take a snapshot of the image of the current volume distribution chart and export it to a file or clipboard. Choose **Export** from the pop up menu.

Shortcuts

Pressing the space bar on the volume distribution chart will open the setup window for volume distribution chart.

CHAPTER 12

Power Indicators Guide

NeoTicker® completely redefines what an indicator can do. NeoTicker®'s indicators can perform calculations that were possible previously only if you write your own charting application.

In this guide, we will present these power indicators to you. These indicators allow you to better analyze the market with unique perspectives. Power indicators also lay the foundation for the creation and utilization of better and more sophisticate indicators.

For example, you can use NeoTicker®'s weighted index indicator to calculate your personal index on multiple securities. Furthermore, the calculated index is present to you as a data series, in bars, which contains open, high, low, close values. You can simply treat your weighted index as a data. You can apply indicators such as stochastic on your personal index and expect a correct result.

More power indicators will be added over time to ensure NeoTicker® to stay as the leader in the computer assisted technical analysis arena.

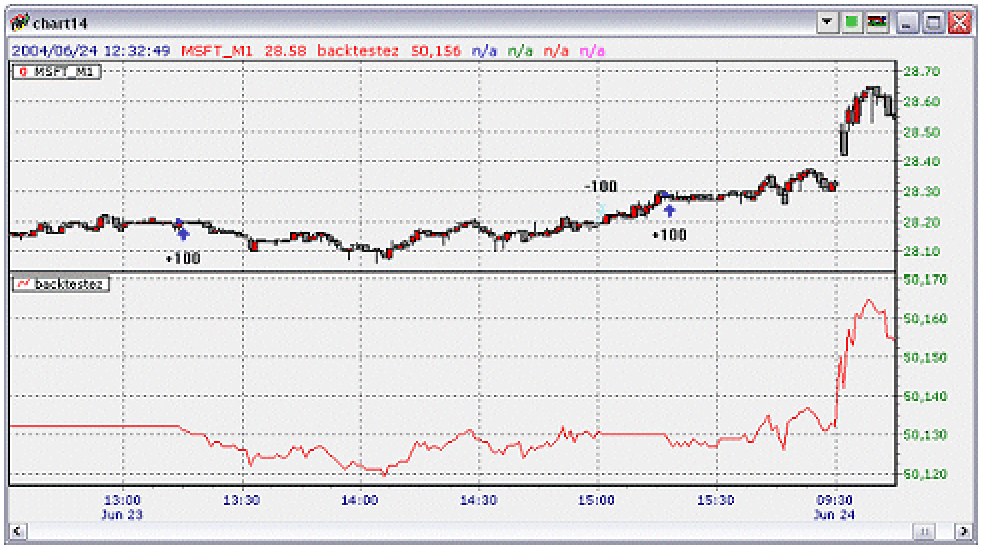
If you are an indicator writer, you will be glad to know that all of these power are available to you. All of the power indicators in this section are first developed and tested as scripts before being converted to internal indicators.

In This Chapter

Backtest EZ

Overview

Backtest EZ enables NeoTicker® users to carry out basic system testing with comprehensive set of money management options. With basic knowledge of the NeoTicker® formula language, you can easily construct trading systems to test out your ideas.



Using the Indicator

For a tutorial on how to use Backtest EZ, go to ***Tutorial: Trading System Testing with Backtest EZ*** (on page 175).

Parameters Settings

Long Entry

A formula describing the condition that you want to buy at market. If the formula returns 1, then a buy signal is triggered. The formula should return 0 otherwise. If this formula is blank, then no long position will be initiated.

Long Exit

A formula describing the condition that you want to exit from a long position. If the formula returns 1, then a sell signal is triggered. The formula should return 0 otherwise. If it is blank, the system will still exit the long position by short signals or money management rules.

Short Entry

A formula describing the condition that you want to sell at market. If the formula returns 1, then a sell signal is triggered. The formula should return 0 otherwise. If it is blank, then no short position will be initiated.

Short Exit

A formula describing the condition that you want to exit from a short position. If the formula returns 1, then a buy signal is triggered. The formula should return 0 otherwise. If it is blank, the system will still exit the short position by long signals or money management rules.

Size

The number of shares or contracts to buy or sell. When it is set to a positive integer, BackTest EZ will place order with this specific size only. If Size is set to 0, then BackTest EZ will dynamically adjust the order size to match the maximum available cash. If **Size** is set to a negative integer, then BackTest EZ will dynamically adjust the order size like the case for 0, but only at exact multiple of size; for example, if size is set to -100, then BackTest EZ will only place orders at 100, 200, etc.

Trailing Style

Trailing stops are used to protect profit in a moving trend. A trailing stop does not kick in if the trailing stop price does not produce a profit. Commission is not included when profitability of the trailing stop is calculated.

Let's use a long position as an example, if you specify a trailing stop of 5 points, a sell stop order will be placed 5 points below the current highest price (provided the stop order will produce a profit). If the price goes up, the old sell stop order will be cancelled and a new sell stop order will be placed. If the price goes down, the stop order will remain in place and if the price touches the sell stop price, a market order will be issued to sell the position.

Many brokers do not provide trailing stop orders. To use trailing stop in real life trading, you will need the discipline to cancel old orders and replace them with new ones.

These options are available - None, Points, Percent, Amount, Bars, ATR.

Trailing Style	Meaning
None	No trailing stop is applied
Points	The **Trail Value** parameter is used as is to offset from the best price (highest or lowest, depending on whether it is a long/short position) level attained so far to calculate the trailing stop price. For example, if the system is in a long position, the **Trail Value** is 1, and the highest price level is 51, then the trailing stop is at 50. If the highest price level moved to 52, then the trailing stop will move up to 51.
Percent	The **Trail Value** parameter is used as the maximum percentage of the best profit to give back. For example, you are in a long position and the best profit so far is $2 per share (contract). If the **Trail Value** is 25, the trailing stop will be 2 x 0.25 = 0.5 below the highest price level.
Amount	The **Trail Value** parameter is used as the total dollar amount to be given back from the profit. For example, you are long MSFT at $50. Then the Trail Value at 200 means that you are willing to give back $200 of the profit before your position is taken out. Thus if the position has 100 shares, your stop will be $200 / 100 = $2 from the highest price level. If the position has 200 shares, your stop will be $200 / 200 = $1 from the highest price level.
Bars	The **Trail Value** parameter is used as number of bars to determine a highest high or lowest low of the data series. Thus if the **Trail Value** is 4, your trailing stop for long position is 4 bars lowest low of the price series and for short position is 4 bars highest high of the price series.

ATR	The **Trail Value** parameter is used as the number of period for the average true range stop. The classic method of ATR is to take out the position from the best price level offset by 3 times the ATR value.

Trail Value

The value parameter for the specified trailing stop style.

Instead of profitability, you can specify an optional points parameter to indicate certain points must be reached before the trailing stop will kick in. If the optional parameter is used, the trailing stop does not guarantee a profit. Instead, the trailing stop will become a hybrid between classic trailing stop and stop loss order.

The following table uses a long position as an example.

Trailing Style	Trail Value	Meaning with a Long Position
Points	5	Trailing stop will be placed at 5 points below current highest price when the trailing stop will produce a profit.
Amount	50, 5	Trailing stop will be placed at $50 below current highest price. The order will be placed only if the current price had reached 5 points above the entry price of the long position.
Percent	10, 5	Trailing stop will be placed %10 below current highest price. The order will be placed only if the current price has reached 5 points above the entry price of the long position.

For short positions, trailing stop will issue buy stop at above the current lowest price.

Stop Loss Style

Stop loss orders are used to control losses. In a long position, when the price goes below the specified threshold, a market order will be issued to close the position.

These options are available - None, Points, Percent, Amount.

Stop Loss Style	Meaning
None	No stop loss orders are placed.
Points	The **Stop Loss Value** parameter is used as is to offset from the entry price to calculate the stop loss price. For example, if the system is in a long position, the **Stop Loss Value** is 1, and the entry price is $50, then the stop loss price is $49. The stop loss price does not change over the life time of a position.

Percent	The **Stop Loss Value** parameter is used as a percentage of the entry price value to calculate the stop loss price. For example, a long position with entry price of $50 having a **Stop Loss Value** of 2 means the stop loss price is set at $49 = $50 - $50 x 2%.
Amount	The **Stop Loss Value** parameter is used as the total dollar amount that you are willing to lose on any given position. It is translated to a stop loss price by taking into account the number of shares or contracts you are trading and the price multiple of the price series.

Stop Loss Value

The value parameter of the chosen **Stop Loss Style**. This parameter is ignored when the **Stop Loss Style** is set to None.

Target Style

Targets are specified to close positions when certain profit is reached. In general, target orders are limit orders. The exception is Bars target which uses on close order. Each target style has a 1/2 position version that closes only half of the position. The remaining half is closed by other criteria such as trailing stops and EOD exit. If you intended to use the 1/2 position version, you must make sure the **Single entry per direction option** is set under the **System** tab in indicator edit window.

In real life trading, limit orders are subject to market conditions and whether market makers will honor the limit order.

These options are available - None, Points, Percent, Amount, Bars, Points 1/2 Position, Percent 1/2 Position, Amount 1/2 Position, Bars 1/2 Position.

Target Style	Meaning
None	No target orders are placed.
Points	The **Target Value** parameter is used as is to offset from the entry price to calculate the target price. For example, if the system is in a long position, the **Target Value** is 1, and the entry price is $50, then the target price is $51. The target price does not change over the life time of a position.
Percent	The **Target Value** parameter is used as a percentage of the entry price value to calculate the target price. For example, a long position with entry price of $50 having a **Target Value** of 2 means the target price is set at $51 = $50 + $50 x 2%.

Amount

The **Target Value** parameter is used as the total dollar amount that you are trying to get on any given position. It is translated to a target price by taking into account the number of shares or contracts you are trading and the price multiple of the price series.

Bars

The **Target Value** is number of bars. For example, if **Target Value** is 5, a market order to close the position is placed 5 bars after the order that initiates the position.

Points 1/2 Position

Same as Points except only 1/2 of the position is closed.

Percent 1/2 Position

Same as Percent except only 1/2 of the position is closed.

Amount 1/2 Position

Same as Amount except only 1/2 of the position is closed.

Bars 1/2 Position

Same as Bars except only 1/2 of the position is closed.

Target Value

The value parameter of the chosen **Target Style**. It is ignored if **Target Style** is set to None.

Choosing a Closing Strategies

Use **Long Exit** or **Short Exit** when the closing strategy can be specified in a formula. For example, if you want to use moving average crossover as an exit condition. Orders are placed as market orders so they are subjected to slippage.

Use trailing stop if you want your trades to ride a trend. Profit is not limited as the stop orders are cancel and replaced as price goes up (for long) or as price goes down (for short). Stop order are market orders so they are subjected to slippage. You also need discipline to carry out trailing stops in real life trading.

Use stop loss orders to limit losses. Stop loss orders are market orders so they are subjected to slippage.

Use target if you have a certain profit objective in mind. Except for bar style, target orders are limit order so if the orders get filled, the profit is guaranteed. Limit orders have its weakness as whether an order will be filled is subjected to whether market makers will honor limit orders in real life trading.

Useful Techniques

A useful tool is provided by indicator edit window's **Save Param** and **Load Param** buttons. You can easily save different Backtest EZ setup into separate files and recall them quickly.

To experiment with new ideas of trading strategies, try something simple in the beginning and then add extra rules like money management later. It is better that you understand the weaknesses and strong points of a raw trading system that is always in the market with either a long or short position before tweaking with other possibilities.

Backtest EZ Advanced Techniques

Backtest EZ is way more powerful than it seems. By using it smartly with more complex formulas for the signals, or together with other indicators within a chart, you can construct serious trading system easily without tedious programming and can quickly investigate new ideas with ease.

Following section will provide some advanced techniques that are proven useful for use with Backtest EZ.

How to generate a simple price type (moving average) cross over signal?

NeoTicker® has 2 built-in indicators called cross above (xabove) and cross below (xbelow).

To create a long entry signal based on the cross over of 5-bar simple moving average over a 20-bar exponential moving average, you can use the following formula:

```
xabove (average (data1, 5), xaverage (data1, 20))
```

To create a simple system that is always in the market based on the cross over of a 50-period simple moving average and a 200-period simple moving average, you can use the follow formulas.

For **Long Entry**:

```
Xabove (average (data1, 50), average (data1, 200))
```

For **Short Entry**:

```
Xbelow (average (data1, 50), average (data1, 200))
```

How to generate a cross over signal on my oscillator (RSI, Stochastic, etc.)?

Similar to cross over signals on moving averages, we can use the xabove and xbelow indicators to generate the necessary signals.

For example, you want to go long when the Slow K is crossing above its 5 period exponential moving average. The formula will look like this.

```
xabove (slowk (data1, 5, 3), xaverage (slowk (data1, 5, 3), 5))
```

How to combine multiple signals and filters together?

To combine multiple signals together you can use the and/or boolean operators to chain the conditions together.

One tricky issue is that when using a single signal-generating indicator like xabove or xbelow, you do not need to check if it is greater than zero. Its value is used directly. On the other hand, when connected with the other conditions, you have to clearly indicate that you are using such indicator as a boolean expression, thus you need to append `> 0` to the expression before you can connect it with the other conditions.

For example, you are trying to construct a long entry with 5 period simple moving average cross above the 100 period simple moving average while the close must be greater than your trend indicator 20 period exponential moving average. The formula will be:

```
xabove (average (data1, 5), average (data1, 100)) > 0
and data1 > xaverage (data1, 20)
```

How to apply time based filters?

After analyzing your system, you determine that if you can avoid trading a particular time, say from 12:00 to 13:00, your system will perform better, you can simply append the following formula to your long and short entries:

```
and (time < maketime (12, 0, 0) or time > maketime (13, 0, 0))
```

In another situation, you want to filter out two segment of time, say, from 12:00 to 13:00 and from 14:00 to 15:00. Then you can append the following to your formula:

```
and (time < maketime (12, 0, 0) or
    (time > maketime (13, 0, 0) and time < maketime (14, 0, 0))
or
     time > maketime (15, 0, 0))
```

How to include more than 2 data series in generating signals?

To construct trading system based on multiple data series, you can utilize the indicator on indicator feature. By having the necessary calculations done by other indicators that are linked to other series, you can then combine all these signals into one single formula indicator as the signal generating Link 2 of Backtest EZ.

For example, you are trading QQQ and would like to use the difference between Advance and Decline Issues of Nasdaq as the source for your signals. You can then add a Subtraction indicator to the chart linked to the advance issue data and the decline issue data. Then you can add Backtest EZ and have the primary link set to QQQ while the second link set to the Subtraction indicator. Now you can utilize this indicator in your Long / Short signals.

How to have both a fixed time exit and a fixed target exit at the same time?

Backtest EZ has a feature that allows adding a Points threshold for the Trailing Stop parameter. An open position must exceed the required threshold before the trailing stop will kick in. By setting the threshold to your target Points value and the actual trailing stop value to 0, you have created a virtual target.

For example, setting the parameter **Trailing Stop** to **Points** and the **Trail Value** to `0, 10`, you are effectively setting a target of 10 points profit.

Now set the **Target** to **Bars**, with the **Target Value** set to number of bars for the time exit. You have now created a trading system that has both fixed time exit and fixed target exit.

How to create my own trailing stop?

You can utilize the **Long Exit** and **Short Exit** parameters as trailing stop rules. Since the long exit and short exit rules are not used when there is no open position in the corresponding direction, you do not need to worry about their presents would open a new position in the wrong direction.

For example, you can construct a formula to exit a long position when the high of the current bar can no longer hold above the 50 period simple moving average. The value for **Long Exit** would be:

```
high < average (data1, 50)
```

This exit signal works well as a trailing stop.

How to exit the trading system by the end of each trading day?

When you add the Backtest EZ indicator to your chart, there is an option called **Close Position EOD** located under the **System** tab. Check this option to force a position to close at the last bar of the trading day.

How to add equity curve filter to my system?

To trade off the equity curve of your Backtest EZ system, add it once to the chart with your original settings. Then add another Backtest EZ indicator to the same chart again. This time use the original Backtest EZ indicator as the second link and then append the following formula segment to your **Long Entry** and **Short Entry** formulas,

```
and average (data2, 50) > data2
```

What this segment does is to filter out entry signals when then equity curve is below its 50 period simple moving average. You can design you own equity filter based on this technique.

If your system is an always in the market system that has no long exit or short exit signals, then you may have to add these rules when you try to filter out the entry signals because when you filter out the entries you are also filtering out the exit for the open positions. A simple way to do this is copy the original long entry to the short exit and copy the original short entry to the long exit. This way, you can guarantee to exit the same way as the original system while the entries are filtered out based on the equity curve.

How do I save my winning systems?

In the Indicator Edit window, there is a **Save Param** button that allows you to save your parameters to a file and retrieve it later in another occasion using the **Load Param** button.

When you save your Backtest EZ parameters, give it a name that is more meaningful such that you can easily identify what it is. For example, you can name a Backtest EZ system you have created as `2003047_SP_1Min_Sys`.

How to change Backtest EZ to recognize that SP emini is $50 per point?

When you add or edit the Backtest EZ indicator in your chart, from within the Indicator Edit window, switch to the **System** tab, look at the **Default Data Settings**. By changing the **Price Multiple** to 50, you can get proper calculation of the profit/loss on trading the SP emini contract.

How to recognize delayed signals?

To recognize a signal 1 bar late such that you can have confirmation by price, you can write your formula similar to the following.

For **Long Entry**:

```
xabove (1, average (data1, 20), average (data1, 50)) > 0 and
close > close (1)
```

The above formula looks for the cross above signal from 1 bar ago and then confirm that with an up close before the long signal is taken.

Deployment

If you want to deploy your trading system for real-life trading, you should read the section ***Trading System Life Deployment Guide*** (see "Trading System Live Deployment Guide" on page 785).

Color Plot Formula

The Color Plot Formula indicator allows you to define color markings on a chart using formula.

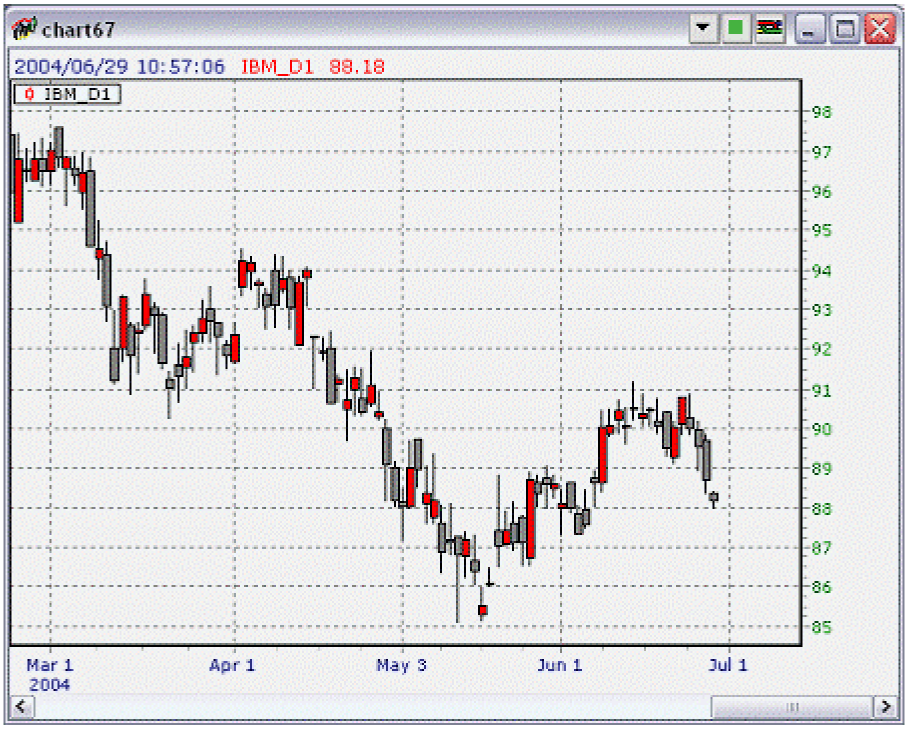
What can be done with Color Plot Formula?

- You can use it as a color plot tool with your own custom formulas. You can easily stack the color plots into the same pane or in different panes within a chart. You can also have up to 5 different colors within each color plot formula indicator.
- You can use it like the Highlight Bar Formula indicator, this time with the possibility of drawing multiple colors. Thus you directly paint over existing price bars with colors to identify and highlight your conditions easily.
- You can construct customized markings at your calculated positions with varying colors to identify special signals with precise price information directly marked on the chart.

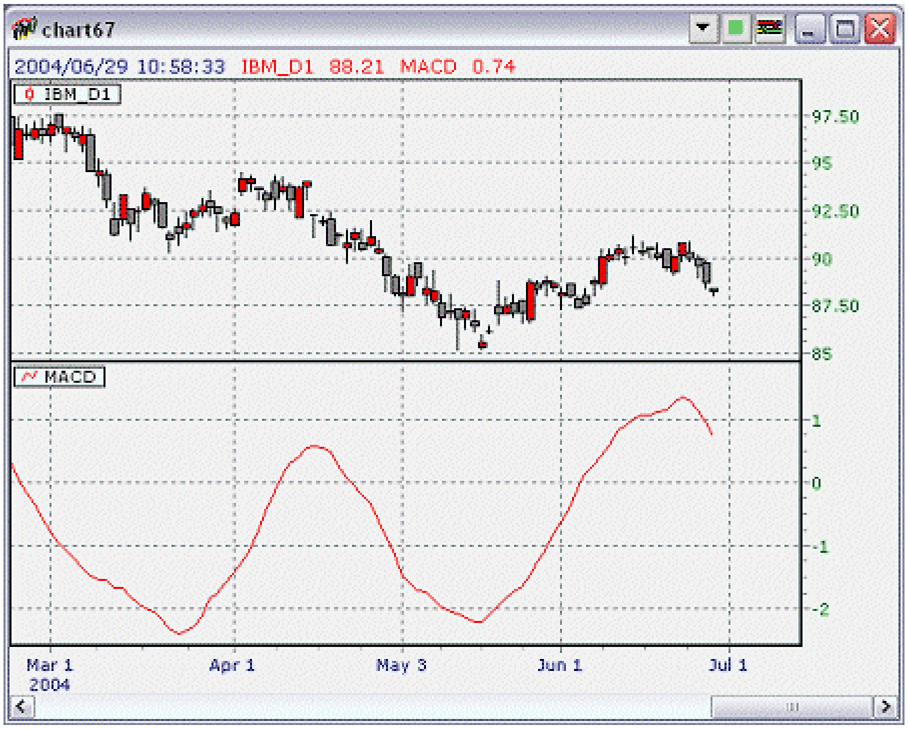
Using Color Plot Formula

Lets say you are interested to identify the current trend of MACD and make that a color plot on your chart.

Create a new chart.

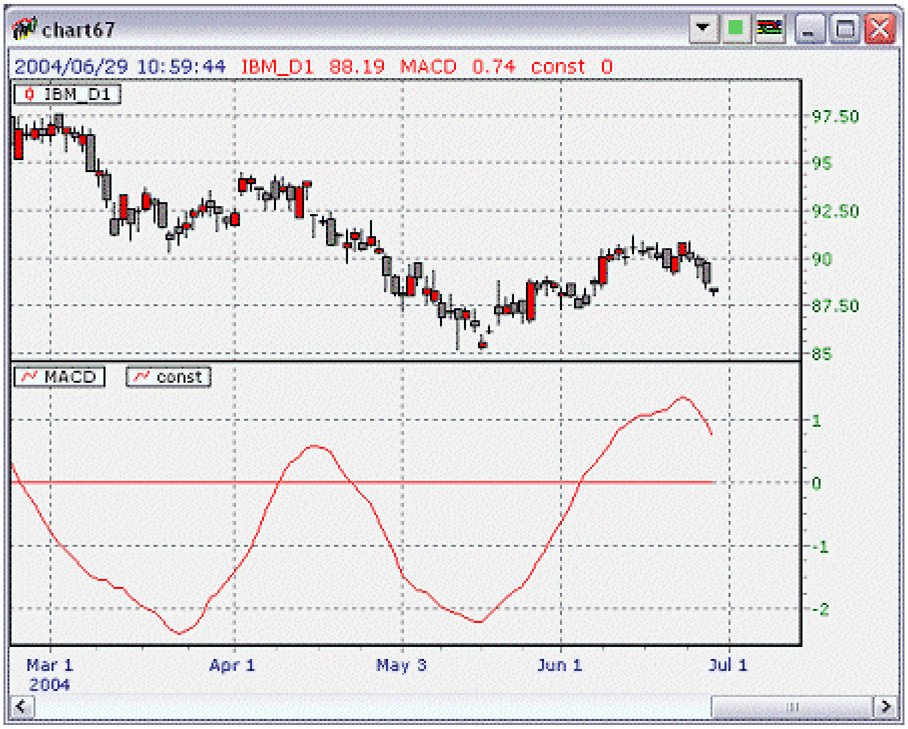


(Optional) Now, add the MACD indicator. It is listed under the name “Moving Average Convergence Divergence”. The default parameters are simple 12 period and simple 26 period. If you do not need to change the parameters, all you need to do is simply double-clicking the indicator name and it will be applied to the chart.



For trend identification, we will use the standard interpretation of greater than zero (0) is up trend while less than zero (0) is down trend.

(Optional) You can make the reading of MACD easier by adding the Constant indicator to the pane with MACD. Set the parameter of constant to 0 to partition the display.

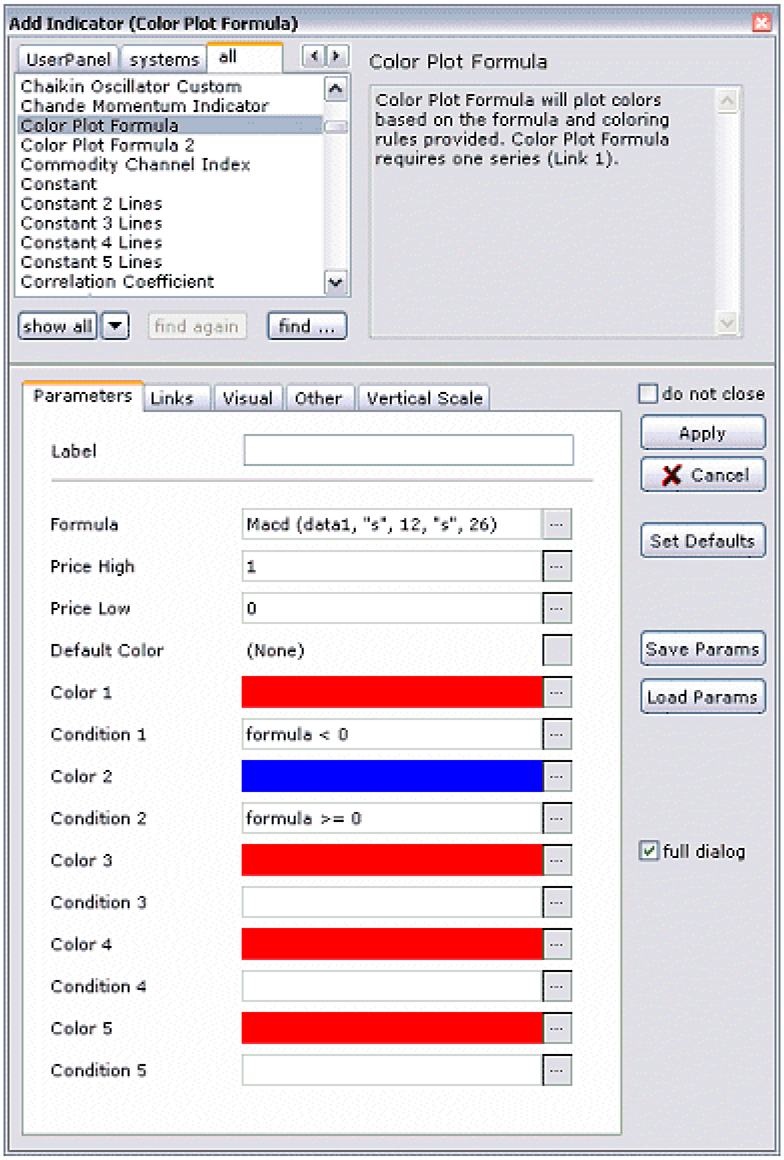


We will add the Color Plot Formula Indicator now. Color Plot Formula has many parameters. To view all parameters, expand the indicator set up window by enlarging it vertically.

Lets put in the formula first. Type the following formula into the **Formula** parameter.

```
Macd (data1, "s", 12, "s", 26)
```

The rest of the parameters can be set to the values in the screen shot below.



Color Plot Formula

Color Plot Formula will plot colors based on the formula and coloring rules provided. Color Plot Formula requires one series (Link 1).

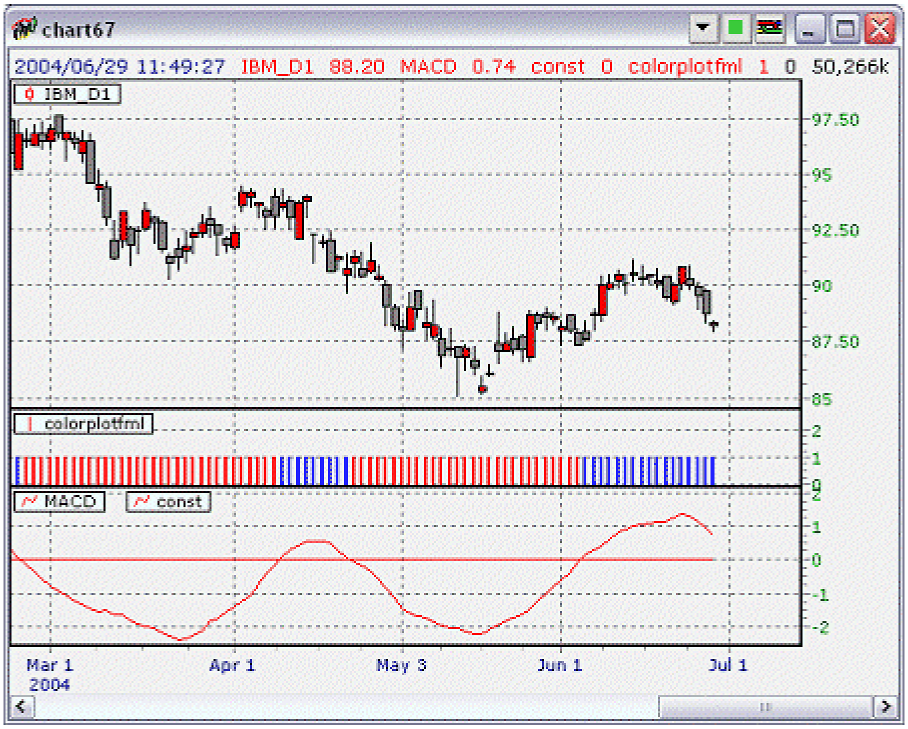
Label	
Formula	Macd (data1, "s", 12, "s", 26)
Price High	1
Price Low	0
Default Color	(None)
Color 1	🟥
Condition 1	formula < 0
Color 2	🟦
Condition 2	formula >= 0
Color 3	🟥
Condition 3	
Color 4	🟥
Condition 4	
Color 5	🟥
Condition 5	

In color plot formula, the Formula parameter can be referenced by the Conditions as `formula`. For example, the parameter **Condition 1** is expressed as `formula < 0`.

What color plot formula does is it marks the region where a condition is true with a specific color. In this example, the two conditions are MACD is below and above 0, which are colored in red and blue respectively. Conditions that do not have formula (e.g. **Condition3**, **Condition4**, etc) are ignored by color plot formula.

(Optional) You may want to change the width of the indicator to 2 to get a thicker color bar.

Apply the indicator now.

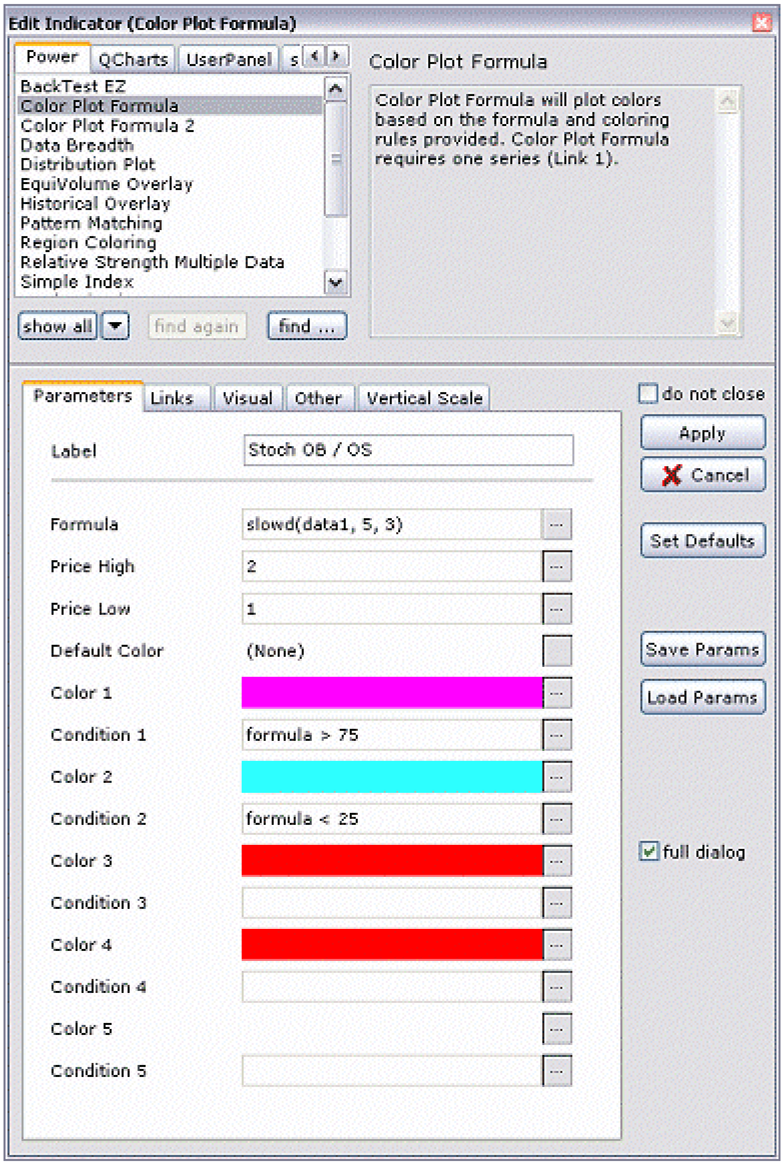


In the chart, the red segment of color plot is where MACD is below 0; the blue segment is where MACD is above 0. You have now summarized the information you wanted from the MACD into a simple color plot that you can spot the changes at once.

But what if you want more than just one color plot formula?

Lets add another one say Stochastic SlowD overbought / oversold color plot to the chart.

Add another Color Plot Formula. Set the parameters as in the screen shot below.



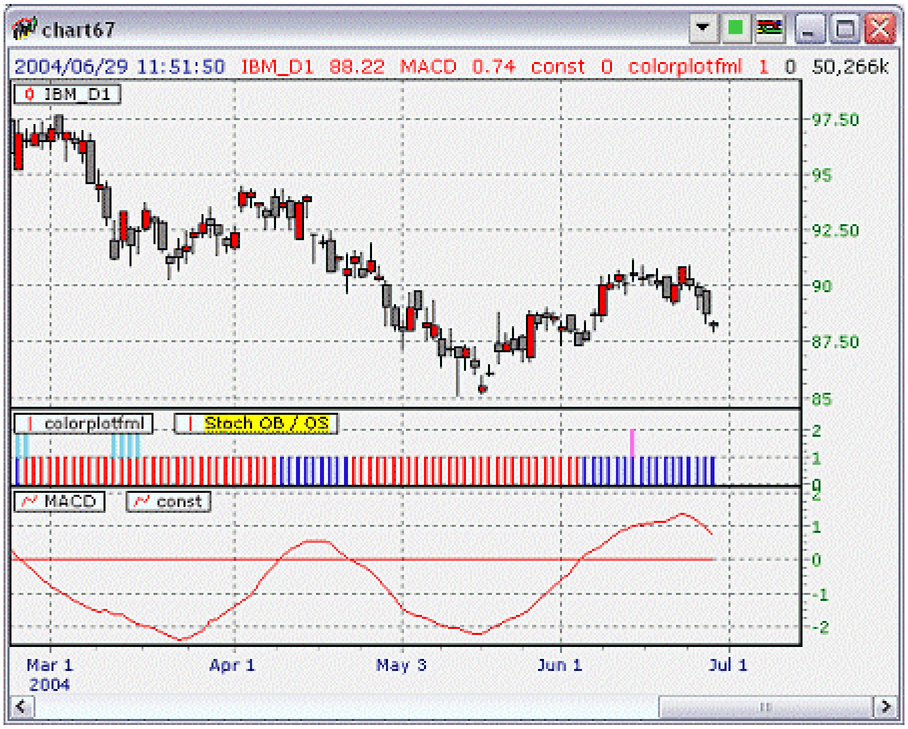
Edit Indicator (Color Plot Formula)

Color Plot Formula

Color Plot Formula will plot colors based on the formula and coloring rules provided. Color Plot Formula requires one series (Link 1).

Label	Stoch OB / OS
Formula	slowd(data1, 5, 3)
Price High	2
Price Low	1
Default Color	(None)
Color 1	🟪
Condition 1	formula > 75
Color 2	🟦
Condition 2	formula < 25
Color 3	🟥
Condition 3	
Color 4	🟥
Condition 4	
Color 5	
Condition 5	

Apply the indicator and move it to the pane where MACD Trend is. To move an indicator, you can drag either the legend or the series itself to the target pane area.



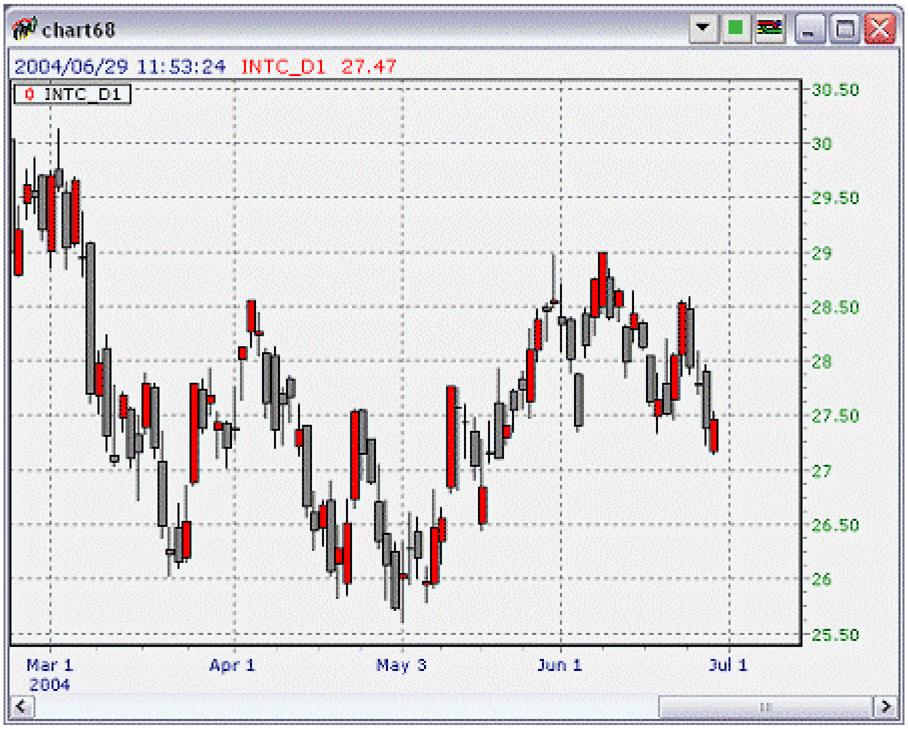
The key to properly stack multiple color plot is the setting of the parameters **Price High** and **Price Low**. As long as the range of the values used do not overlap, you will be able to see all your color plot formulas clearly. You can also use object ordering in time chart to help you stack.

Try modifying the conditions to see what effects they have on the chart. It will help you learn using the color plot formula faster.

Using Color Plot Formula as Bar Highlight

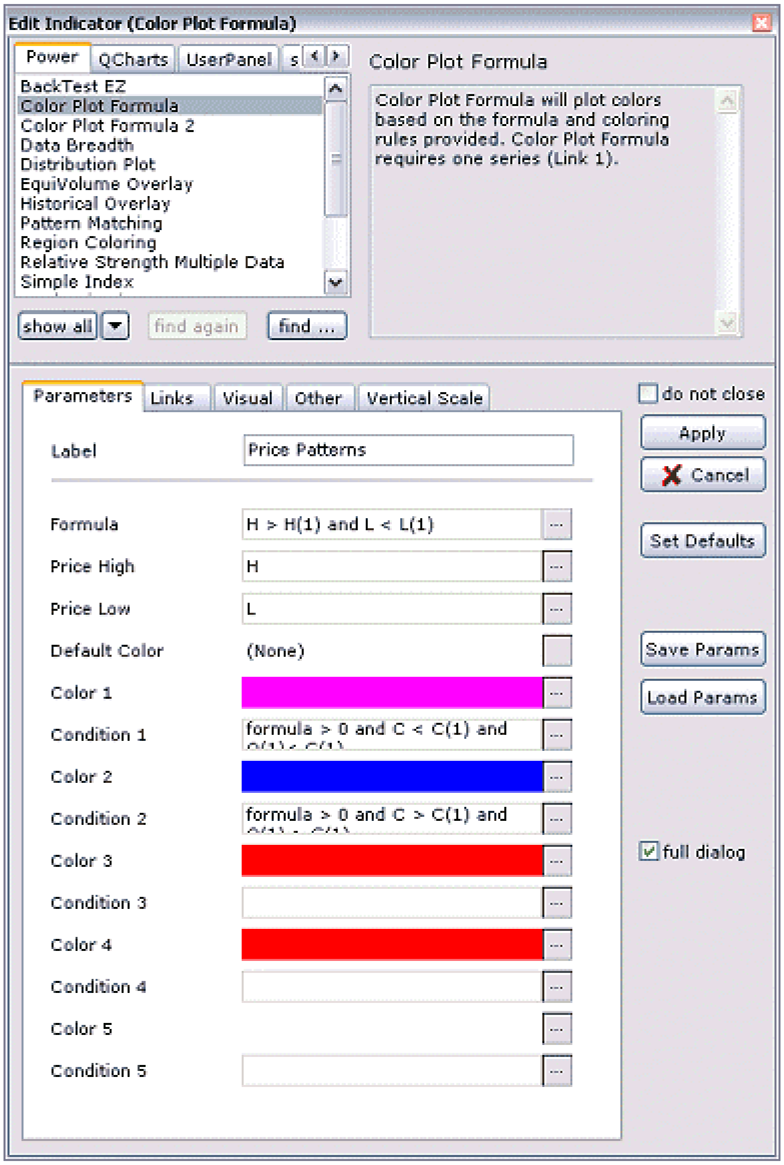
Lets say you are interested in highlighting some price patterns and would like to make sure you do not miss them when they happen.

Create a new chart.



Now add the color plot formula indicator.

Set the parameters as follows:



Edit Indicator (Color Plot Formula)

BackTest EZ
Color Plot Formula
Color Plot Formula 2
Data Breadth
Distribution Plot
EquiVolume Overlay
Historical Overlay
Pattern Matching
Region Coloring
Relative Strength Multiple Data
Simple Index

Color Plot Formula

Color Plot Formula will plot colors based on the formula and coloring rules provided. Color Plot Formula requires one series (Link 1).

□ do not close

☑ full dialog

Label	Price Patterns
Formula	H > H(1) and L < L(1)
Price High	H
Price Low	L
Default Color	(None)
Color 1	🟪
Condition 1	formula > 0 and C < C(1) and
Color 2	🟦
Condition 2	formula > 0 and C > C(1) and
Color 3	🟥
Condition 3	
Color 4	🟥
Condition 4	
Color 5	
Condition 5	

The **Formula** parameter simply defines an outside bar pattern. The formula is:

```
H > H(1) and L < L(1)
```

The **Condition 1** parameter checks for an up bar 1 period ago and then a down close. The formula is

```
formula > 0 and C < C(1) and O(1) < C(1)
```

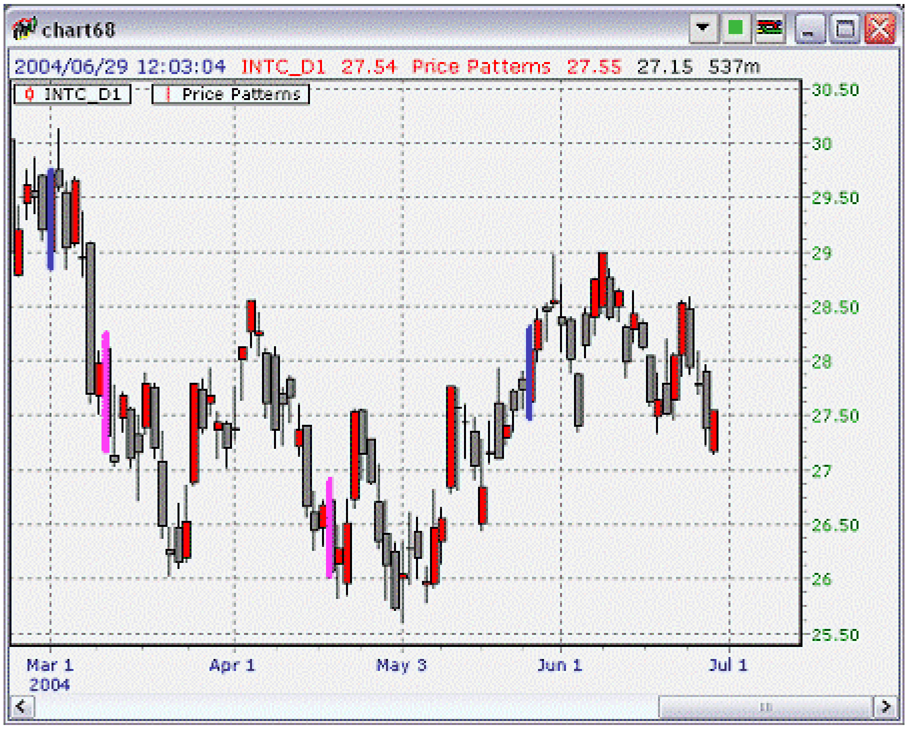
The **Condition 2** parameter checked for a down bar 1 period ago and then a up close. The formula is

```
formula > 0 and C > C(1) and O(1) > C(1)
```

Notice the **Price High** and **Price Low** parameters are set to H and L respectively, telling Color Plot Formula indicator to mark the price from High to Low.

Under the **Visual** tab, set the **Pane** to **Active Pane**. Set **Width** to 4.

Apply the indicator.



To add more conditions simply edit the indicator and type your conditions into **Condition 3**, **Condition4**, etc.

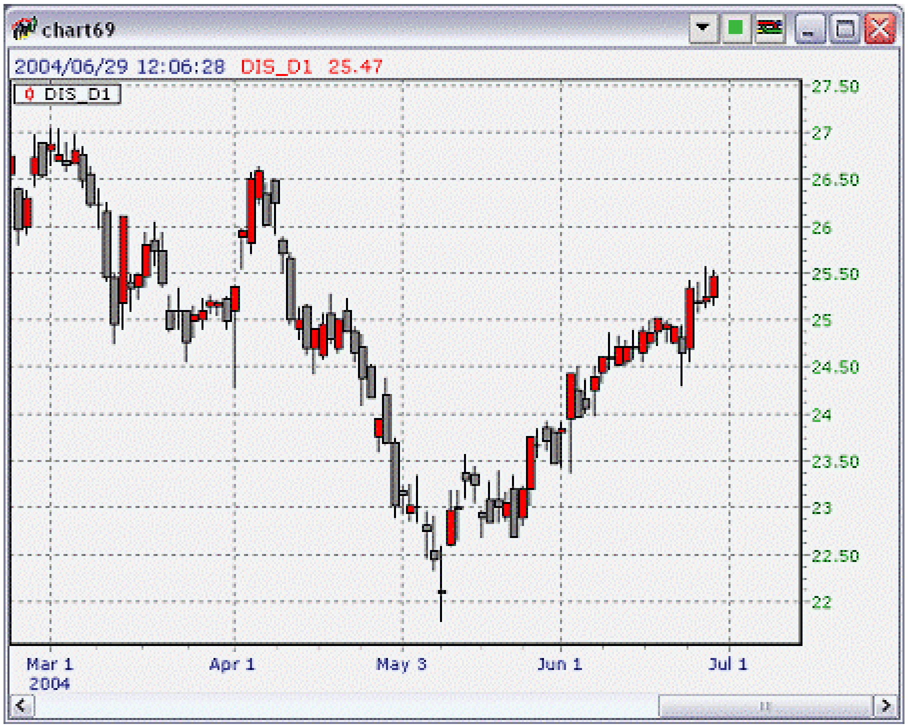
Using Color Plot Formula as Special Price Markers

There are times that you want to mark specific price area for your price targets, no matter it is an area to buy or sell.

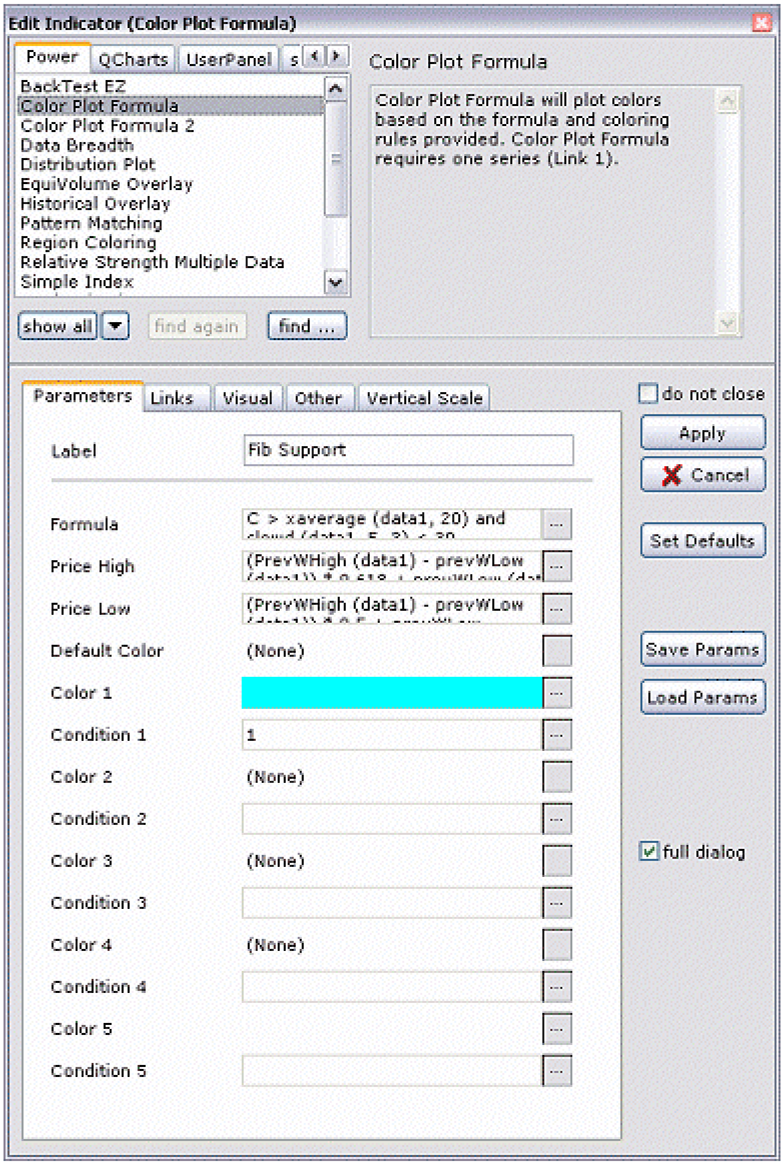
How do you draw those areas onto your chart to let you see them visually?

Lets say you are interested in possible buying condition of low stochastic with price pulling back to previous month’s Fibonacci retracement level of 0.618 to 0.5.

Create a new chart.



Then you can apply the color plot formula indicator to the chart with the following parameters.



Color Plot Formula will plot colors based on the formula and coloring rules provided. Color Plot Formula requires one series (Link 1).

Label	Fib Support
Formula	C > xaverage (data1, 20) and
Price High	(PrevWHigh (data1) - prevWLow
Price Low	(PrevWHigh (data1) - prevWLow
Default Color	(None)
Color 1	🟦
Condition 1	1
Color 2	(None)
Condition 2	
Color 3	(None)
Condition 3	
Color 4	(None)
Condition 4	
Color 5	
Condition 5	

☐ do not close

☑ full dialog

The **Formula** parameter specifies that the current price is higher than and close to the retracement range and stochastic is low:

```
C > xaverage (data1, 20) and slowd (data1, 5, 3) < 30
```

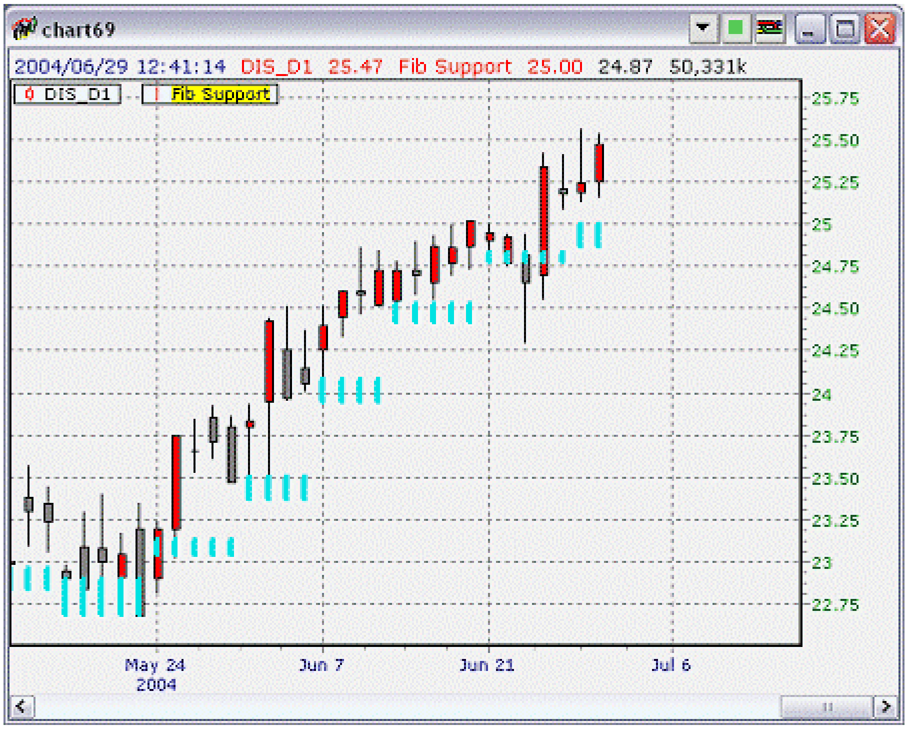
The **Price High** and **Price Low** parameters set the area to plot as the previous month’s Fibonacci retracement area.

For **Price High**, the formula is:

```
(PrevWHigh (data1) – prevWLow (data1)) * 0.618 + prevWLow
(data1)
```

For **Price Low**, the formula is:

```
(PrevWHigh (data1) - prevWLow (data1)) * 0.5 + prevWLow (data1)
```



Data Breadth

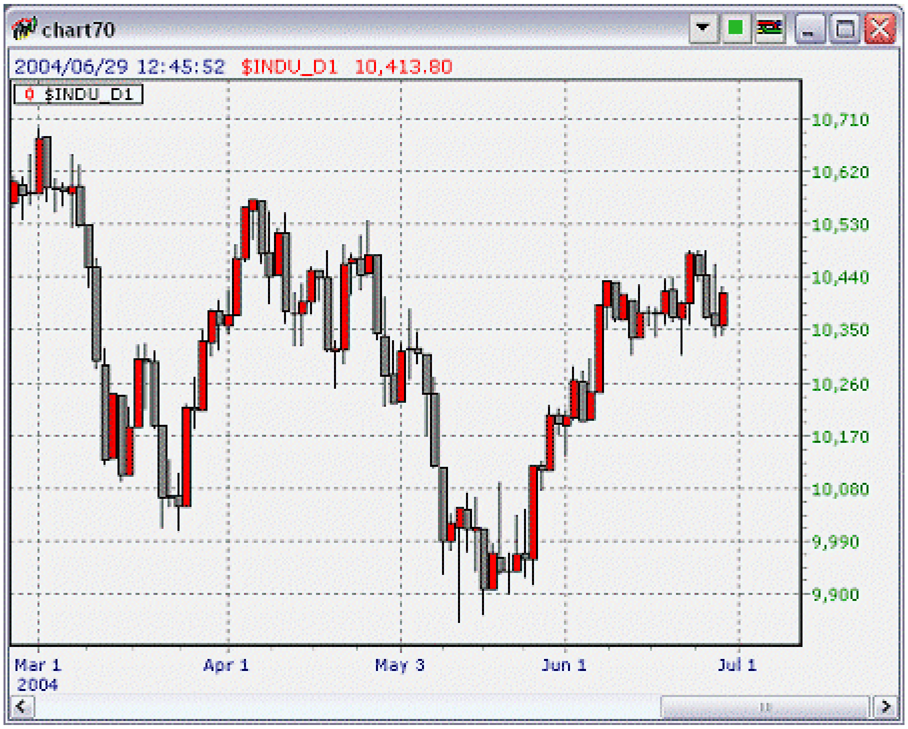
Data Breadth indicator provides a way to collect portfolio specific breadth data with adjustable time frame.

Data Breadth returns the following breadth data.

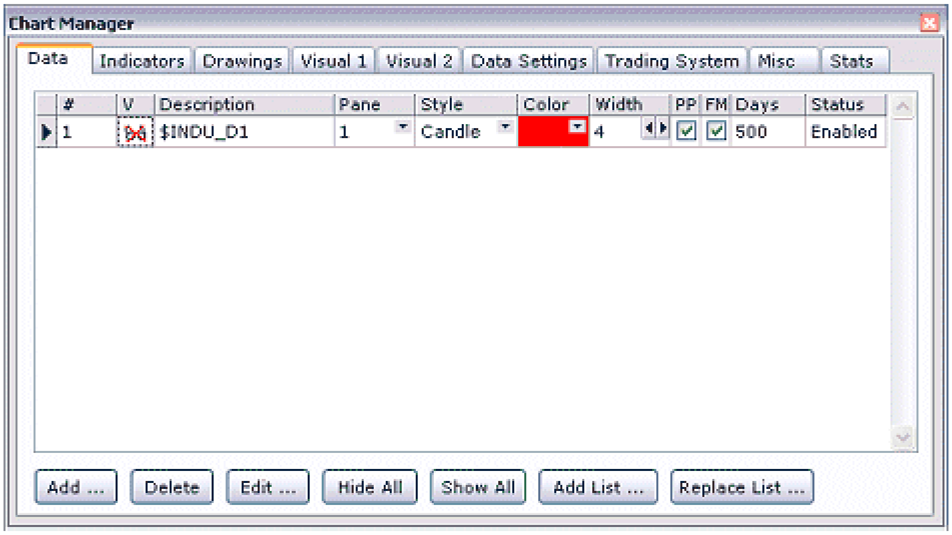
- advance issue
- decline issue
- advance volume
- decline volume

Lets go through a simple example first.

Create a chart with daily Dow Jones Industrial Average. The exact symbols is different depending on data feed (e.g. ^DJI, $INDU)



Open Chart Manager, press the **Data** tab. Turn off visibility of Dow Jones by pressing on the eye glasses 👓 .

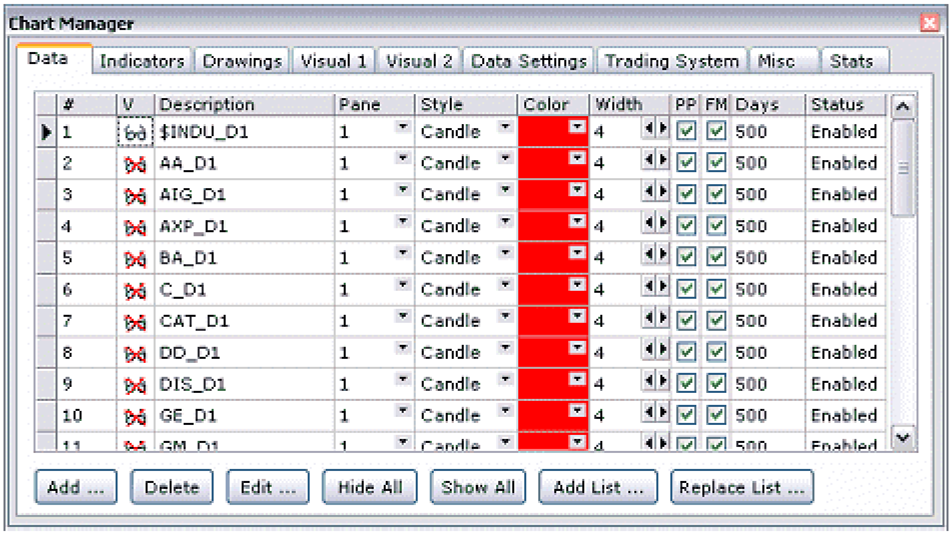


#	V	Description	Pane	Style	Color	Width	PP	FM	Days	Status
▶ 1	☒	$INDU_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled

Press the **Add List** button to add a symbol list to a chart using the setting of the first data series in the chart. Choose the Dow Jones 30 symbol list.

Symbols added by **Add List** will inherit the properties of the first data series. We will be calculating breadth using the stocks and visibility is not important. Thus we turn off visibility of Dow Jones first so the symbols that we are adding will inherit the invisibility.

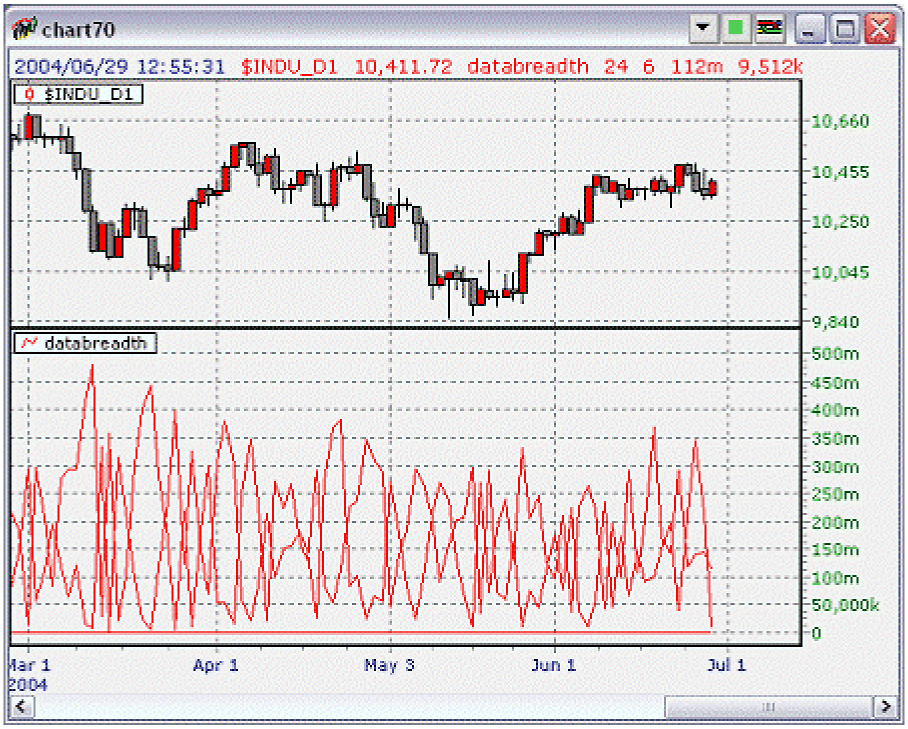
Time chart will now add the data one by one. After it is done with the data addition, you will be able to see them all on the chart manager. Turn the visibility of Dow Jones back on.



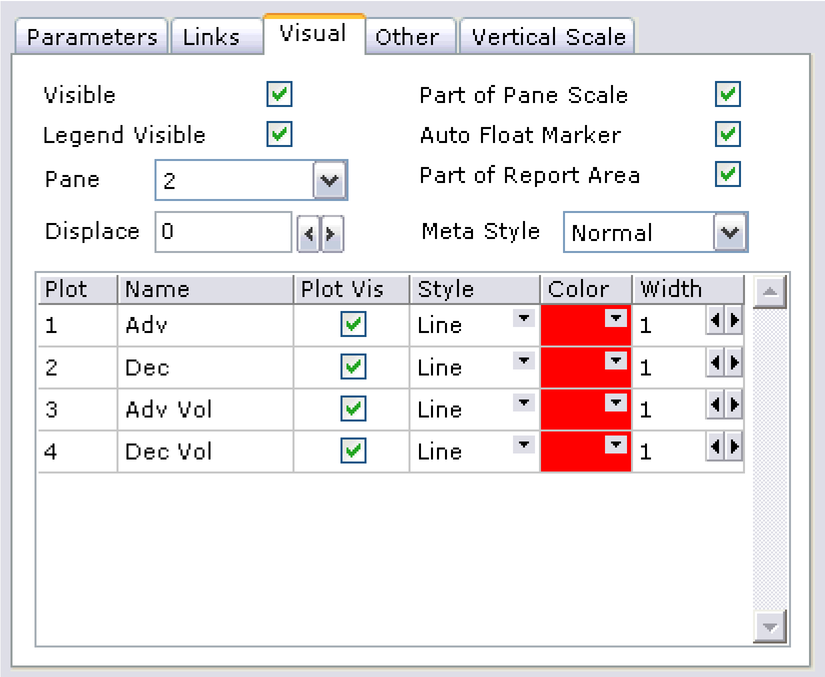
#	V	Description	Pane	Style	Color	Width	PP	FM	Days	Status
▶ 1	👓	$INDU_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
2	☒	AA_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
3	☒	AIG_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
4	☒	AXP_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
5	☒	BA_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
6	☒	C_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
7	☒	CAT_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
8	☒	DD_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
9	☒	DIS_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
10	☒	GE_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled
11	☒	GM_D1	1	Candle	🟥	4 ◄►	☑	☑	500	Enabled

Close Chart Manager.

Add the indicator DataBreadth. Since we are calculating breadth from the second data series, not including the Dow Jones index, the parameter **First Series** should be set to 2 while the parameter **Last Series** should be set to 31 (the last data series).



Recall that the first and second plots of Data Breath are the advancing and declining issues.



Visible ☑

Legend Visible ☑

Pane 2

Displace 0

Part of Pane Scale ☑

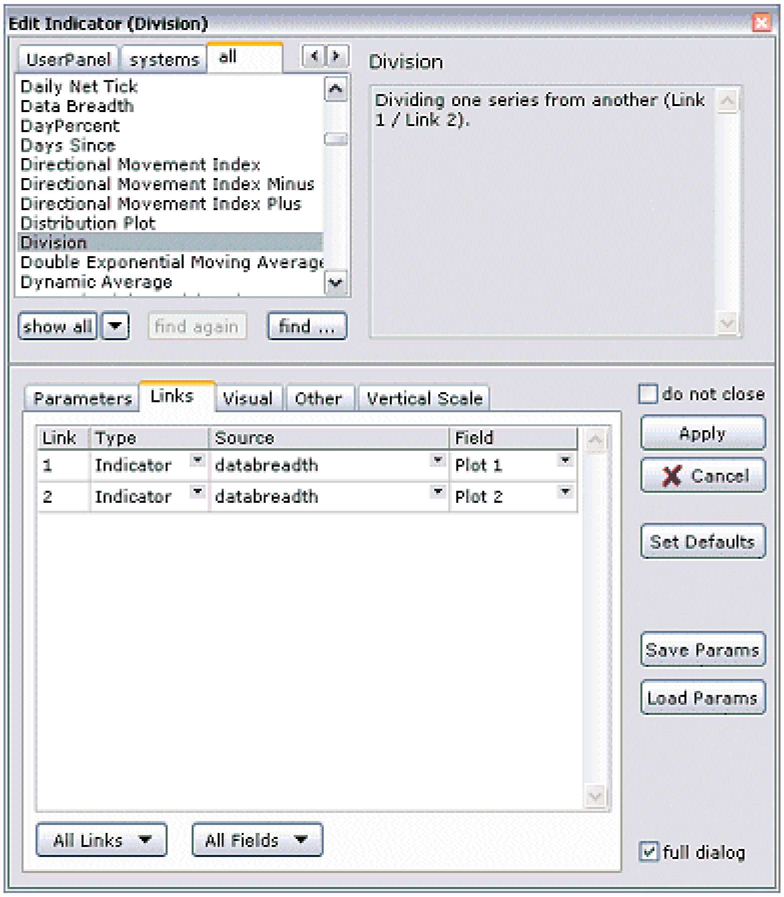
Auto Float Marker ☑

Part of Report Area ☑

Meta Style Normal

Plot	Name	Plot Vis	Style	Color	Width
1	Adv	☑	Line	🟥	1
2	Dec	☑	Line	🟥	1
3	Adv Vol	☑	Line	🟥	1
4	Dec Vol	☑	Line	🟥	1

You can create an advance decline ratio calculations by applying a division indicator to first and second plots of Data Breadth.



Edit Indicator (Division)

Daily Net Tick
Data Breadth
DayPercent
Days Since
Directional Movement Index
Directional Movement Index Minus
Directional Movement Index Plus
Distribution Plot
Division
Double Exponential Moving Average
Dynamic Average

Division

Dividing one series from another (Link 1 / Link 2).

Link	Type	Source	Field
1	Indicator	databreadth	Plot 1
2	Indicator	databreadth	Plot 2

□ do not close

☑ full dialog

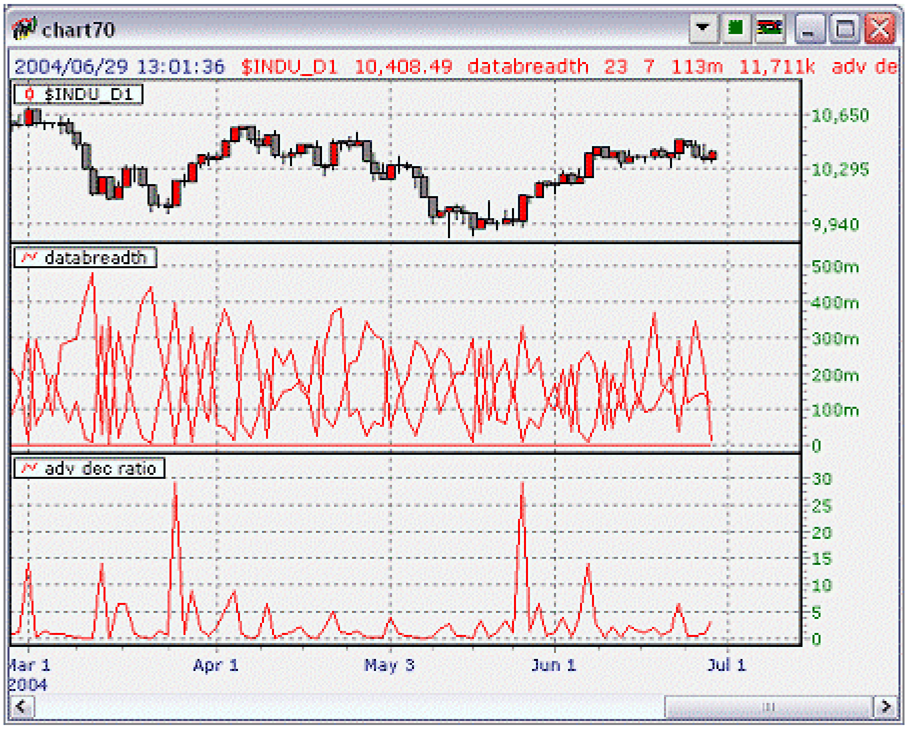


chart70
2004/06/29 13:01:36 $INDU_D1 10,408.49 databreadth 23 7 113m 11,711k adv de
0 $INDU_D1
10,650
10,295
9,940
databreadth
500m
400m
300m
200m
100m
0
adv dec ratio
30
25
20
15
10
5
0
Mar 1
2004
Apr 1
May 3
Jun 1
Jul 1

Distribution Plot

Overview

Distribution Plot provides an integrated way for the user to research on price patterns and other recurring conditions. You can enter any complex conditions into the indicator and freely construct measurements that you are interested in. Both historical and real-time results can be displayed together so that you can easily compare current condition against historical account.

For example, do you know the historical distribution of up down bars ratio for the symbol MSFT? And how is the statistics doing lately?

Or, you are interested to know if the stochastic indicator is in fact mean reverting and would like to design a trading system around its properties. Studying the distribution of statistics would definitely help in unveiling interesting facts about the indicator on the data you are analyzing.

If you need to further customize the distribution display, the indicator script is provided for you to customize the script directly so that you can carry out even more complex studies based on the foundation of distribution plot.

Using the Indicator

First, lets try to take a look at the up down bars ratio of MSFT.

Create a chart of MSFT daily data, loading 2000 days.

(Optional) Turn off grid lines using Chart Manager, under Visual 1 tab.

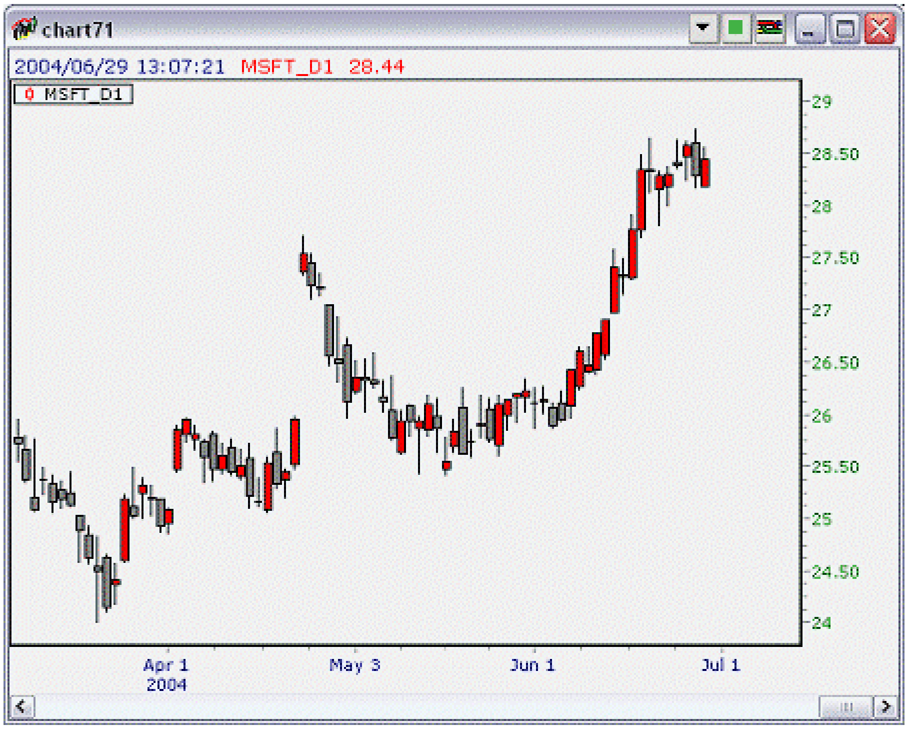
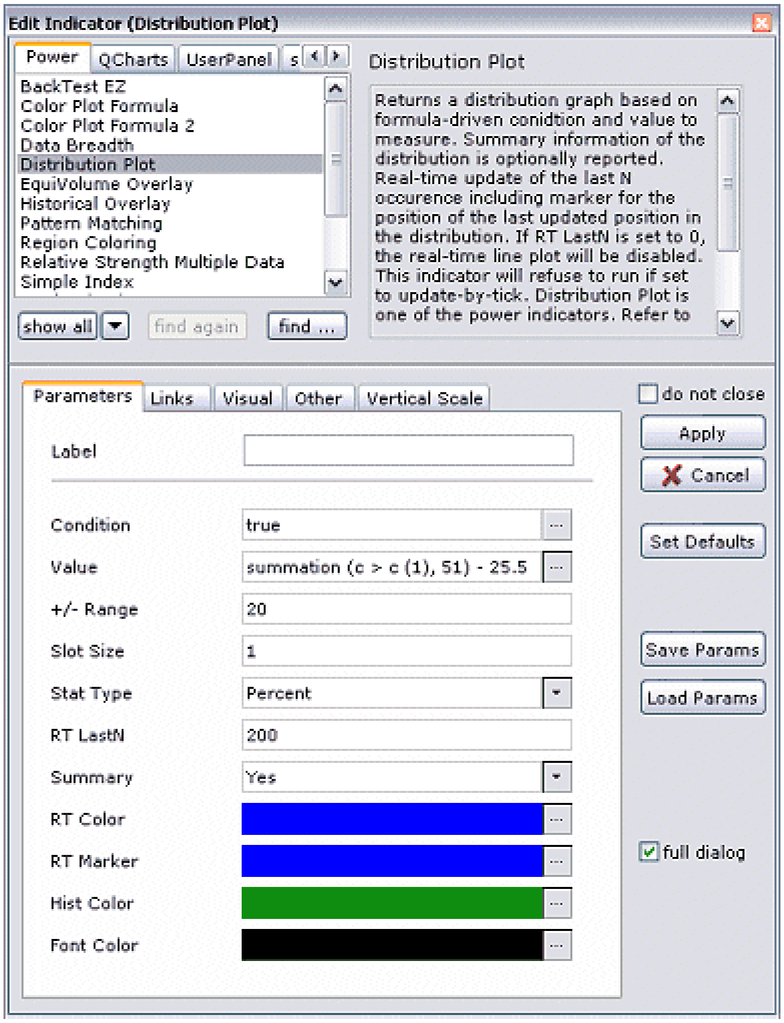


chart71
2004/06/29 13:07:21 MSFT_D1 28.44
0 MSFT_D1
29
28.50
28
27.50
27
26.50
26
25.50
25
24.50
24
Apr 1
2004
May 3
Jun 1
Jul 1

Add indicator distribution plot with the following parameters:



Returns a distribution graph based on formula-driven conidtion and value to measure. Summary information of the distribution is optionally reported. Real-time update of the last N occurence including marker for the position of the last updated position in the distribution. If RT LastN is set to 0, the real-time line plot will be disabled. This indicator will refuse to run if set to update-by-tick. Distribution Plot is one of the power indicators. Refer to

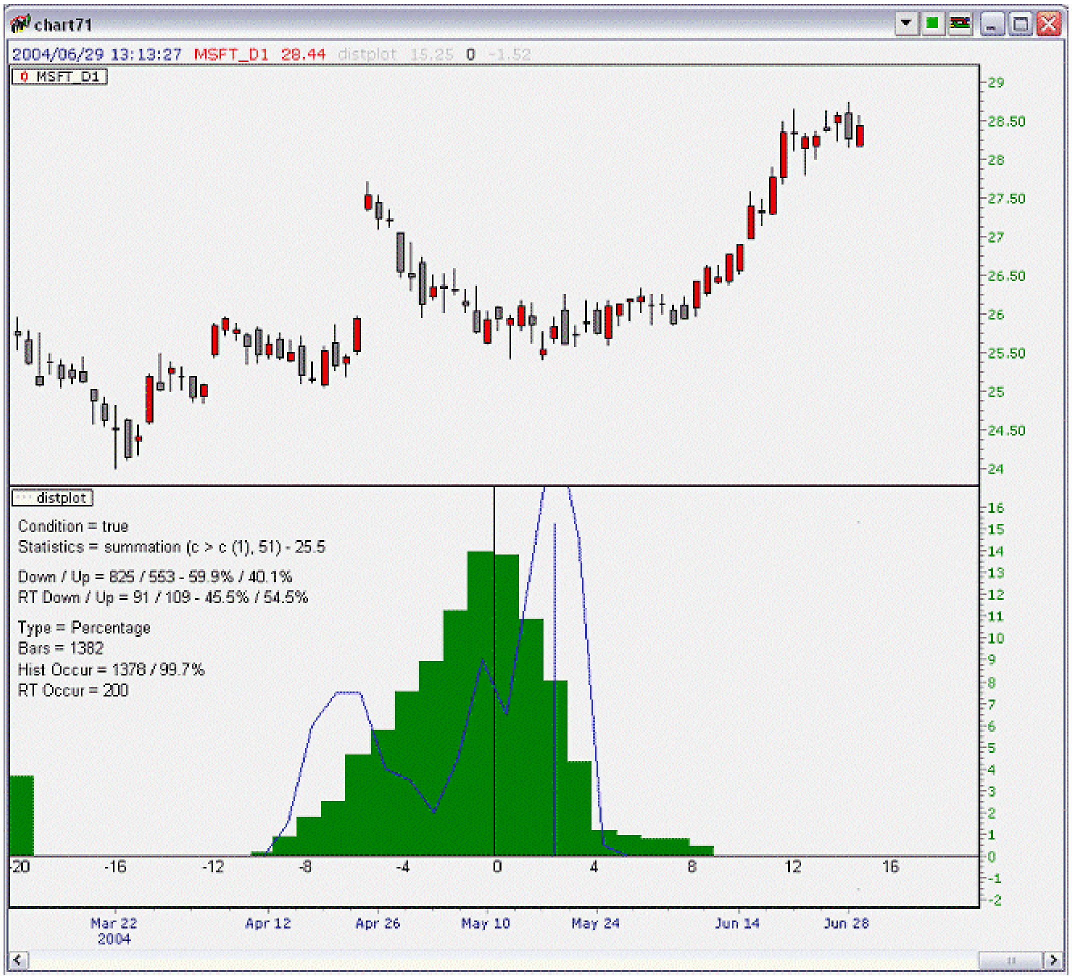
Label	
Condition	true
Value	summation (c > c (1), 51) - 25.5
+/- Range	20
Slot Size	1
Stat Type	Percent
RT LastN	200
Summary	Yes
RT Color	🟦
RT Marker	🟦
Hist Color	🟩
Font Color	⬛

☐ do not close

☑ full dialog

Parameter	Value
Condition	true
value	`summation (c > c (1), 51) - 25.5`
+/- range	20
slot size	1
stat type	Percent
RT LastN	200
Summary	Yes
RT Color	Blue
RT Marker	Blue
Hist Color	Green
Font Color	Black

Press the **Apply** button and the distribution will show up after the statistics are collected.



Condition = true
Statistics = summation (c > c (1), 51) - 25.5

Down / Up = 825 / 553 - 59.9% / 40.1%
RT Down / Up = 91 / 109 - 45.5% / 54.5%

Type = Percentage
Bars = 1382
Hist Occur = 1378 / 99.7%
RT Occur = 200

The formula entered in the parameter value measures the number of up closes over every 51 days. By subtracting 25.5 from the summation, it now shows the number of up closes offset from the expected mean point of 51 / 2.

The result shows that over the historical data tested, MSFT is closing up approximately 40% of the time only.

Parameters Settings

Condition

Condition can be any valid script indicator formula. In the example, TRUE is used so that every bar in the data series are counted. If you fit in a formula like `C > C(1)`, then you will limit which bar to obtain statistics from.

Value

Value is the data to collect statistics on. In the example, the summation result is counted and the distribution graph is the end result.

+/- Range

The range of values to plot in the distribution graph. The distribution graph always center at zero (0). Thus you may want to offset your calculation similar to what we have done in the example to adjust the data to center around zero.

Slot Size

In order to count, you need to identify how the values are treated as similar item. Slot Size provided a way to group the values you are counting. For example, if slot size is set to 1, then values from 0 up to but not including 1 will be treated as similar data and collected together. Then the next slot are values ranging from 1 up to but not including 2, and so forth.

Stat Type

There are two statistics type - Count and Percent.

When you choose Count, the raw counting of the values is used. Thus you can examine the graph to get an idea how many times a particular value range has happened in the data series.

On the other hand, if you choose Percent, the distribution is normalized by the total count of the number of data values used. Thus the bars in the distribution graph will add upto 100 percent. This particular mode makes comparing the historical distribution against the real-time one easier.

RT LastN

By providing a number greater than 0, distribution plot will plot a line graph above the historical bar graph showing the distribution of the last N occurrences. At the same time, the last occurrence will also put a vertical line marker on top of the bar graph to show where this last occurrence is positioned at.

When set to zero (0), the real-time plot will be disabled.

Summary

When you choose to show the summary, a table of information is generated at the left side of the graph. The information provided is summary of the distribution being graphed.

RT Color

Color setting of the real-time line graph.

RT Marker

Color setting of the real-time last occurrence marker.

Hist Color

Color setting of the historical bar graph.

Font Color

Font Color setting of the graph labels and summary area.

Helpful Techniques in Using Distribution Plot

To collect data for say a buy set up, you can specify the condition to reference a few bars back, which will then enable you to measure the net effects easily.

For example, you can set the condition parameter to

```
xabove (3, average (data1, 10), average (data1, 30))
```

which will return true (1) if 3 bars ago there was a cross over.

Then you can set the value parameter to

```
(c - c (3)) / c (3) * 100
```

This would measure the net % change from the point of the cross over to 3 bars later.

If there is a bias in the positive side of the distribution, then you know the condition has some sort of buy bias which you can then proceed to construct trading system using the concept.

An useful service provide is provided by edit indicator window's **Save Param** and **Load Param** buttons. You can easily save different distribution plot setup into separate files and recall them quickly.

Distribution Plot vs. Excel

Similar distribution plot can be created by using Excel. Because time chart can send the data series to Excel directly, the obvious question is when to use distribution plot and when to use Excel.

Use Distribution Plot When:

- You require real-time update of the distribution plot
- You want to use NeoTicker®'s formula language to modify the data before analysis
- You want to use the condition parameter to help you filter data

Use Excel When:

- You do not require real-time update. Excel is not suitable for real-time updating large amount of data.
- What you analyze can be easily export to Excel, or constructed within Excel.
- You require publishing quality distribution chart.

Equivolume Overlay

Equivolume overlay indicator draws equivolume charts, which is drawn based on equal amount of traded volume. Equivolume chart is useful to gain a sense to trading frequency, as well as support resistance levels based on volume information.

Example

1 Open a 1-minute CSCO chart.

2 You may want to add volume histogram for reference.

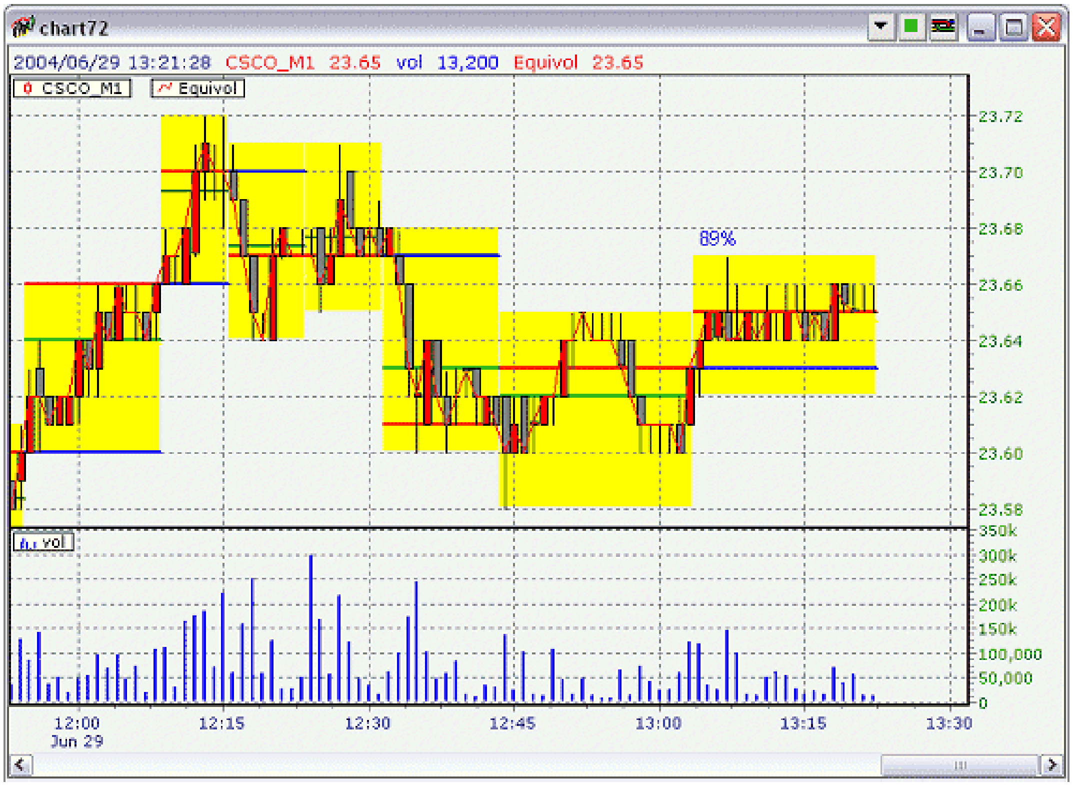
3 Select the CSCO data series, and add the EquiVolume Overlay indicator with the following parameters:

Style to **Volume**

Size to 1000000

Line Width to 2

The figure below shows an Equivolume chart with volume histogram as reference.



Reading the Equivolume Chart

Each yellow box represents a region where 1,000,000 shares are traded. The horizontal span is the time spent trading the volume. If you look at the volume histogram, you can see that regions with larger volume will have a narrower equivolume block. The horizontal span covers the high and low of the bars within a block.

The blue line is the open price of the box. The red line is the close price of the box. The green line is the average price of the box.

The last box is incomplete. It does not cover the number of shares yet. A percentage is shown for how much of the box is completed. In the chart above, it is 89%, which translates to 890000 shares traded.

Equivolume charts tell important support and resistance information. Take the average price for example (green line), it tells you that many trades are happening on average at that price. Because traders use their entry price as reference, you can expect the average price to be a strong support/resistance level. If you look at prices after a yellow block, you can see that they follow the support/resistance more often than not (in the above figure, the 4th, 5th, 6th, 7th blocks from the left).

Reference

Following is the parameters for equivolume.

Parameter	Meaning
Style	**Tick** or **Volume**. Whether the boxes are constructed by ticks or traded volume.
Size	Number of ticks or traded volume for a box.
Box Color	Color of the box
Open Color	Open line color
Close Color	Close line color
Avg Color	Average price color
VWAP Color	Volume weighted average price color
Line Width	Line width
Last Pct	Whether to show last box's completion percentage
Last Pct Size	Font size for the last percentage
Last Pct Color	Color for the last percentage

Historical Overlay

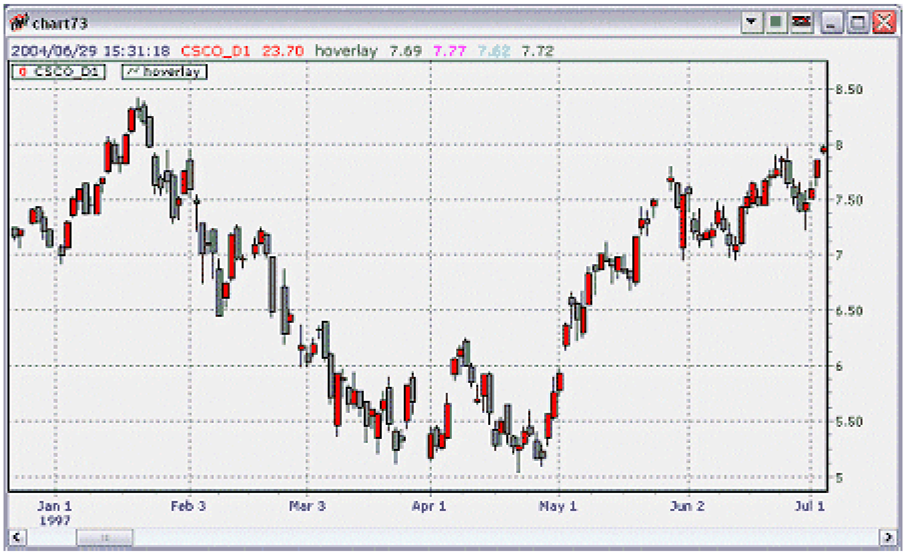
Historical overlay indicator overlaps historical data to see if certain pattern matches. It also provides projection into the future.

The hypothesis is when an instrument trades similarly to its history, history will repeat and there is certain validity to the projection.

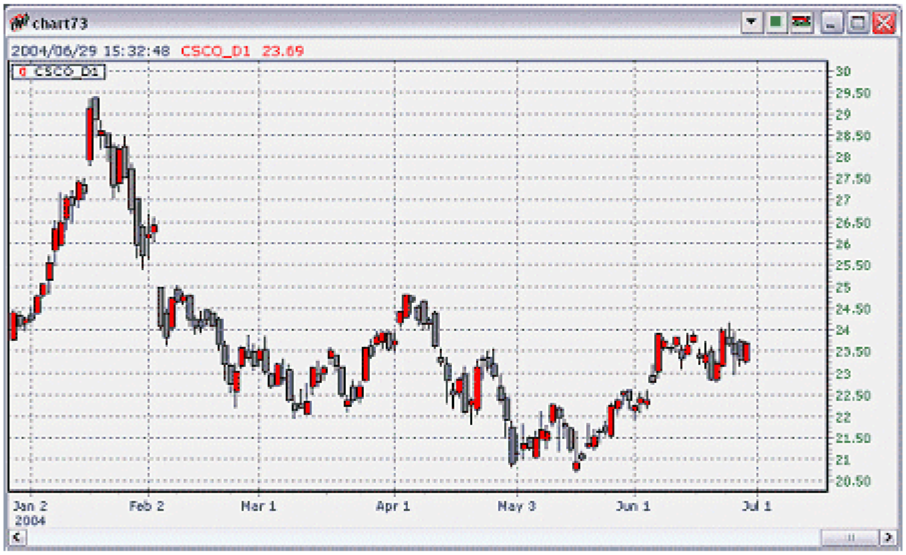
Example

Open a daily chart of CSCO. Make sure you load data to cover up to 1/1/1997 (use a large day to load setting).

Scroll the chart to the beginning of 1997.



Now scroll the chart to the beginning of 2004.



Notice the chart patterns for 1997 and 2004 are somewhat similar.

Historical overlay indicator will let us compare the two patterns.

Add historical overlay indicator to CSCO. Set the followings:

1 **Hist Start** to 1/1/1997

2 **Hist End** to 12/31/1997

3 **Target DateTime** to 1/1/2004

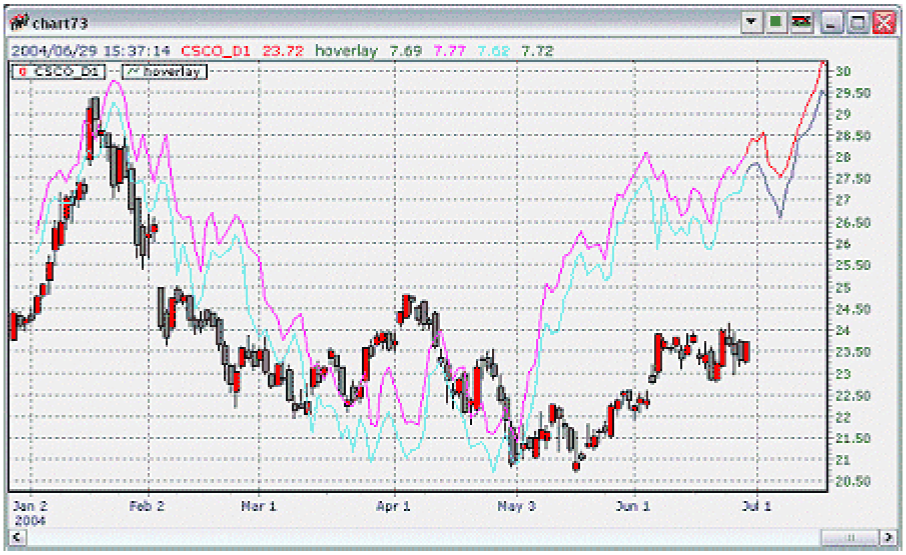
4 **Projection** to **High Low**

5 Leave other parameters to the default.

6 Press the **Vertical Scale** tab. And enable **Overlay**.

7 Press the **Apply** button.

This will overlay the historical data from 1/1/1997 to 1/1/2004, covering a one year period. The data of 1997 is shown as historical overlay, plotted as lines.



Note that we have enabled price level overlay because we are interested in the price pattern, not the absolute price itself.

The magenta and cyan lines are historical high and low values from 1997. They are plotted until the current trading day. The red and blue lines are historical high low, they are plotted after the current trading day and as as a high low projection.

If history repeats itself, we will expect CSCO to go up.

You can add more right side spacing for projection in chart manager, under **Visual 2** tab.

Reference

The following is the parameter reference for historical overlay.

Parameter	Meaning
Hist Start	Start of historical data to be overlaid
Hist End	End of historical data to be overlaid
Target DateTime	Target where **Hist Start** is mapped to
Inverse	If set to yes, the data is shown in its left right mirror image
Projection	Projection style can be set to **None**, **Close**, **High Low**
Proj Style	Projection drawing style, either **Line** or **Dot**.

Proj Width	Width of projection drawing
Proj Color 1	Color of the high or close projection line
Proj Color 2	Color of the low projection line

Pattern Matching Indicator

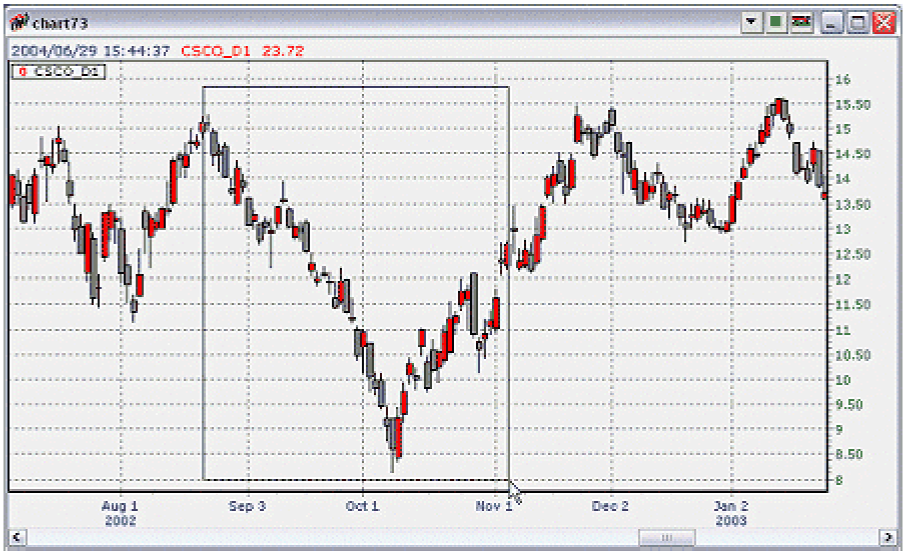
Pattern Matching indicator matches a given pattern with data and returns a confidence level score (from 0 to 100) as the output. By looking at the score, you can see how close the pattern matches with the data and make projection on the data behavior.

Pattern matching indicator uses the same pattern matching algorithm as the pattern scanner's visual pattern matching.

Example

In this example, we will be matching V shape reversal pattern in a CSCO daily chart.

First, we will need to find a pattern to match. Open a CSCO daily chart. Set days to load to a large number to cover August 2002. We can see from the chart below that a V shape reversal happens in the period Aug 22, 2002 to Dec 2, 2002.

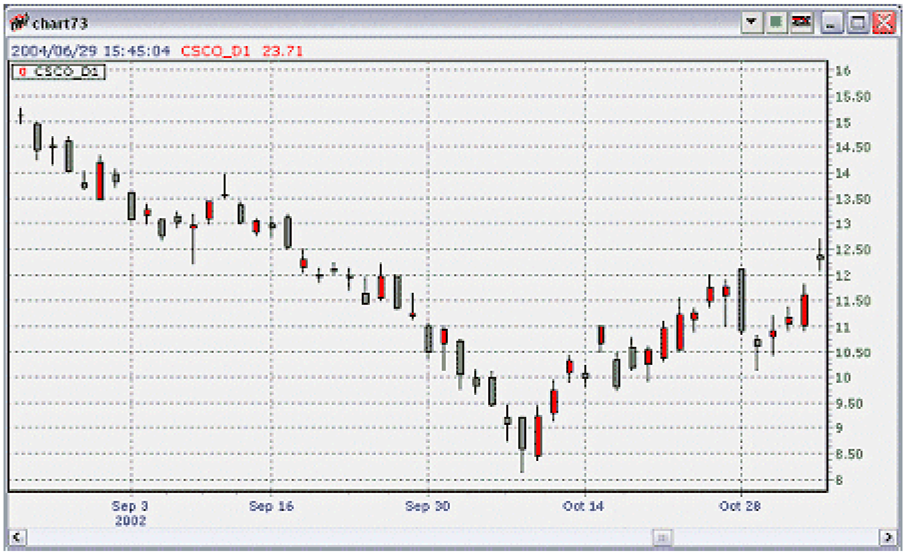


The next step is to mark the pattern for matching, Left click and hold the mouse button at around Aug 22 at $15.50, then drag the mouse to around Nov 5 at $8. Then release the mouse button. This will zoom into the region of the pattern we want to match.

Note that we do not include the complete Aug 22 to Dec 2 period. The reason is we want to match the early stage of a V shape reversal so we can buy into the market before the final rise in the V shape. If you match the complete V shape pattern, by the time you find the pattern, the V shape is already finish, and there is no point buying.

The chart before is zoomed into the pattern region to be matched.

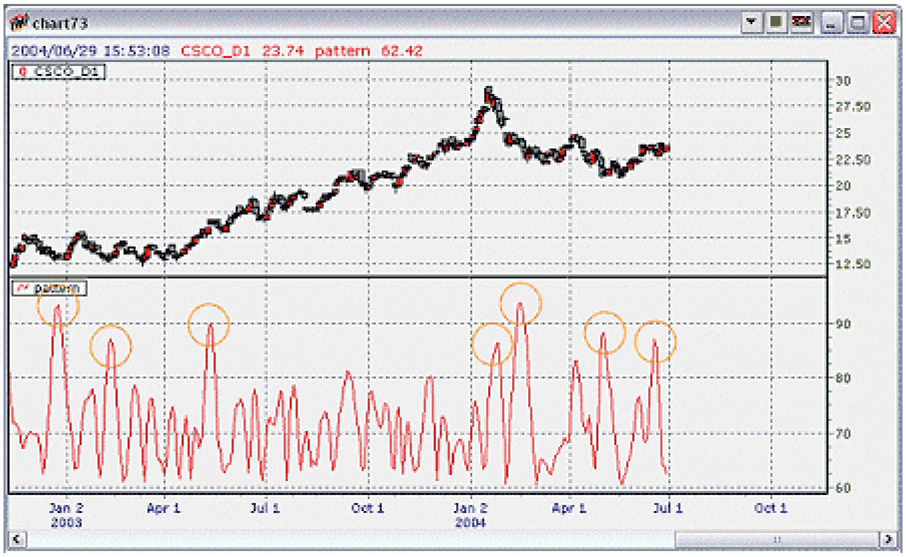
For reliable pattern matching, it is advised you use a pattern is at least 20 bars long.



The next step is saving the pattern, right click on the chart to open pop up menu, and choose **Save Pattern**. Save in a file name you can remember, for example, `vshape`.

Then select CSCO in the chart, right click to open pop up menu and choose **Add Indicator**. Choose the Pattern Matching indicator. For the parameter **Pattern File**, enter the file name you just saved, e.g. `vshape`. Press the **Apply** button to add the indicator. You may want to move the indicator to another pane for better visibility.

The following chart shows the pattern indicator, with the chart zoomed out.



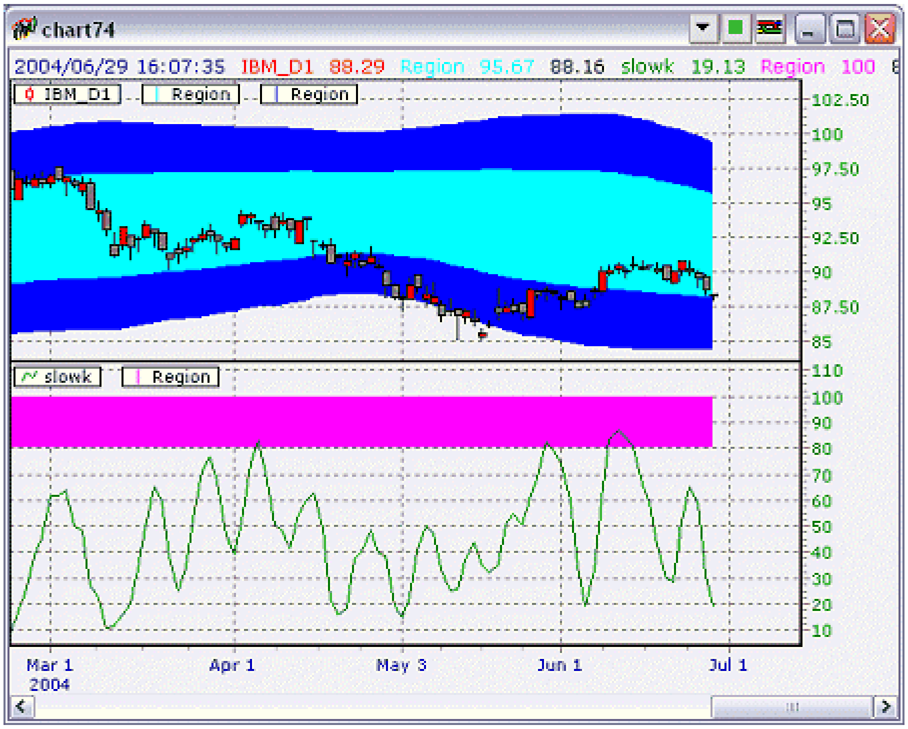
Notice that the vshape pattern is matched at 100% at the place we marked the pattern. What is more interesting is to the right hand side of the pattern (near Jan 1, 2003), we have a match that is over 90%. This indicates another V shape reversal happening. If you buy the market at the 90% matched, you will be making money as the V shape reversal reaches its high.

We've also marked several V shape reversal in the chart. Most have successfully mark a reversal. However, it is possible for a V shape reversal to fail. In which case, you should use money management technique to get out of the trade.

Region Coloring

Overview

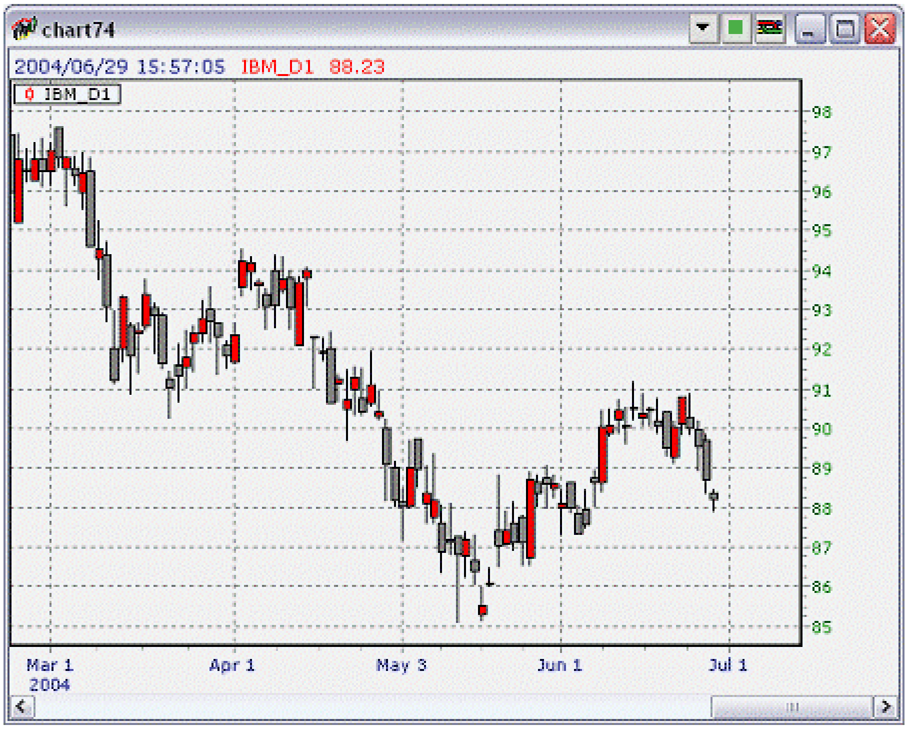
Region Coloring plots a surface bounded by the parameter formulas. Region Coloring is useful for highlighting area of interest while many indicator lines are already in use, thus you need another presentation method to add more information to a chart with better clarity.



Use Region Coloring as a Dynamic Band

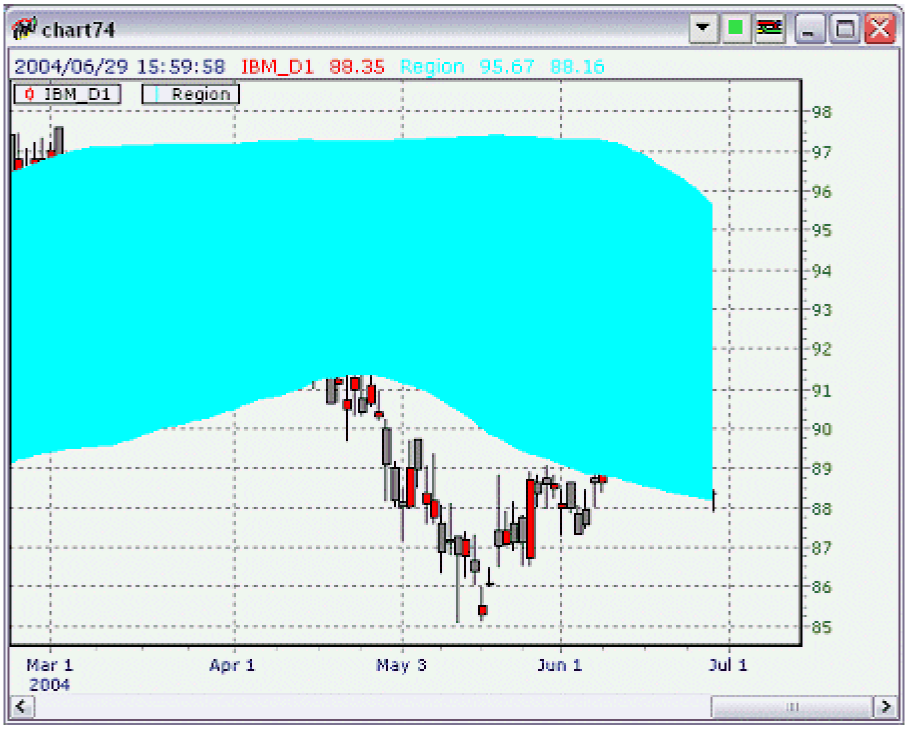
Lets walk through an example.

Create a new chart.

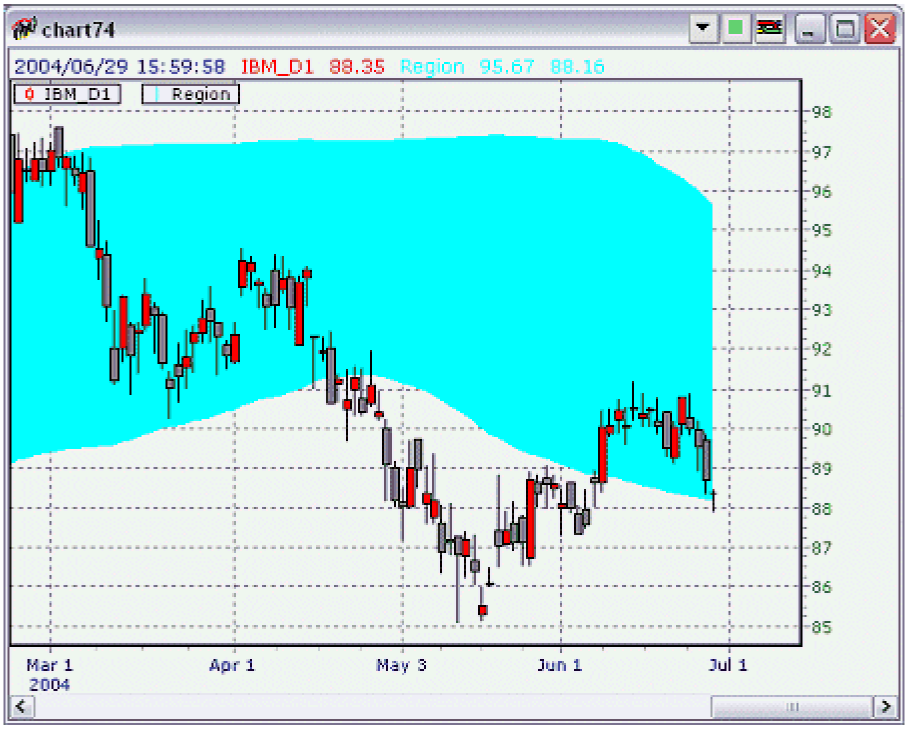


Add indicator Region Coloring and set the parameters as follows and change the color to cyan.

Parameter	Formula
High	`bband (data1, 100, 1)`
Low	`bband (data1, 100, -1)`

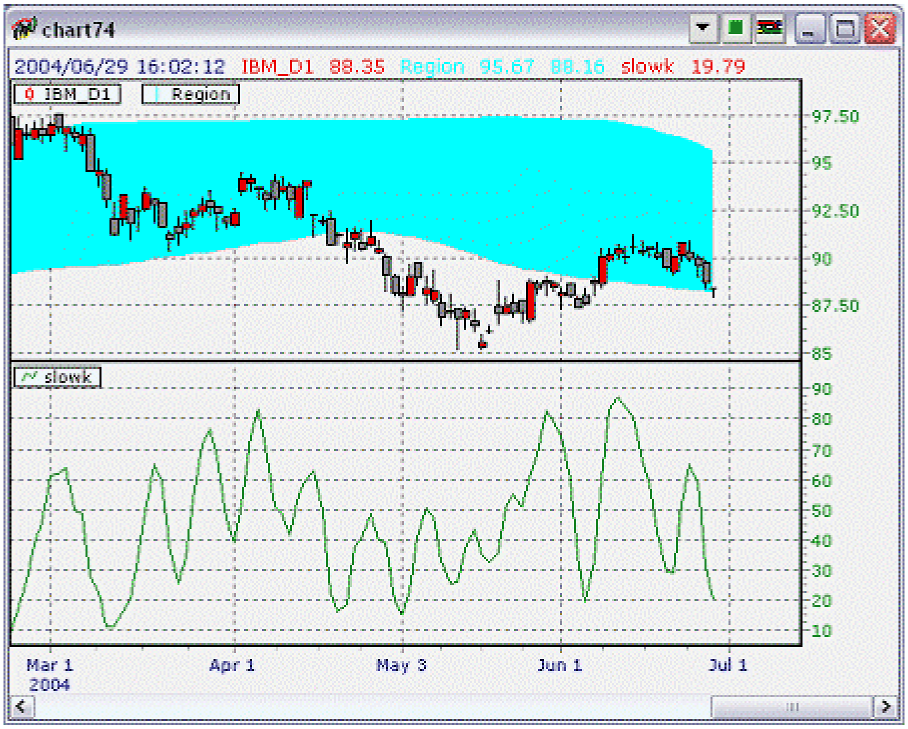


Now set the display order of the Region Coloring indicator after the IBM data by choosing **Object Ordering** from the chart's pop up menu. Adjust the ordering so that Region Coloring is displayed below the data.



Use Region Coloring to Highlight Constant Area

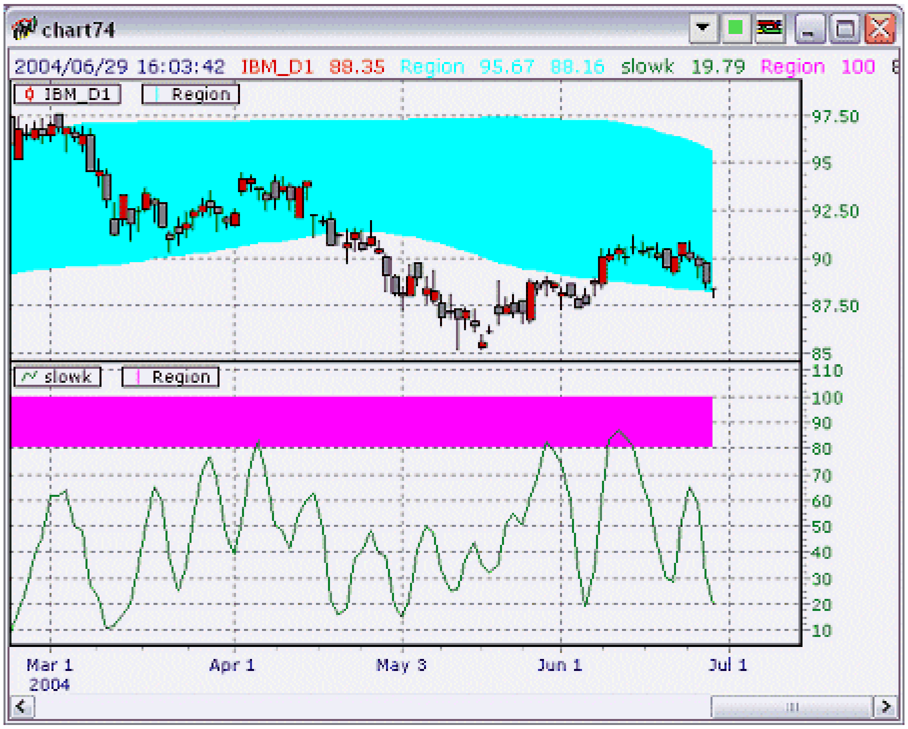
Following the same example, lets add the SlowK indicator to the chart.



Say you want to highlight the overbought area 80 to 100 such that you can spot the condition easily. Add the Region Coloring indicator and set the parameters as follows and change the color to purple.

Parameter	Formula
High	`100`
Low	`80`

Adjust object ordering to show the constant area behind SlowK.



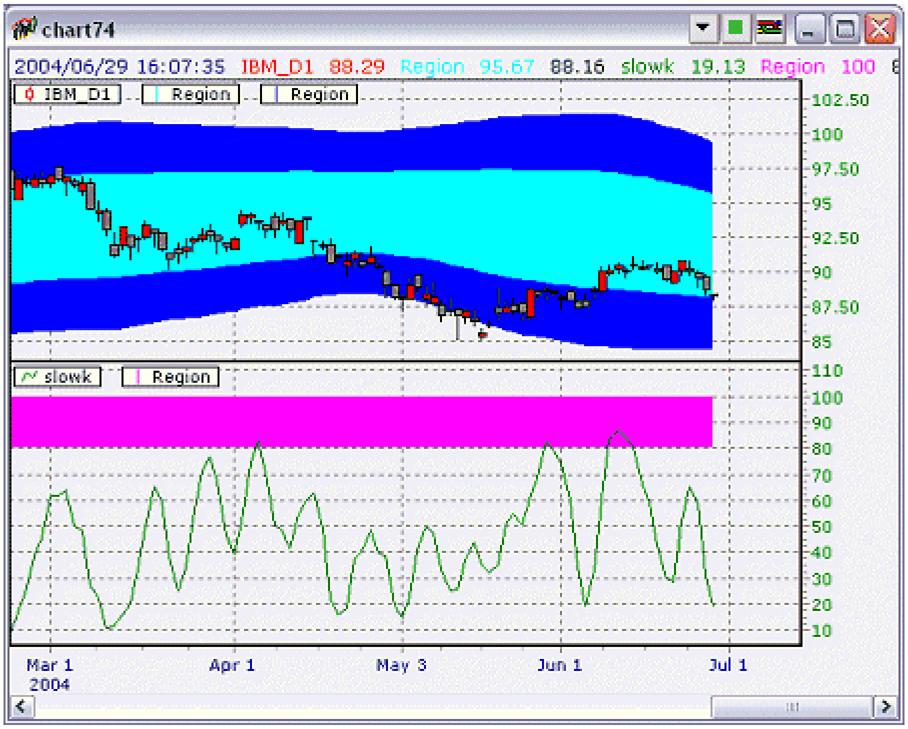
Overlay Multiple Region Coloring Together

Now, say you are interested in overlaying another set of Bollinger band on top of the current set.

Add the indicator Region Coloring with the following parameters,

Parameter	Formula
High	`bband (data1, 100, 2)`
Low	`bband (data1, 100, -2)`

And modify the object order accordingly.



Relative Strength Multiple Data

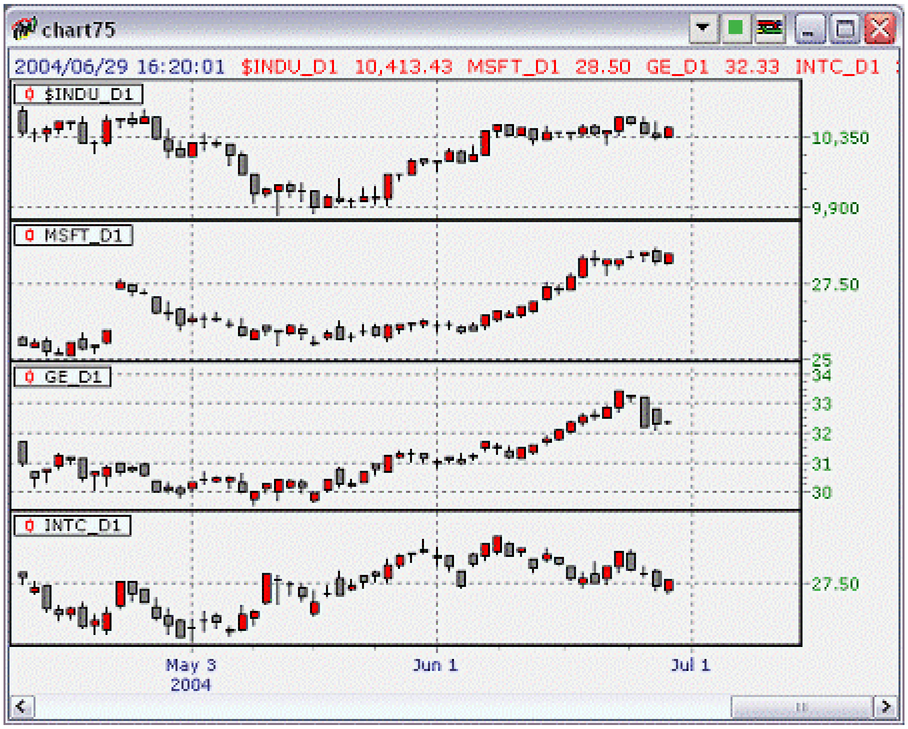
Relative Strength Multiple Data indicator can be used for displaying multiple data series in terms of percentage change or absolute change from a user defined starting time. Each relative strength plot will then reference to their own individual starting price information to compute the changes.

Example

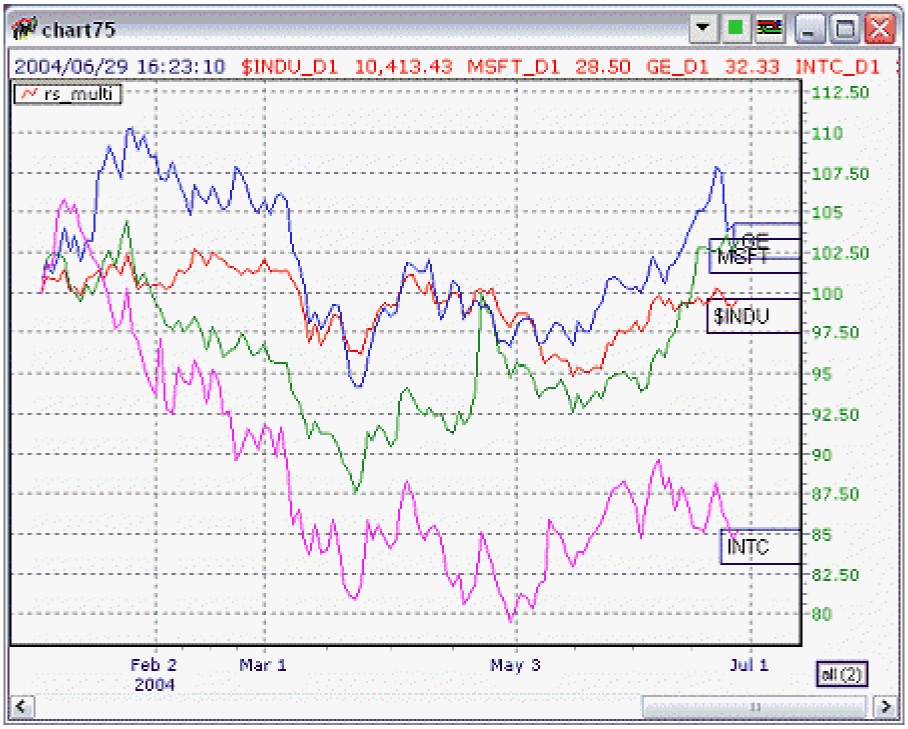
Lets say you are interested in charting the relative strength of 4 symbols, Dow Jones Industrial Average, MSFT, GE, INTC.

Symbol for Dow Jones Industrial Average is data feed dependent.

Create a new chart. Then add the 4 data series to the chart using daily data.



Now add the Relative Strength Multiple Data indicator and set the date to a common date, e.g. Jan 1, 2004. Double click on the pane that contains the indicator to switch to single pane mode for better viewing.



Options

If you need to plot more than 10 relative strength plots, you can simply add more Relative Strength Multiple Data indicator and set the **First Series** parameter to instruct the indicator to plot from which data series. Setting of 1 is the default value that means you are plotting from the first data series to the tenth one. So you can set the First Series parameter to 11 such that you can plot from the eleventh data series to the 20th one.

If you would like to turn on/off a particular plot, simply go to the **Visual** tab in the indicator setup window to toggle the visibility directly.

The label at the right can be turned off by the **Label** parameter.

To change the starting date use the **Ref DateTime** parameter. There is a drop down **Date Time** tool to assist you entering the date time.

If you are plotting real time data, you probably want to control which price to use as the reference price. You can set the **Ref Price** parameter to either **Open**, which means using the first available open price on or after **Ref DateTime**, or, **Close,** which means the first available closing price.

Advantages

Relative Strength Multiple Data indicator has many advantages over other simple relative strength chart:

- You can easily plot as many relative strength plots as you want with one central switching control.
- You can apply indicators onto the relative strength plots to further your analysis.
- You can plot data of different time frame together in the same relative strength study.

Simply experiment with the indicator to get familiar with this powerful tool.

Simple Index

Constructing your own index has many advantages over standard index provided by the exchange. For example, custom index often provide a high quality and faster data, especially at low time frames.

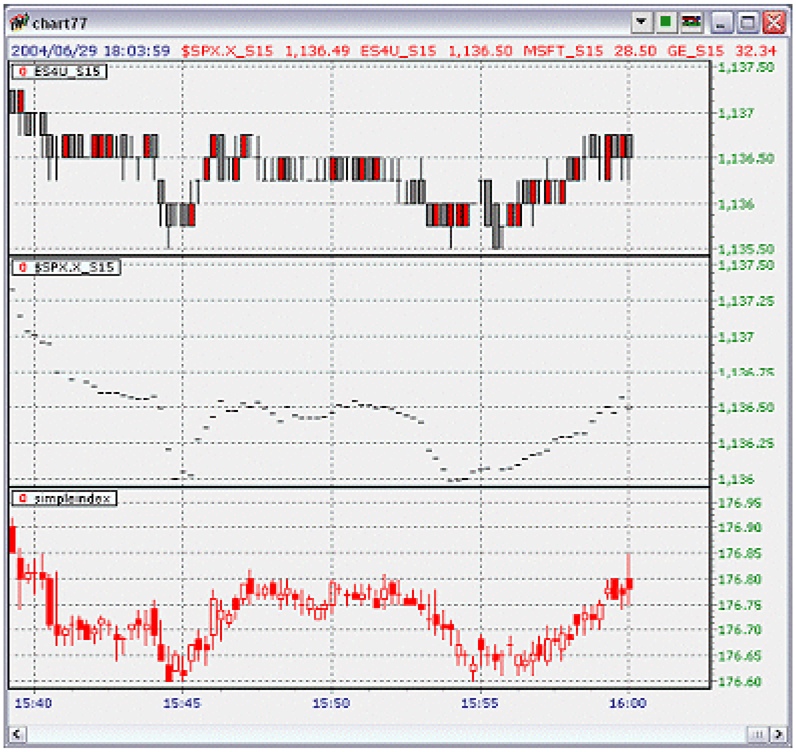
One way to construct custom index is to use the Weighted Index indicator (see ***Weighted Index*** (on page 1359)). Weighted index allows you to enter weights to individual symbols in the index. If you only need uniform weighting, i.e. weight of 1 to every symbol, you can use Simple Index discussed in this section.

Example

This example uses symbols that are data vendor dependent. You should make appropriate adjustment to symbols when you follow the example.

1. Open a chart, add a 15-second emini S&P data series, e.g. `ES U4`.
2. In a new pane, add a 15-second S&P cash index, e.g. `$SPX`.
3. Create a third pane, add 15-second `GE, IBM, INTC, MSFT` to the pane. Use chart manager to turn `GE, IBM, INTC, MSFT`'s visibility to off.
4. Add Simple Index indicator, make sure you link the indicator to one of the stocks. Set indicator parameter **Exclude Symbol** to `$SPX, "ES U4"`
5. Under the **Visual** tab, set **Style** to **Candle**.

Here is what the chart would look like:



As shown in the chart, Simple Index provides faster response than the e-mini contract. It also provides much high quality data than the cash index in a 15-second time frame.

Reference

Simple Index uses all data series in the chart to construct a index by adding all values of the data series. The parameter **Exclude Symbol** is provided to exclude symbols from the index. You can enter multiple symbols by separating the symbols by comma or space. If a symbol to be excluded contains space (e.g. `ES U4`), you must put the symbol in quotes (e.g. `"ES U4"`).

Tick Precise Volume Profile

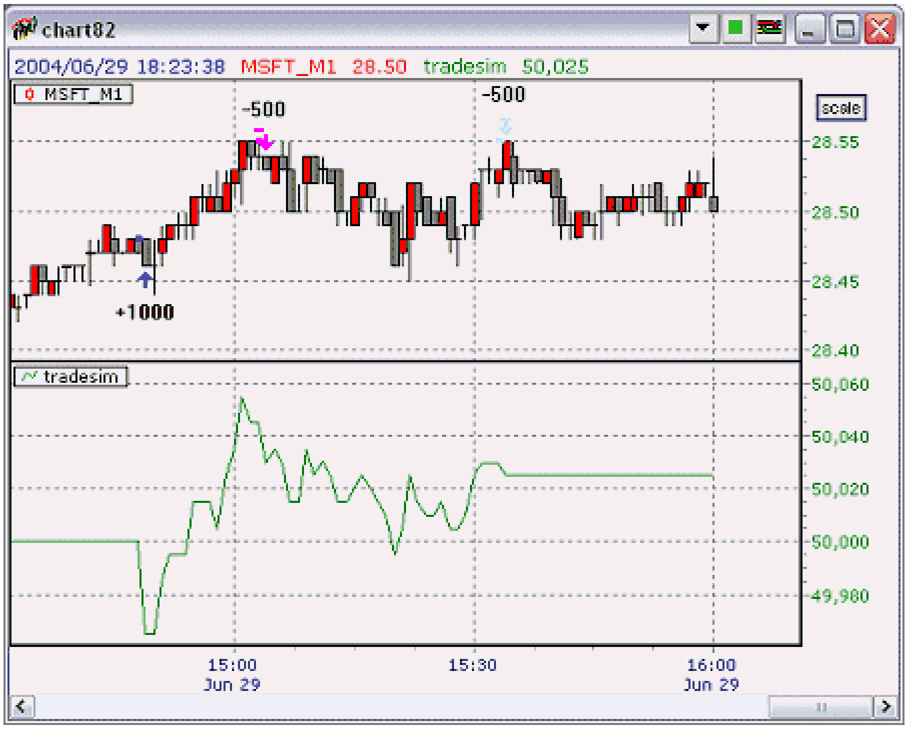
To be written.

Trade Simulator Indicator

Trade Simulator indicator lets you reconstruct an equity curve from trades entered in Trade Simulator. For more information about trade simulator, see ***Trade Simulator*** (on page 755).

The following chart have 3 trades entered from Trade Simulator, and the Trade Simulator Indicator added to the data.

- Bought 1000 shares of MSFT
- Sold 500 shares of MSFT
- Sold 500 shares of MSFT



As you can see, the equity curve is reconstructed from the trades and the data.

Trade Simulator is a full trading system. You can enter trading system information such as initial capital, interest rate, margin requirement and such under the **System** tab when you edit the indicator. Once these information are available, you can open System Performance Viewer by selecting Trade Simulator indicator, and choose **Trading System>Open Performance Viewer** from the pop up menu. Advance system reporting facilities are available in System Performance Viewer.

For more information about System Performance Viewer, see ***System Performance Viewer*** (on page 951).

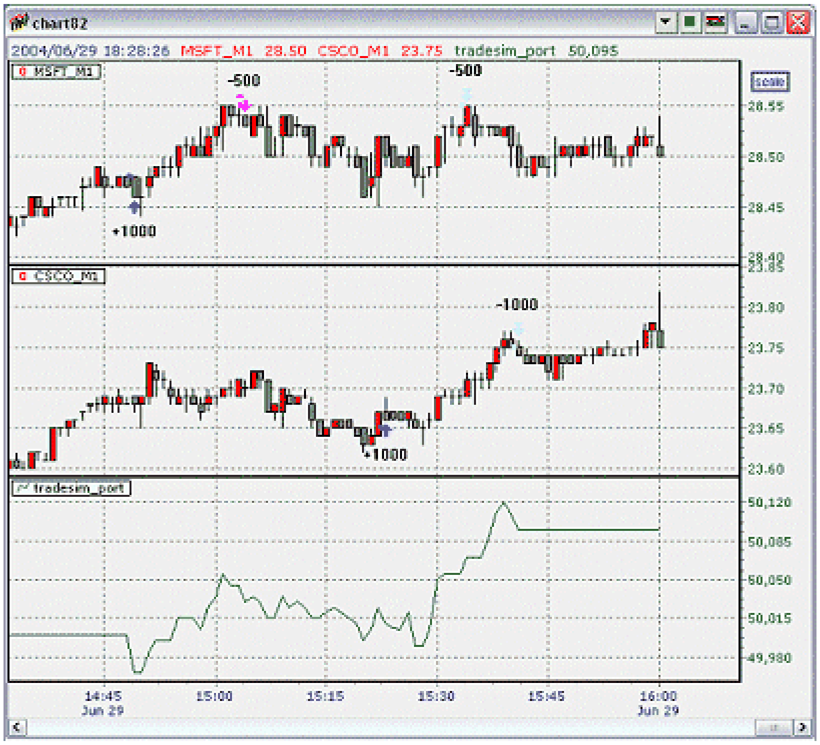
Trade Simulator Portfolio Indicator

Trade Simulator Portfolio indicator is the portfolio version of Trade Simulator Indicator ***(see Trade Simulator Indi***cator (on page 1341)). It generates equity curve based on a trades you enter in Trade Simulator for multiple symbols.

The following trades have been entered in Trade Simulator:

- Bought 1000 shares of MSFT
- Sold 500 shares of MSFT
- Sold 500 shares of MSFT
- Bought 1000 shares of CSCO
- Sold 1000 shares of CSCO

The following chart has MSFT, CSCO and Trade Simulator Portfolio indicator in it.



Trade Simulator Portfolio has generate the equity curve for the trades.

Trade Simulator Portfolio is a full trading system. You can enter trading system information such as initial capital, interest rate, margin requirement and such under the **System** tab when you edit the indicator. Once these information are available, you can open System Performance Viewer by selecting Trade Simulator indicator, and choose **Trading System>Open Performance Viewer** from the pop up menu. Advance system reporting facilities are available in System Performance Viewer.

For more information about System Performance Viewer, see ***System Performance Viewer*** (on page 951).

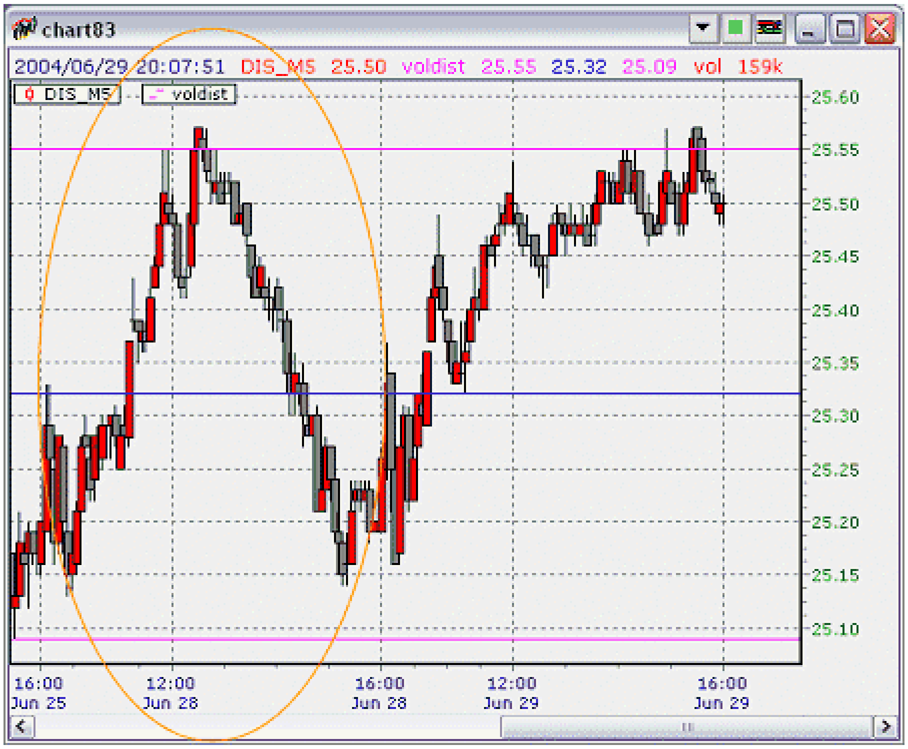
Volume Distribution

Overview

Volume Distribution indicator encapsulates the power of the volume distribution chart and provides new flexibility into the usage of the concept of volume distribution.

Two models are provided.

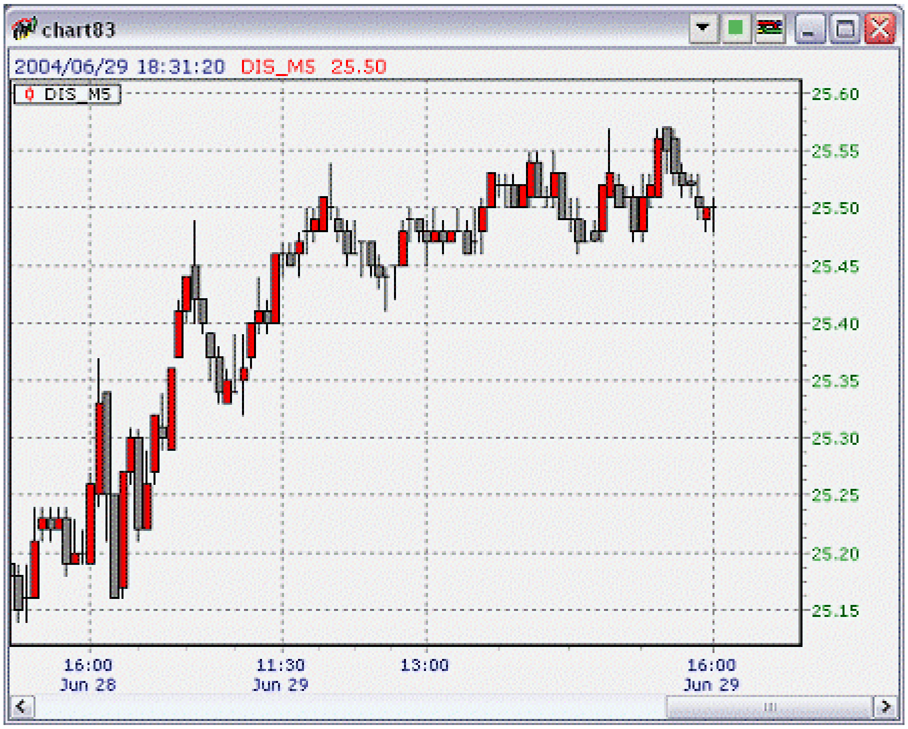
- Normal Model provides classic usage of finding price level within the historical data.
- Gaussian Model provides advanced usage of projecting price level outside of the historical data.



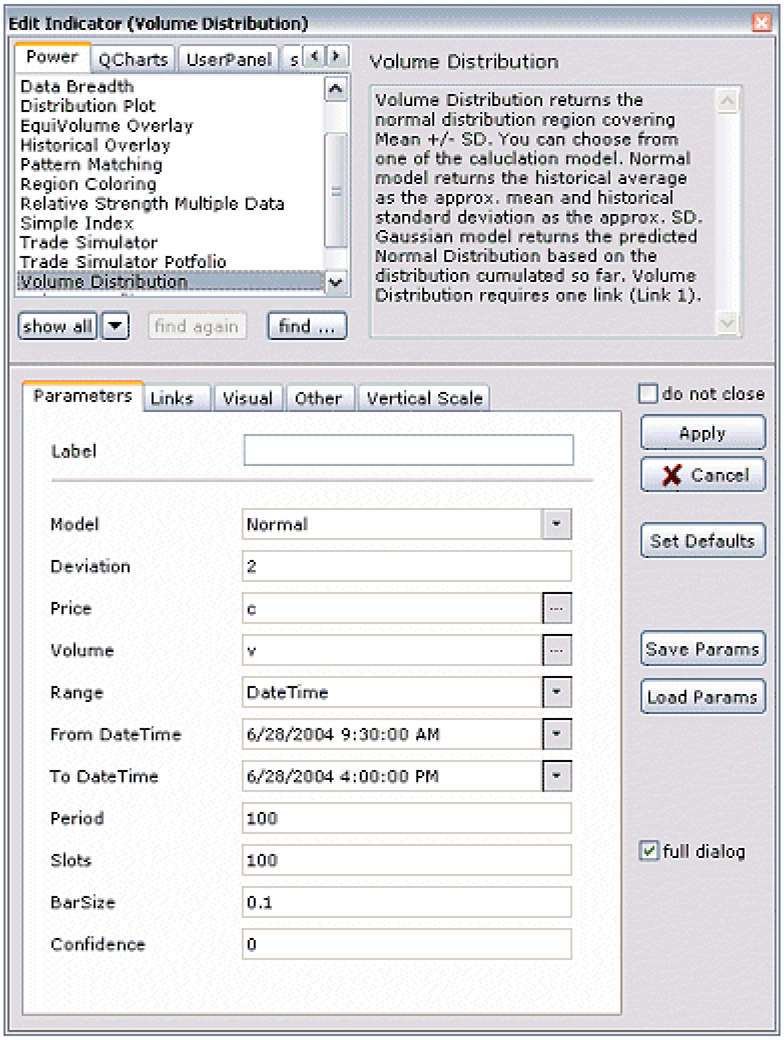
Using the Indicator

Lets go through an example here.

Create a chart of DIS 5-minute bars with 5 days of data.



Now add the volume distribution indicator with parameters as follows,



Volume Distribution

Volume Distribution returns the normal distribution region covering Mean +/- SD. You can choose from one of the caluclation model. Normal model returns the historical average as the approx. mean and historical standard deviation as the approx. SD. Gaussian model returns the predicted Normal Distribution based on the distribution cumulated so far. Volume Distribution requires one link (Link 1).

Label	
Model	Normal
Deviation	2
Price	c
Volume	v
Range	DateTime
From DateTime	6/28/2004 9:30:00 AM
To DateTime	6/28/2004 4:00:00 PM
Period	100
Slots	100
BarSize	0.1
Confidence	0

Parameter	Value
Model	Gaussian

Deviation	2
Price	c
Volume	v
Range	DateTime
From DateTime	Open time of last trading day. If today is 6/29/2004, set this to 6/28/2004 9:30:00am
To DateTime	Closing time of last trading day. If today is 6/29/2004, set this to 6/28/2004 4:00:00pm
Period	100
Slots	100
BarSize	0.1
Confidence	0

You can now see the volume distribution indicator.

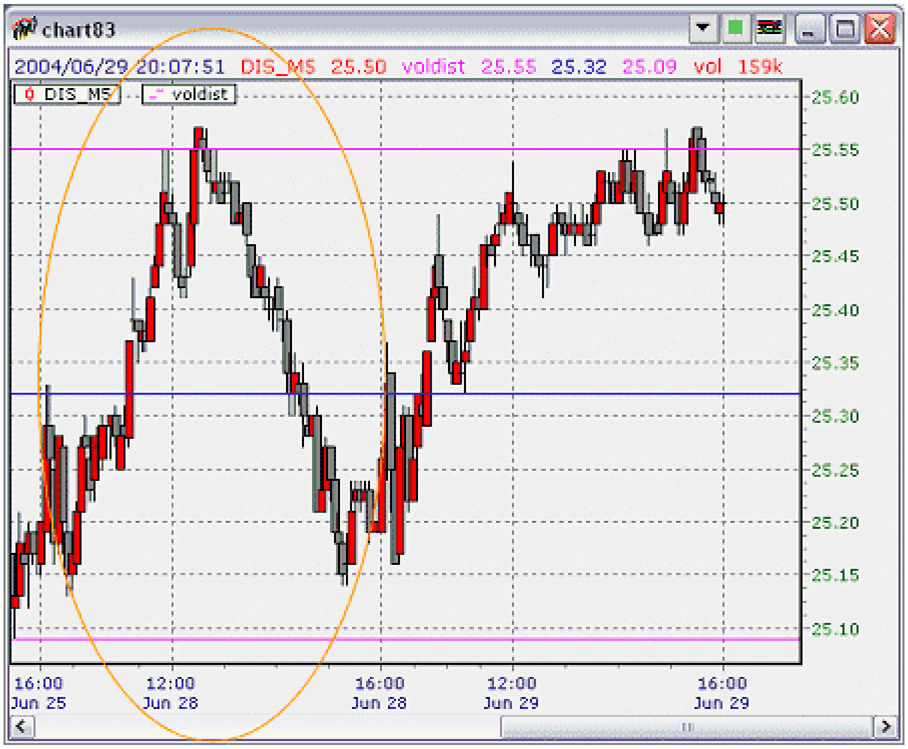


chart83
2004/06/29 20:07:51 DIS_M5 25.50 voldist 25.55 25.32 25.09 vol 159k
DIS_M5
voldist
25.60
25.55
25.50
25.45
25.40
25.35
25.30
25.25
25.20
25.15
25.10
16:00
Jun 25
12:00
Jun 28
16:00
Jun 28
12:00
Jun 29
16:00
Jun 29

Volume Distribution indicator is plotting the projected mean and deviation from the volume data from the last trading day. In the example above, we have circle the area in orange. Volume Distribution is available from the start of the next trading day. In this example, we see that the next trading day's price do not move significantly away from the projected deviation.

Parameter Settings

Model

You can choose between Normal and Gaussian.

Normal model is simply the computation of the volume weighted average of the prices and the related standard deviation of the prices in use.

Gaussian model is the fitting of the normal distribution curve onto the sampled price-volume distribution so far and project if there exists a normal distribution for these data. The result of this model is different from the one generated by the Normal model.

Deviation

A multiplying factor used for plotting the upper and lower band of this indicator.

Upper Band = Mid + SD * Deviation

Mid = Computed Mean

Lower Band = Mid – SD * Deviation

SD is the computer standard deviation.

Price

A formula for calculating the price to be fitted into model computation. The default formula is “c” which is the closing price of a bar. You can apply any formula you like.

Volume

A formula for calculating the volume to be fitted into model computation. The default formula is "v" which is the volume of a bar. You can set the formula to “1” which will then equal weights all the prices. You can also set the formula to anything you want.

Range

It can be either DateTime or Period.

If you choose DateTime, then the parameters From Datetime and To Datetime are used to specify the time range to collect data. You can enter date or date time to these two (2) parameters.

If you choose Period, then the parameter Period will be used to determine the number of periods to look back for the model computation.

Slots

Used with Gaussian model only.

Slots specify the number of slots to allocate on top (and bottom) of the first price used in calculation.

The number of slots works together with the parameter BarSize to define the price range that you can use in model computation.

For example, with the BarSize of 0.1 and Slots of 100, you can map a range of –10 to +10 from the first price point.

Each slot will be responsible for holding and accumulating the amount of volume that happens in the respective price range.

BarSize

Used with Gaussian model only.

The range of value that each slot represents.

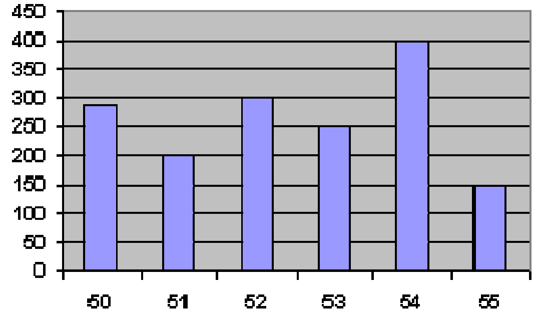
Confidence

Used with Gaussian model only.

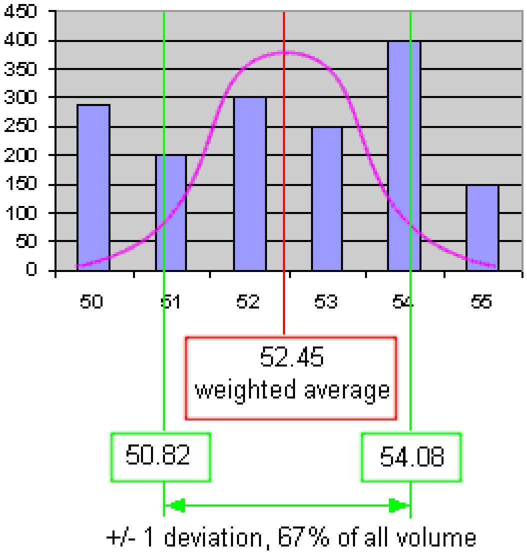
Confidence ranges from 0 to 1. When confidence is 1, you need an exact fit of the data to the Normal Distribution to get calculated results from the Gaussian model. When confidence is set to 0, as long as there is some results computed from the model, it will be returned for your use.

An Explanation of the Normal Model

Given some price data with related volume information, we have a distribution of price.



By adding up these values weighted by their volume, you can figure out the volume weighted average and the sample standard deviation.

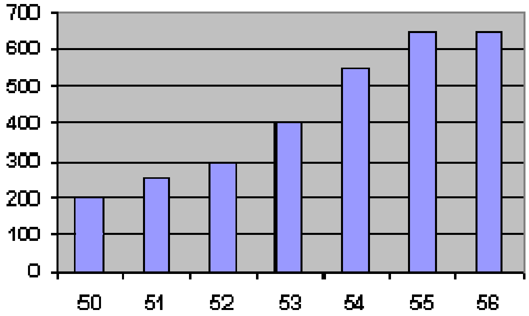


Using this model you can locate the area where most of the volume took place historically. It does not have any extra information beyond stating what had happened to the data.

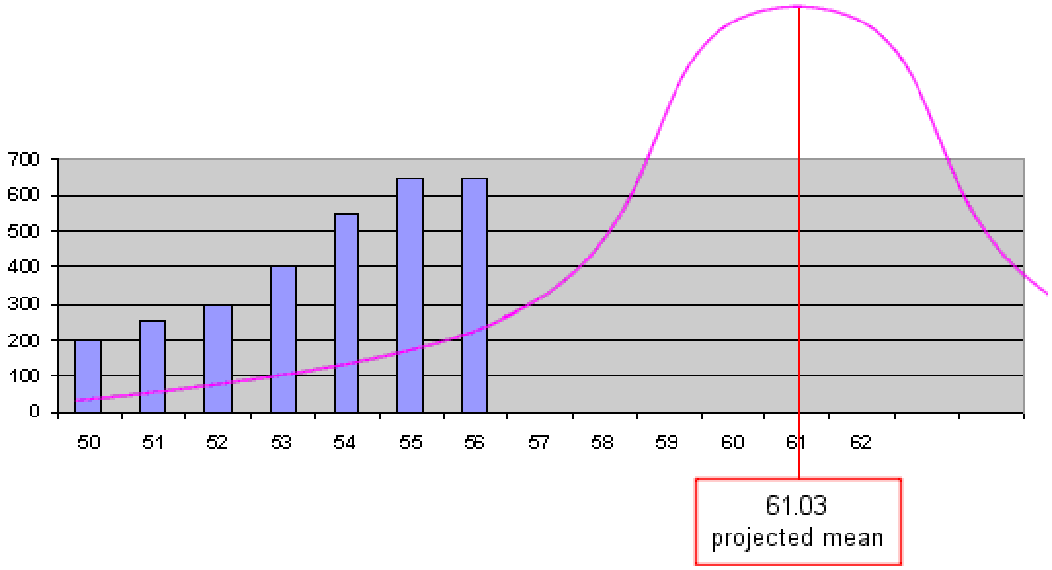
This model is useful if you are looking for support / resistance area within the data.

An Explanation of the Gaussian Model

Given some price data with related volume information, we have a distribution of price.



By fitting the normal distribution function (bell curve) onto the sample distribution, you can generate a modified version of the bell curve which will provide you with projected mean and deviation values.



Using this model you can project the area where most of the volume will take place. Similar to any projections or forecast tools, the projected values should be used as a reference only.

This model is useful if you are looking for support / resistance area outside of the historical data.

Volume Profile

Volume Profile Indicator plots traded volume at different price levels. It is a generalized version of Volume Distribution.

A profile is a statistical distribution chart based on volume. Each slot (horizontal bar) in a profile covers a price range. The slot is incremented by volume of a data bar if the price of the data bar is covered by this slot.

Volume Profile offers the follow benefits:

- Volume Profile works on all time frames, including tick chart. In a 1-tick chart, Volume Profile works like Volume Distribution.
- Price and volume can be modified using formula. For example, you can modify the indicator to use the value 1 instead of the actual volume. In this case Volume Profile will display occurrence profile.
- A formula-based filter is provided. For example, you can only include bars with a large traded volume in the profile.

Example - Monthly Volume Profile for CSCO

1 Create a chart using CSCO daily data.

2 From chart's pop up menu, choose **Add Indicator**. Select Volume Profile indicator from the **All** tab to apply to CSCO data series. The following parameters will create a volume profile for every $0.05 movement. Each profile covers a month. Price that has volume slots within 1 standard deviation will be colored in red. The volume weighted mean price (point of control) will be colored in yellow.

Price - leave this blank

Volume - v

Filter - true

Slot Size - 0.05

Every N Bars - Monthly

Value Area - On

VA Std Dev - 1

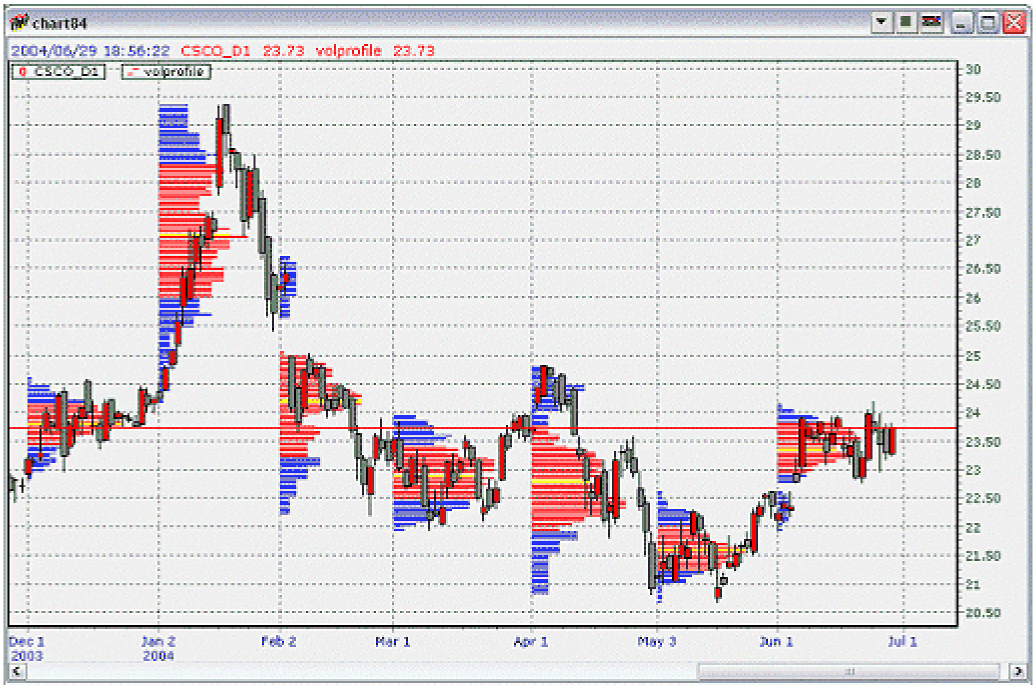
Max Vol - 0

Bar Color - Blue

VA Color - Red

POC Color - Yellow

Here is the chart:



In the chart, based on the December profile, we can see that CSCO has a large amount of shares traded at $23.75. This tells us that a support level has been formed at this price. In January, CSCO continues to go up and form another support level at around $27.00. A possible strategy is shorting CSCO if it breaks below $27.00, and covering at $23.75.

Parameter Reference

Price - If this parameter is blank (default), then profile calculation includes the low to high value of a data bar (i.e. slots that cover prices from low to high will be incremented). You can use a formula for Price. For example, if you use `(H+L+C)/3`, the slot that covers `(H+L+C)/3` will be incremented

Volume - If this parameter is `v` (default), then each slot is incremented by the actual volume when the price is covered by the slot. You can use a formula for volume. If this parameter is 1, then each slot is incremented by 1 when the price is covered by the slot (occurrence profile).

Filter - If this parameter is true (default), then all bars are included in profile calculation. You can use a formula for filter. For example, if you set filter to `V>1000`, then only bars with a volume larger than 1000 will be included in profile calculation.

Slot Size - Slot size of profile. For example, for stocks a value of 0.05 produces a slot for every $0.05 movement (stock). For S&P e-mini, a value of 0.25 produces a slot for every quarter point.

Every N Bars - If this parameter is a positive number, a profile is created for every N data bars. If this parameter is a negative number, a single profile is created, counting from the current bar. Negative number profile will moves with the data series as the data series is updated in real-time. This parameter also recognizes the following profile period: **All**, **Daily**, **Weekly**, **Monthly**, **Yearly**. For **All**, a single profile is created for all bars in the chart. For **Daily**, a profile is created every day. For **Weekly**, a profile is created every week, and so on.

Special Break - None or daily. This parameter controls whether volume profiling will break at a certain interval to start a new profile.

Setting a daily break is useful in situations where there are not enough bars to create a profile. For example, if you are profiling from 9:30 to 4:00, 39 10-minute bars, there will be enough bars on a regular trading day to create a profile. On a partial trading day however, there will not be enough bars and the current profile will continue with the bars on the next trading day. If you do not agree with this behavior, you can set **Special Break** to daily. Volume Profile will then stop at day end and create a new profile on the next trading day.

Starting Pos - First Bar or Last Bar. This parameter controls where the profile is drawn.

By default, profile is drawn at the position of the first bar of data that create the profile. The profile will visually cover the area where the bars that create the profile are located.

If you set **Starting Pos** to Last Bar, profile is drawn at the projected last bar position. Drawing at last bar is useful when you want to line up the profile with higher time frame data. For example, if your chart contains 1-minute and 30-minute data series and you are profiling 30 1-minute bars, by setting **Starting Pos** to Last Bar, you will make the profile lining up to the right side of the 30-minute bars.

Value Area - If on (default), then slots within a specified standard deviation will be colored differently. Also, the slot that is at the point of control is also colored differently.

VA Std Dev - # of standard deviation that will be colored differently if Value Area is on. Default is 1.

Max Vol - this parameter is used to scaled the profile horizontally. Each profile has a horizontal space for displaying the slots (about 2/3 of the horizontal space covered by the data bars). A value of 0 (default) indicates relative spacing, i.e. each profile is scaled to cover the available space. A non-0 value indicates absolute spacing, i.e. all profiles are scaled such that the the right most point of the profile is at the specified value (in volume).

Bar Color - default color for the slots.

VA Color - color for slots that are within the specified standard deviation. This parameter is applicable only if **Value Area** is on.

POC Color - color for the slot that covers the Point of Control. Point of Control is the mean volume weighted price. Thus it is always at the middle between the standard deviations. This parameter is applicable only if **Value Area** is on.

Weighted Index

Weighted Index takes a list of symbols and their weighting and returns a virtual data series that has full open, high, low, close data for use with regular indicators.

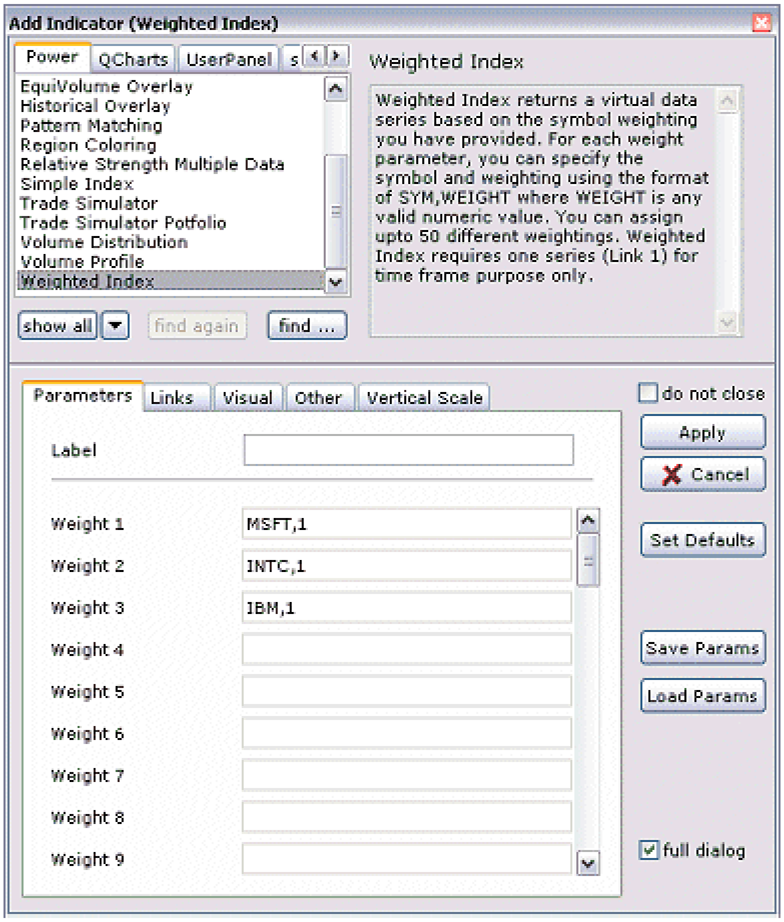
Weighted Index requires the member symbols to be present in the chart.

For basic workflow using Weighted Index, read the section ***Tutorial: Introduction to Power Indicators*** (on page 223).

If you prefer a non-weighted index, i.e. weight of 1 to every symbol, you can use Simple Index instead. See ***Simple Index*** (on page 1339).

You can specify up to 50 symbols at a time.

To specify weighting for a particular symbol, simply open the indicator setup window and add the weighting to one of the **Weight** parameters. The format is `Symbol,Weight` where the `Weight` is any valid number, including negative values.



Weighted Index

Weighted Index returns a virtual data series based on the symbol weighting you have provided. For each weight parameter, you can specify the symbol and weighting using the format of SYM,WEIGHT where WEIGHT is any valid numeric value. You can assign upto 50 different weightings. Weighted Index requires one series (Link 1) for time frame purpose only.

Label	
Weight 1	MSFT,1
Weight 2	INTC,1
Weight 3	IBM,1
Weight 4	
Weight 5	
Weight 6	
Weight 7	
Weight 8	
Weight 9	

☐ do not close

☑ full dialog

To combine more than 50 symbols, you can simply use two weighted index to map 50 symbols each. Then apply a formula2 indicator to take the average of the two.

Since Weighted Index requires using specific symbols and their weights in the parameters, you have to ensure that the necessary data are added into the chart for correct calculation.

If a requested symbol is not available in the chart, that particular data will not be used in the index calculation and the total weighting will take this into account and ignore the weight of this particular symbol.

To temporarily disable a particular symbol from being used in the calculation, you can change the weight parameter for that symbol to a symbol that does not exist in the chart.

For example, you originally have an entry “MSFT,1.231” and you would like to disable it temporarily, then modify the entry to “X_MSFT,1.231”. This way, your weighting information is preserved and when you want to use the original symbol again, you can do so by just removing the extra characters.

When you switch datafeed or data source, remember that you may have to adjust the symbol names as well due to difference in naming convention by different data vendors.

For example, you have the stock symbol “AA” used in one of the weight parameters, using the Internet Stock data source. Then if you modify the data source to your real-time data vendor say Quote.com, then the symbol needed is “AMEX:AA” as Quote.com do not recognize the symbol “AA” as is.

When applying other indicators on the weighted index, remember to set the Link Field in the Indicator Setup Window to the appropriate field that you want. NeoTicker® can recognize a virtual data series and would automatically switch to list the Link Fields as Open, High, Low, Close for you such that you can apply indicators to virtual data series easier.

For example, when you are applying the stochastic FastK to a weighted index. Normally you would want to set the Link Field to Close of a data series. In this case, you would want to do the same on the weighted index as well, thus you should change the Link Field to “Close – P4”.

Weighted Index has many usages. You can use it as a summary data series for the basket of data you are tracking and analyze it as an overview of the scenario instead of analyzing each issues individually.

You can use it as a confirmation tool comparing to existing indexes by having your own weighting and symbol selections. You can freely draw trend lines and other drawing objects to manually analyze the data.

You can even use the weighted index for price pattern recognition to unveil information that is not available in a broad market index.

CHAPTER 13

Programming Guide

In This Chapter

Introduction to Programming

What can be Done with Programming

NeoTicker® is programmable. There are several categories of programming that you can do to extend the capabilities of NeoTicker®, they are:

- indicators
- trading systems
- custom statistics for trading systems performance analysis.
- custom drawing tools

When Not to Program

NeoTicker® can do a lot without programming.

For simple to moderately complex indicator, you should consider using formula to implement the indicator. See ***Tutorial: Making an Indicator with Formula Language, Example 1*** (on page 364). Formula is easy to write and fast, but does not support complex programming logic.

For trading systems, if your focus is trading signals, but not complex money management and portfolio, Backtest EZ is a great alternative. See ***Tutorial: Trading System Testing with Backtest EZ*** (on page 175).

You can also create trading system using NeoTicker®'s formula language, which is considerably easier than using a scripting language. See ***Creating Trading System with Formula*** (on page 311).

Methodology

This section will give you a big picture how programming works in NeoTicker®. It will give you an idea how things you programmed fits into NeoTicker®.

Programming Model: Scripts

You can program indicators, trading systems and custom drawing tools with built-in scripting support.

NeoTicker® can be programmed with Delphi Script, VBScript, and JavaScript. VBScript and JavaScript are industrial standard scripting languages.

NeoTicker® is a self contained programming environment. Script editor and debug window are provided to help you work with scripts.

Debugging window is provided to help you monitor the status of the scripts.

Script advantages:

- Fully integrated into NeoTicker® , you can develop, install, and launch scripts easily.
- Use of industrial standard script languages. Resource are readily available from books and Internet.
- Many indicators come with source code in script format.

Programming Model: IDL

For indicators and trading systems, NeoTicker® provides IDL interface to external programs created by programming tools such as Microsoft Visual Studio, Microsoft Visual C++, Microsoft Visual Basic, Borland C++ Builder, Borland Delphi, etc.

IDL is the interface to the NeoTicker®'s full set of programming objects.. For example, indicators, trading systems, etc are available to external programs via IDL. There is no need to recreate what's available in NeoTicker® from scratch.

IDL advantages:

- Extremely fast (comparable to internal indicators).
- You work in a professional development environment.
- If you use Visual Basic or Delphi as your programming tool, your code will be almost identical to VBScript and Delphi Script. If you need to translate from scripts to external program or vice verse, the work is minimal.

You can only create custom drawing tools with scripts.

Programming Tutorials

Tutorial: Creating an Indicator, Example 1

NeoTicker® supports programmable indicators. You can use one of the many scripting languages supported by NeoTicker® to develop your own indicator.

In this tutorial, we go through the steps to develop an indicator that plots two moving averages in VBScript.

The techniques shown here is equally valid for IDL indicators because the interface is the same.

Installing Windows Script Technologies

Requirements to run scripts:

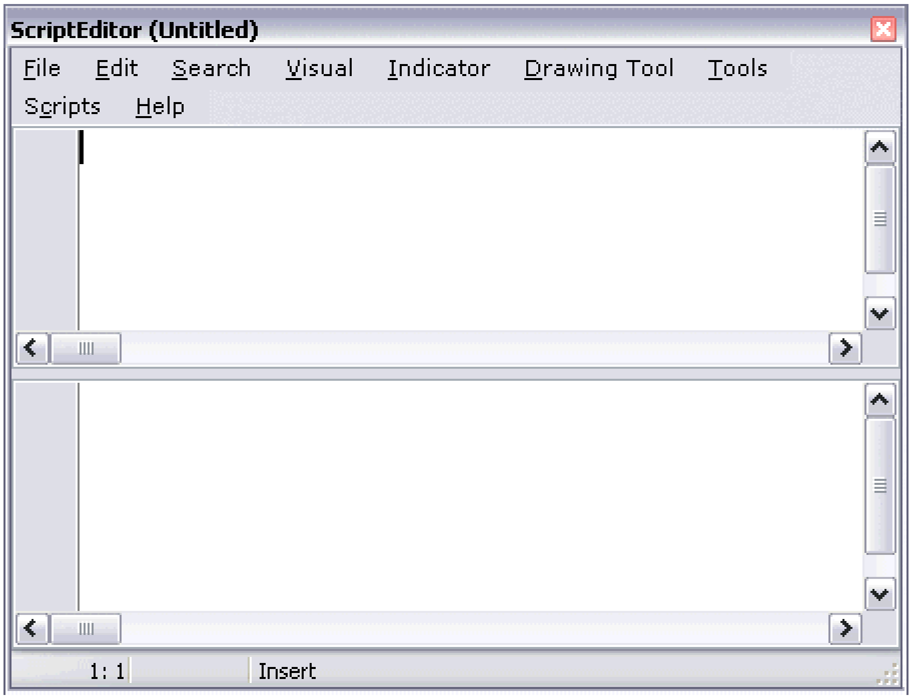
VBScript, JavaScript	Require Microsoft's Windows Script Technologies
Delphi Script	No Requirement

Windows Script Technologies is installed automatically if you have Internet Explorer version 4.0 or higher installed. If you are not sure, you can obtain ***Windows Script Technology*** (http:\\msdn.microsoft.com\scripting) for free.

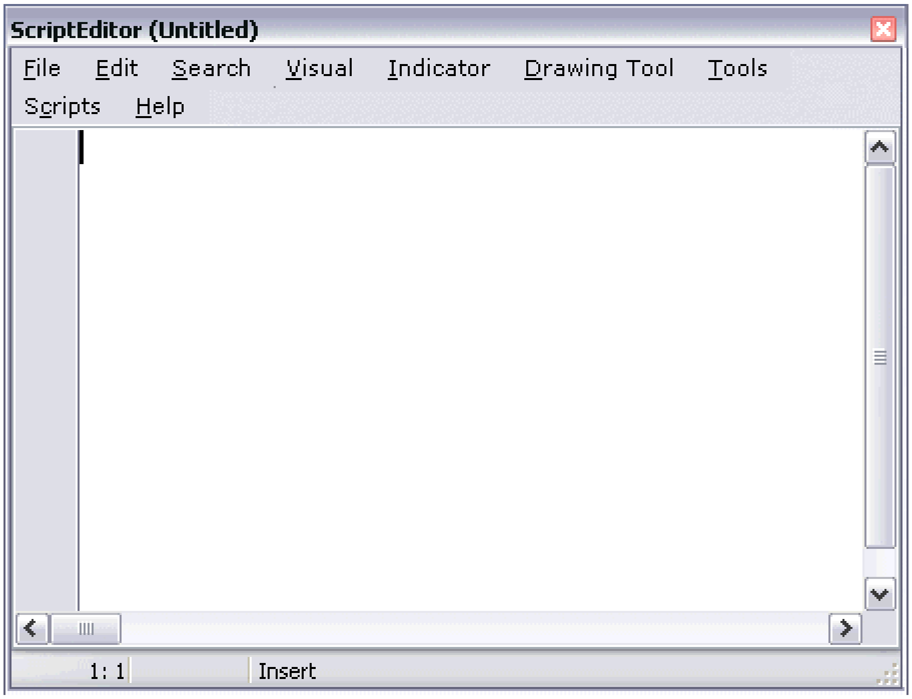
NeoTicker® works with Windows Script Technologies version 3.0 and above. At TickQuest, we use version 5.5 to develop and test programmable indicators.

Creating Script

Select **Program>Script Editor>New** from the main window. The script editor will be opened with a new blank script.



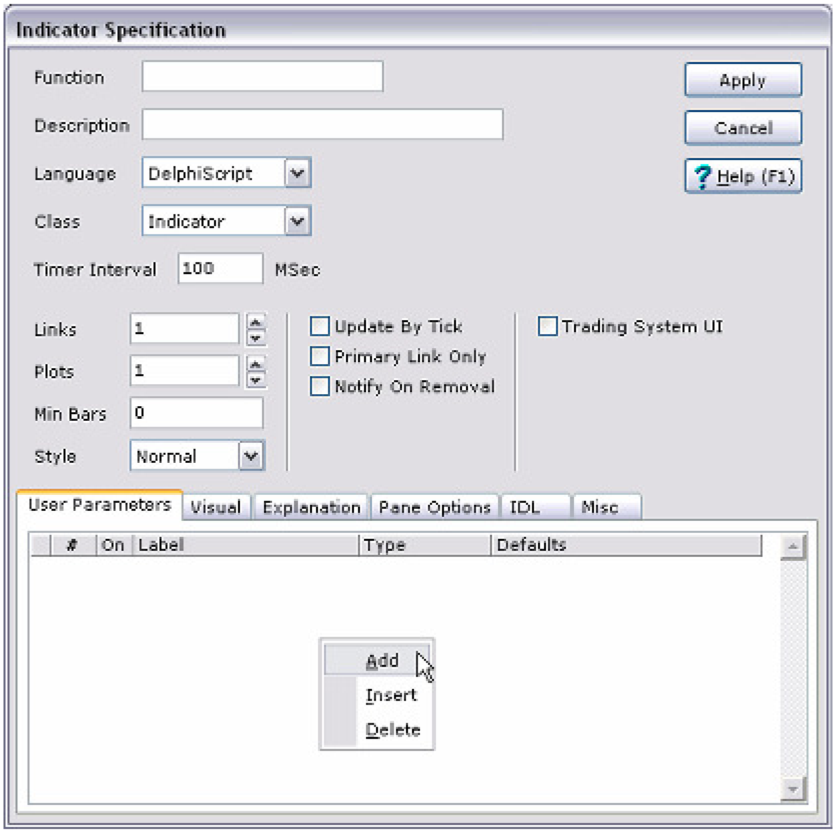
The script editor works like most other text editors. You can use the arrow keys, mouse, INSERT key, DELETE key, etc., to edit the script. By default, the script editor shows you two panes. The two panes let you view the same script at different positions. This will come in handy if you have to edit a large script. However, our example script is very small, so you can close the bottom pane by choosing **Visual>Visible 2**.


ScriptEditor (Untitled)
File
Edit
Search
Visual
Indicator
Drawing Tool
Tools
Scripts
Help
1: 1
Insert

Setting up Script as Indicator

First you will need to set up the script. Choose **Setup** from the **Indicator** menu in the script editor.

This will open the indicator specification window:



Indicator Specification

Function

Description

Language: DelphiScript

Class: Indicator

Timer Interval: 100 MSec

Links: 1

Plots: 1

Min Bars: 0

Style: Normal

□ Update By Tick

□ Primary Link Only

□ Notify On Removal

□ Trading System UI

	#	On	Label	Type	Defaults

You will need to enter several parameters to tell NeoTicker® how to the script as an indicator.

Funtion

Enter `mymov`. This is the name of the function NeoTicker® calls when it uses the indicator. You need to provide the name since you may have more than one function inside a script.

Description

Enter `My Moving Average`. This is the name of the indicator you will see in the indicator list when you add an indicator to a chart.

Script language

Choose VBScript

Links

Choose 1. This is the number of series the script links to. For moving average, one link is enough.

Plots

This is the number of plots of the indicator. Our indicator plots two line, so choose 2.

Min Bars

This is the minimum number of bars this indicator will start calculating. Keep the default 0.

Update By Tick

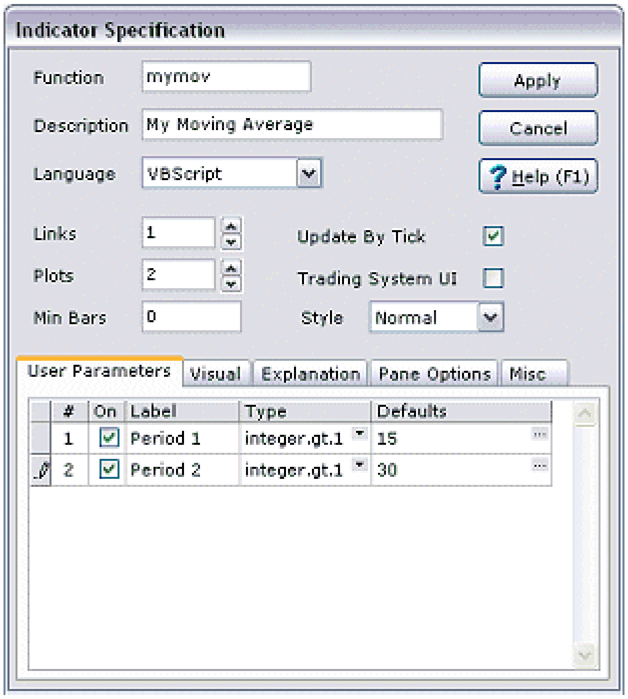
Whether the indicator will be updated by tick in real-time. Check the option.

Trading System UI

This will include a trading system UI for users to fill in system initial values. This is not a trading system, so keep this option off.

Then right click in the empty area under **User Parameters** tab to open the pop up menu, choose **Add** to a couple of parameters. These parameters are input values the user of the indicator can specify. Make 2 integer parameters. Call them Period 1 and Period 2. Set **Type** to integer.gt.1. Set Defaults to 15 and 30.

Your window will look like:

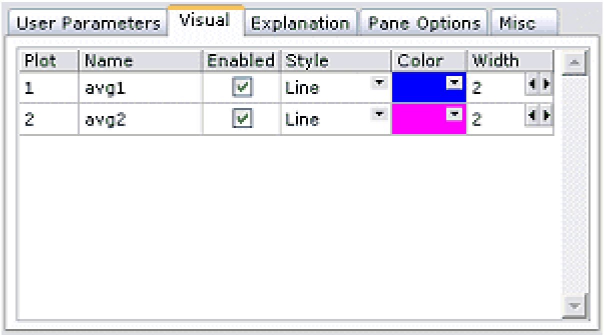


Indicator Specification

Function: mymov
Description: My Moving Average
Language: VBScript
Links: 1
Plots: 2
Min Bars: 0
Update By Tick ☑
Trading System UI □
Style: Normal

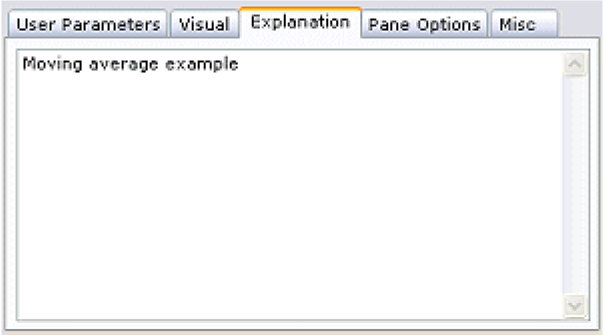
#	On	Label	Type	Defaults
1	☑	Period 1	integer.gt.1	15
2	☑	Period 2	integer.gt.1	30

Press the **Visual** tab. Under this tab, you can set up the default way of drawing the two plots in the chart. You can specify the name, whether the plot is enabled when created, plotting style, color and width for each individual plots. We've assigned different color for the plots with a width of 2, as shown below.



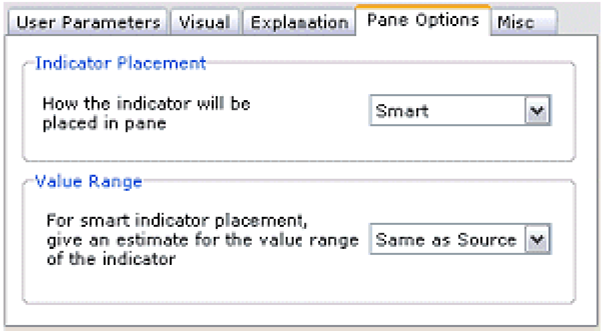
Plot	Name	Enabled	Style	Color	Width
1	avg1	☑	Line	🟦	2
2	avg2	☑	Line	🟪	2

Press the **Explanation** tab. This tab is the optional long explanation of the indicator to the user. Enter the following message in the blank area: `Moving average example.`



Moving average example

Press the **Pane Options** tab. This tab tells where the indicator should be placed in relation with the source data. Keep the **Smart** and **Same as Source** settings.



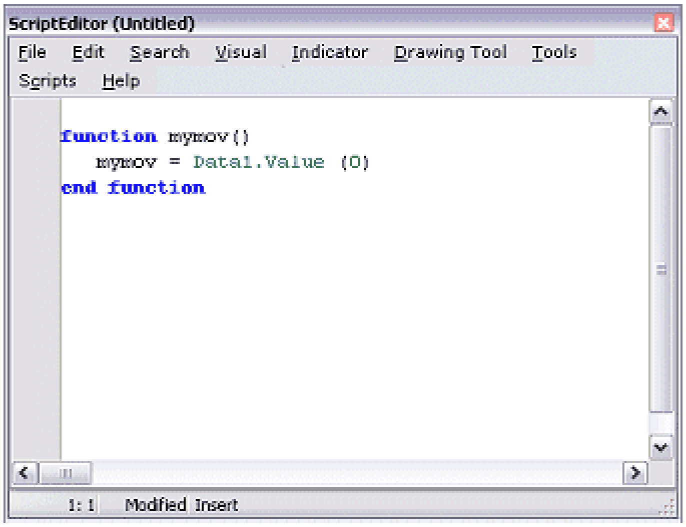
Indicator Placement

How the indicator will be placed in pane: Smart

Value Range

For smart indicator placement, give an estimate for the value range of the indicator: Same as Source

Once you are done with the specifications, press the **Apply** button. Script Editor will fill in a basic template for the script.



```
function mymov()
   mymov = Data1.Value (0)
end function
```

If you enable the script header by choosing **Visual>Show Script Header**, you will see the script header being displayed. The script header is the lines between the lines:

```
# NEOTICKER DATA BEGIN
```

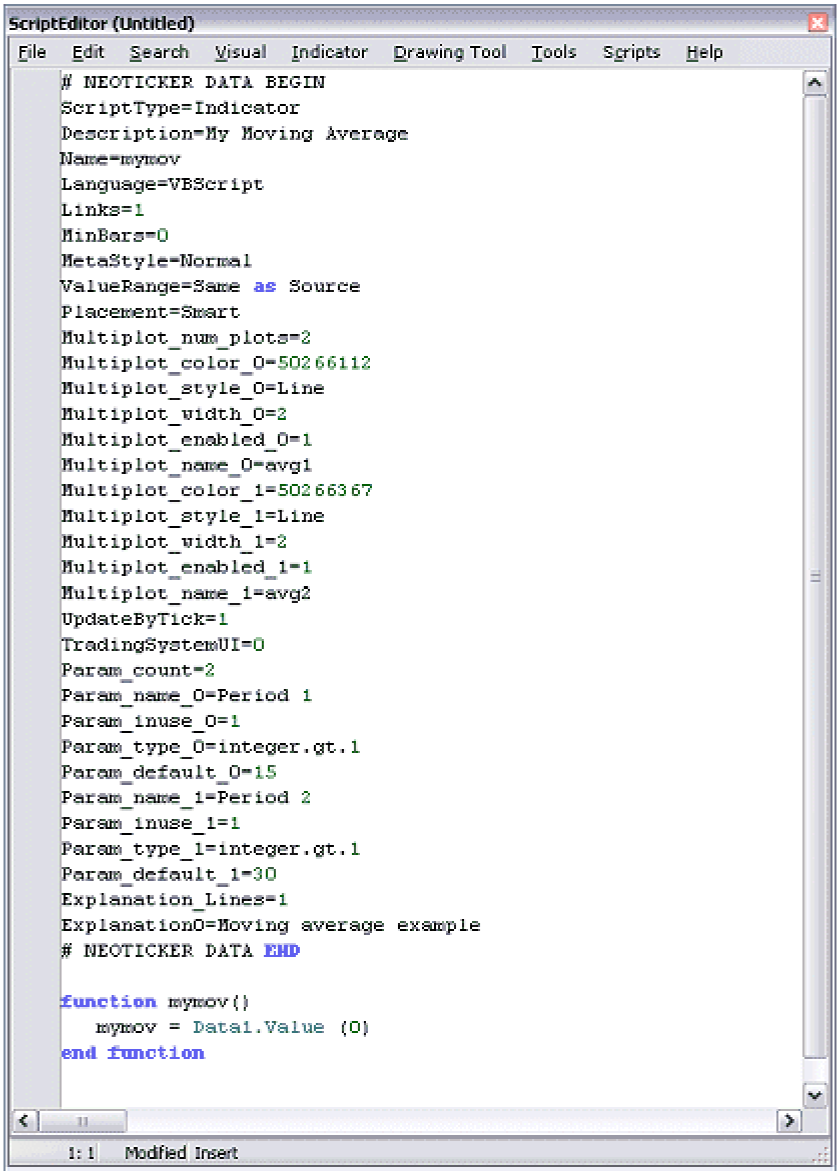
and

```
# NEOTICKER DATA END
```

The script header contains settings of the script. Under most circumstances, you can change the script settings by choosing **Indicator>Setup**, and there is no need to manually edit the header.

The main usage for showing the script header is to allow you to copy script and paste into other Script Editor. Because the header is included, the other Script Editor will set up the script properly.

Figure below shows the script with header.


ScriptEditor (Untitled)
File Edit Search Visual Indicator Drawing Tool Tools Scripts Help
1: 1 Modified Insert

```
# NEOTICKER DATA BEGIN
ScriptType=Indicator
Description=My Moving Average
Name=mymov
Language=VBScript
Links=1
MinBars=0
MetaStyle=Normal
ValueRange=Same as Source
Placement=Smart
Multiplot_num_plots=2
Multiplot_color_0=50266112
Multiplot_style_0=Line
Multiplot_width_0=2
Multiplot_enabled_0=1
Multiplot_name_0=avg1
Multiplot_color_1=50266367
Multiplot_style_1=Line
Multiplot_width_1=2
Multiplot_enabled_1=1
Multiplot_name_1=avg2
UpdateByTick=1
TradingSystemUI=0
Param_count=2
Param_name_0=Period 1
Param_inuse_0=1
Param_type_0=integer.gt.1
Param_default_0=15
Param_name_1=Period 2
Param_inuse_1=1
Param_type_1=integer.gt.1
Param_default_1=30
Explanation_Lines=1
Explanation0=Moving average example
# NEOTICKER DATA END

function mymov()
   mymov = Data1.Value (0)
end function
```

Entering Script

If you still have the script header showing from last step, turn it off by choosing **Visual>Show Script Header**.

Enter the following script to replace the script body you create from the last step.

```
function calcmov(p)
    'Calculate the moving average
    sumx = 0
    for i = 0 to p - 1
        sumx = sumx + data1.value(i)
    next
    calcmov = sumx / p
end function

function mymov()
    'Do not return a result if there is
    'not enough bars to calculate moving
    'average

    period1 = Params(1).int
    if data1.barsnum(0) <= period1 then
        Itself.SuccessEx(1) = false
    else
        ItSelf.Plot(1) = calcmov(period1)
    end if

    period2 = Params(2).int

    if data1.barsnum(0) <= period2 then
        ItSelf.SuccessEx(2) = false
    else
        'Calculate the moving average
        ItSelf.Plot(2) = calcmov(period2)
    end if
end function
```

The script is standard VBScript. If you are familiar with VBScript under web context, you will notice the language itself is exactly the same. NeoTicker® uses the same Microsoft scripting engine for VBScript.

The difference is NeoTicker® supplies objects to VBScript. These objects let VBScript access NeoTicker® data and functions.

Do not feel intimidated by the size of the script. It really is very simple once you understand what it does.

Functions

The script has two functions.

The first function `calcmov` performs a moving calculation given a period, p. It performs the calculation using the data from the first data link which is the data series (or other indicator) you apply the indicator on. Because the indicator will plot two moving averages, the two moving averages share `calcmov`.

The next function `mymov` is entry point of the indicator. This is set when you set up the indicator in the previous step of this tutorial. This function calculates two moving averages using the two parameters the indicator users provide.

Accumulation of Values

```
sumx = sumx + data1.value(i)
```

The function call `data1.value(i)` returns the value of i bars ago of the first data link. This value is accumulated over the period. The sum is then divided by the period to perform a moving average calculation.

Accessing User Parameters

```
period1 = Params(1).int
```

This line calls `Params(1).int`. This call returns the integer value of the first parameter. This value is then saved to the variable `period1`.

Checking Against Insufficient Amount of Data

```
if data1.barsnum(0) <= period1
```

The call `data1.barsnum(0)` returns the index number of the current bar after the minimum number of bars is meet. The if statement makes sure if there are not enough bars, the moving average will not be calculated.

```
ItSelf.SuccessEx(1) = false
```

This line is activated when the if statement above is true, i.e. there are not enough bars to calculate moving average. The `ItSelf` object represents the indicator itself. By setting the `SuccessEx` property to false. The script tells NeoTicker® that the first plot is not valid.

Calculating Moving Average

```
ItSelf.Plot(1) = calmov(period1)
```

This lines is activated when the if statement above is false, i.e. there are enough bars to calculate moving average. It calls the function `calmov` to calculate the moving average. The result is assigned to `ItSelf.Plot(1)`, which is the value of the first plot.

The same logic is reapplied to calculate the second moving average, which uses the period from the second parameter. The result is assigned to the second plot.

Verifying Script

You can verify an indicator script by choosing **Indicator>Verify**. Script editor will check the syntax of the script and will complain if there is a problem with the script's syntax.

If there is a problem with the script, script editor will position the cursor to the line where the problem occurs.

Verifying status is shown in a dialog. If you have confirmation dialog turned off, you can see the verifying status in the status bar of Script Editor.

Installing Script

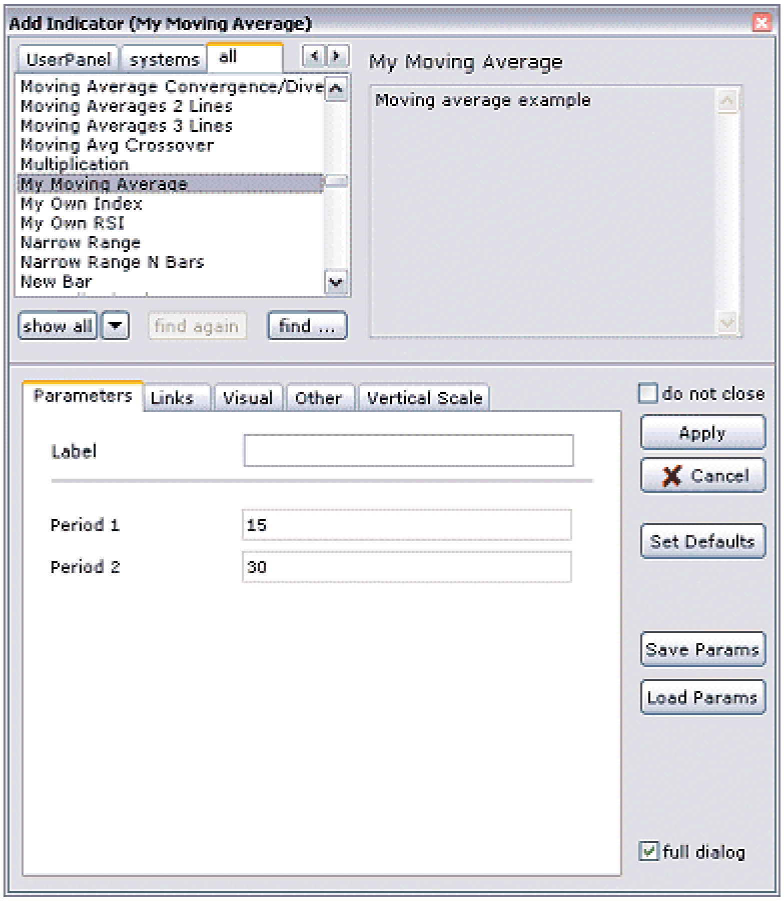
You must install the indicator script before you can use it. Choose **Indicator>Install**.

Installation will perform the following actions:

- Verify the script
- Save the script to file. If the script is new, script editor will ask you for a file name. The name and the extension of the file name do not matter, as long as the file name is unique
- Notify NeoTicker® a script is available to use

Running Script

Now it's time to see how the new indicator works. Create a new time chart and add a data series to it. Choose **Add Indicator** from the pop up menu. You will see **My Moving Average** is now an indicator you can choose.


Add Indicator (My Moving Average)

Label

Period 1: 15

Period 2: 30

Notice that after you choose **My Moving Average**, you can enter the **Period1** and **Period2** parameters you specified earlier. **Period1** and **Period2** have a default value of 15 and 30.

Press the **Apply** button to see the result:

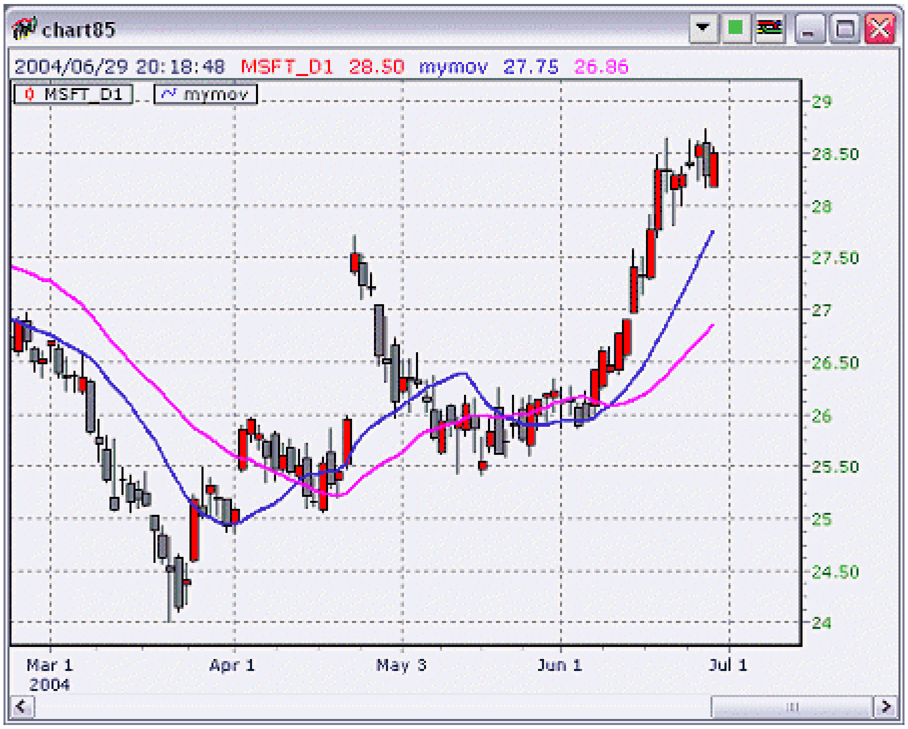


chart85
2004/06/29 20:18:48 MSFT_D1 28.50 mymov 27.75 26.86
0 MSFT_D1
mymov
29
28.50
28
27.50
27
26.50
26
25.50
25
24.50
24
Mar 1
2004
Apr 1
May 3
Jun 1
Jul 1

Tutorial: Creating an Indicator, Example 2

The second example of creating indicator shows you how to:

- Modify a script, re-install it and have the chart reflects the changes.
- Investigate and handle invalid data.
- Calling another indicator within the script.

The techniques shown here is equally valid for IDL indicators because the interface is the same.

Setting Up an Average Price Indicator

We will set up a new script indicator for this tutorial.

From the main window, choose **Program>Script Editor>New**. In the script editor, choose **Indicator>Setup**.

We are creating an indicator with function name `myavgprice`, choose VBScript as the language Set up the indicator as follows:

Function - myavgprice

Description - Z average price

Language - VBScript

Link - 1

Plot - 1

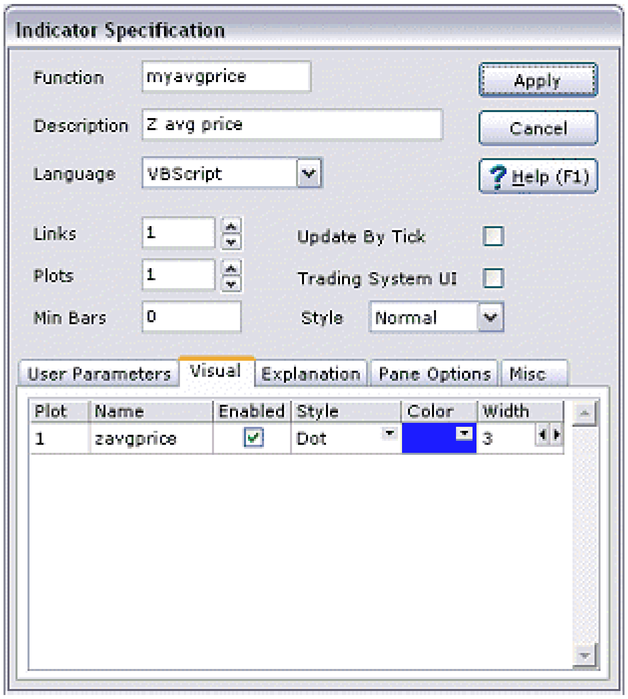
Min Bars - 0

Updated By Tick - off

Trading System UI - off

User Parameters - leave it blank

Under the **Visual** tab, set up the indicator to draw dots of width 3:



Function: myavgprice

Description: Z avg price

Language: VBScript

Links: 1

Plots: 1

Min Bars: 0

Update By Tick □

Trading System UI □

Style: Normal

Plot	Name	Enabled	Style	Color	Width
1	zavgprice	☑	Dot	🟦	3

Set plot 1 to use blue color and dot style. After you press the **Apply** button you should see the following script in the script editor:

```
function myavgprice()
    myavgprice = Data1.Value (0)
end function
```

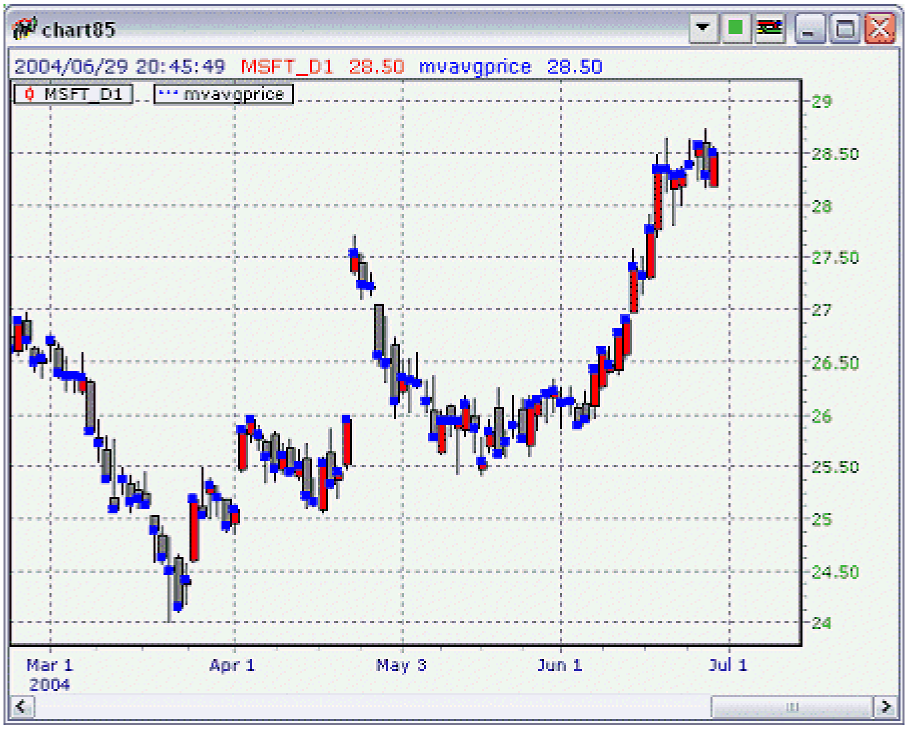
This is what you get initially after setup process is completed. NeoTicker® provide you with a function template to develop custom indicator.

The body of your indicator script is enclosed within function and end function statement. This function has one statement:

```
myavgprice = Data1.Value(0)
```

Data1 is an object. `Data1` exposes values of Link 1 to the indicator, `Value(0)` refer to the value of the current bar.

Choose **Indicator>Install** in the script editor to install this indicator. Create a chart. Then load a MSFT daily data series, add this indicator to the chart, you will see a series of blue dots plotting the daily close of MSFT.



Back to the script editor. Replace the line:

```
myavgprice = Data1.Value(0)
```

with the line:

```
myavgprice = (Data1.High(0) + Data1.Low(0) + Data1.Close(0))/3
```

The statement above adds the high, low, close price of Data1 and takes the average. The script should look like this:

```
function myavgprice()
    myavgprice = (Data1.High (0) + Data1.Low (0) + Data1.Close
(0))/3
end function
```

From script editor, choose **Indicator>Install** to install the script. NeoTicker® will ask you with a dialog box if you want to replace the current working one, click **Yes**.

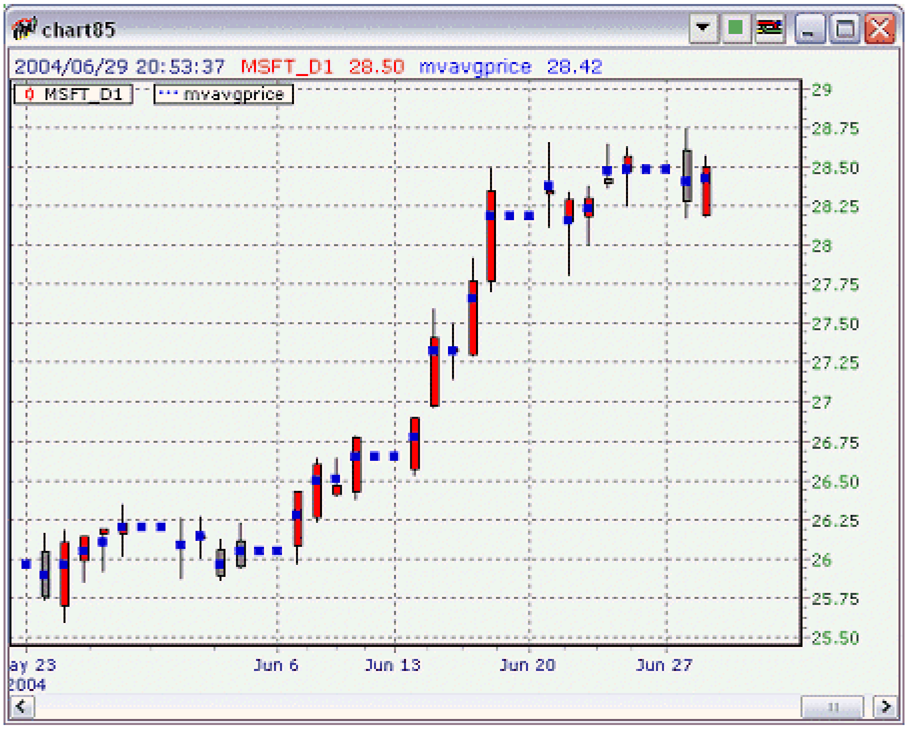
Afterward NeoTicker® will have a dialog box telling you script installation is successful.

If you revise a script and re-install it, the new definition is applied only on new data that comes into the chart in real-time.The old indicator values are not changed. This design avoids long delay problem if you have a lot of data in the chart.

To force the whole indicator to be recalculated using the revised script, press the refresh chart button 🔄 in the chart tool.

In the chart, open Chart Manager, go to Data Settings tab, and change the trading days to 7 days a week.

The chart would look like:



When you look closely at the candlestick chart you will see that there are gaps in the price series, those gaps are caused by missing data when there is no trading (Saturdays and Sundays). However the plotting of average price still continue when there is no price data, this is a misrepresentation of the data. We will add additional code to skip over invalid data points.

In the script editor, we will add the following lines in front of the price calculation.

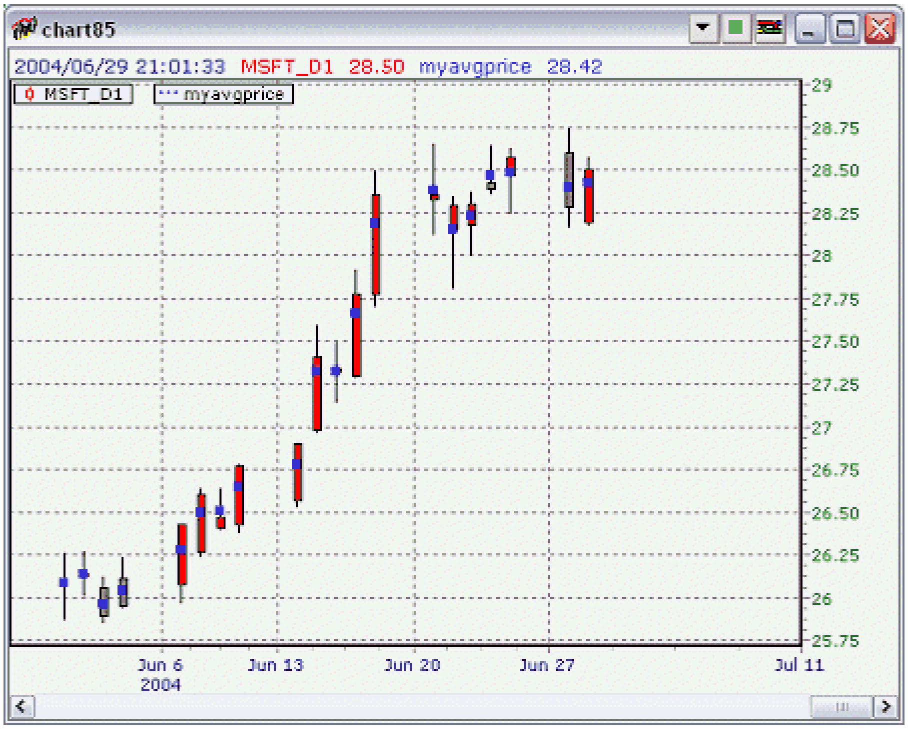
```
if not Data1.valid(0) then
    Itself.success = false
```

```
        Exit function
    End if
```

The above statements checks for the validity of the current Data1 bar. If the bar is not valid, then the script assign an invalid state to the indicator and aborts indicator calculation, so the plot will not show up in the chart. The script should look like:

```
    function myavgprice()
        If not Data1.valid(0) then
            Itself.success = false
            Exit function
        End if
        myavgprice = (Data1.High (0) + Data1.Low (0) + Data1.Close
    (0))/3
    end function
```

Choose **Indicator>Install** to install the script and refresh the chart then you should see the indicator not plotting when there is no data.



Using Another Indicator within Indicator

An alternative to call other indicators is by using formula within a script/program. To find out more, refer to ***Quick Combination of Multiple Indicators*** (on page 1395).

In the last section, we have constructed a basic indicator call `myavgprice`. In this section, we will show you how to call myavgprice indicator from another indicator, which calculates the 10 period smoothing on `myavgprice`.

Choose **Program>Script Editor>New** to create a new indicator. Setup a new indicator by clicking on **Indicator>Setup**. In the **User Parameters** tab, name it **mysmoothaprice**, set **Link** to 1, **Plot Number** 1, **Language** to VBScript. In the **Visual** tab, set **Style** to line and **Color** to red.

After you apply the settings, in the script editor, change the script to:

```
function mysmoothaprice()
    dim avgp

    itself.makeindicator "myap", "myavgprice", Array("1"),
Array("")
    avgp = itself.indicator("myap")
    mysmoothaprice = avgp.value(0)
end function
```

When you install and run the above code you should end up with a data series of values exactly like the one you get from Z Average Price, because we call `myavgprice` and simply return the result of the calculation.

We will break down the `makeindicator` line to explain NeoTicker® nested indicator syntax.

The statement `itself.makeindicator` creates a new instance of the indicator `myavgprice`.

The first string we've passed in is `"myap"`, this can be any unique string, this string will be how you refer to this instance of the indicator object within the scope of this function.

The second string is `"myavgprice"` which is the function name of the indicator we are calling, this name is the name in the function column of Indicator Manager and Indicator Quick Reference.

The `Array("1") parameter` is a VBScript specific syntax. It specifies the Link 1 of `myavgprice` by assigning Link 1 of `mysmoothaprice` to Link 1 of `myavgprice`.

The `Array("")` parameter is the parameter list to `myavgrprice`. In this case of `myavgprice`, it does not require any parameter therefore there is nothing in between the quote.

Now we can the code to smooth average price series. First define another object holder call sap. Add this line bellow the `dim avgp` line:

```
dim sap
```

Add the following lines after the forth line avgp = itself.indicator("myap") :

```
ItSelf.Makeindicator "myavg", "average", Array("myap"),
Array("10")
sap = itself.indicator("myavg")
```

This is an example how you can pass the result of an indicator series into another indicator.

The makeindicator line calls the moving average indicator, by linking `"myap"` to the moving average indicator as link 1. It passes a parameter of "10" to the moving average indicator for a 10-period smoothing.

When you pass parameters to the indicator created by MakeIndicator, the parameters must be specified as an array of strings.

Change the line:

```
mysmoothprice = avgp.value(0)
```

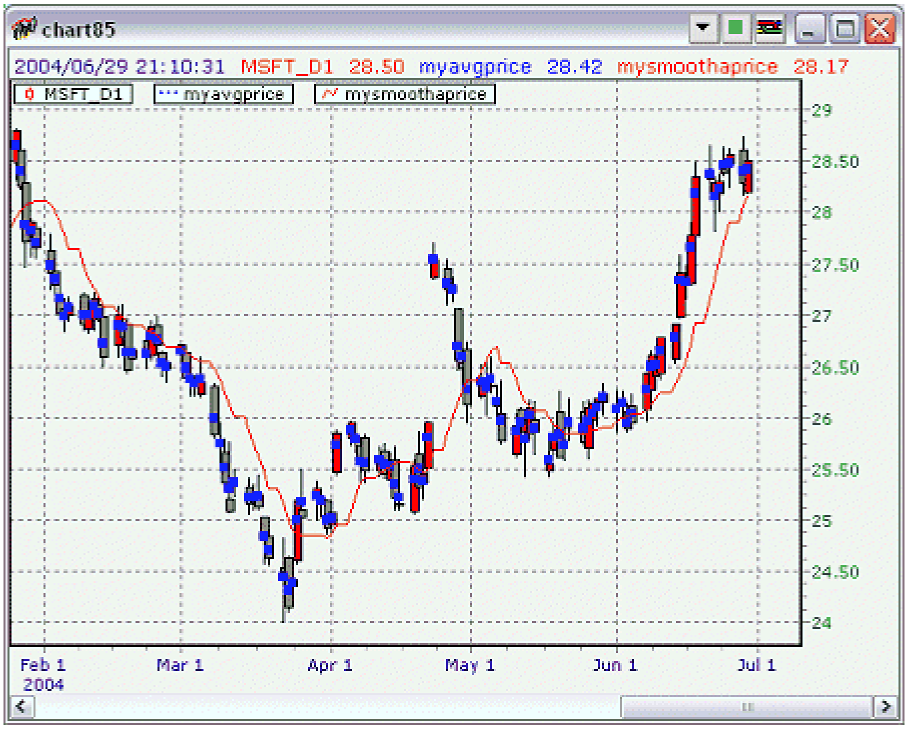
to

```
mysmoothprice = sap.value(0)
```

Your resulting code should look like this:

```
function mysmoothaprice()
    dim avgp
    dim sap

    itself.makeindicator "myap", "myavgprice", Array("1"),
Array("")
    avgp = itself.indicator ("myap")

    itself.makeindicator "myavg", "average", Array("myap"),
Array("10")
    sap = itself.indicator ("myavg")

    mysmoothaprice = sap.value(0)
end function
```

When you apply this indicator to the chart, it will look like:



To verify the result you can utilize NeoTicker®'s ability to apply indicator on indicator visually from within the chart. Since what we did is a 10 period smoothing of the average price. You can first add the average price indicator to the price series. Then add the Simple Moving Average and use myavgprice as the link. This should result with a line the same shape as the one we have created from this tutorial's code.

For a complete discussion on using other indicators within a script, see ***Using Other Indicators*** (on page 1479).

Performance Note

You can improve indicator performance by using UpdateIndicator to obtain an indicator instance. See ***Example: UpdateIndicator*** (on page 1488).

Tutorial: How to Evaluate Formula in Scripts and IDL

When we want to define complex price and indicator relationships within a script, the readability of the script will start to drop because you will need to write quite a number of lines of code to describe the price patterns and indicator conditions. Equipped with the built-in indicator “Formula” (short form “fml”), this task can be greatly simplified.

The following walk through assumes you have some basic knowledge of the formula language. For more information on formula, refer to ***Formula Tutorials and Examples*** (see "Formula Topics and Tutorials" on page 255).

Short Review on Using the MakeIndicator and Indicator Methods

To create an embedded indicator within your indicator instance, we use the MakeIndicator method. For example,

In Delphi Script and Borland Delphi:

```
ItSelf.MakeIndicator ('MyAvg', 'average', ['1'], ['10']);
```

In VBScript and Visual Basic

```
ItSelf.MakeIndicator "MyAvg", "average", Array ("1"), Array
("10")
```

The statement above creates an instance of simple moving average with parameter period equals to 10, on the first (1) link series of the indicator you are writing.

You can then access the indicator values using the ItSelf.Indicator function as follows.

In Delphi Script and Borland Delphi:

```
ItSelf.Indicator ('MyAvg').value [0];
```

In VBScript and Visual Basic:

```
ItSelf.Indicator ("MyAvg").Value (0)
```

Create an Embedded fml Indicator

Now say you are interested in the price pattern Harami Age described in the section ***Tutorial: How to Highlight Complete Bars with Custom Conditions Using the Indicator Highlight Bar Formula*** (on page 385).

If you want to write script code to recognize this pattern, your script could look like this.

In Delphi Script and Borland Delphi:

```
if (data1.high [2] > data1.high [1]) and
```

```
        (data1.low [2] < data1.low [1]) and
        (data1.close [2] < data1.open [2]) and
        (data1.close [1] > data1.open [1]) and
        (data1.close [0] > data1.close [1]) then
            MyCond := true;
```

In VBScript and Visual Basic:

```
if (data1.high (2) > data1.high (1)) and _
   (data1.low (2) < data1.low (1)) and _
   (data1.close (2) < data1.open (2)) and _
   (data1.close (1) > data1.open (1)) and _
   (data1.close (0) > data1.close (1)) then
      MyCond = true
end if
```

There are two deficiencies to this method.

First, the syntax is hard to read across different scripting languages.

Second, you cannot reference to the previous values of the condition. Thus you cannot check for historical happenings of the condition.

To address both issues, you can use the fml indicator as followings.

In Delphi Script and Borland Delphi:

```
ItSelf.MakeIndicator ('MyPattern', 'fml', ['1'],
    ['h (2) > h (1) and l (2) < l (1) and c (2) < o (2) and c
(1) > o (1) and c > c (1)']);
```

In VBScript and Visual Basic:

```
ItSelf.MakeIndicator "MyPattern", "fml", Array ("1"), _
Array ("h (2) > h (1) and l (2) < l (1) and c (2) < o (2) and
c(1) > o (1) and c > c (1)")
```

You have now defined the `MyPattern` indicator instance using the fml indicator and you can use the following code to replace the bulky if statement above.

In Delphi Script and Borland Delphi:

```
if itself.indicator ('MyPattern').value [0] <> 0 then
```

In VBScript:

```
if itself.indicator ("MyPattern").value (0) <> 0 then
```

If you want to check if the condition has happened in a previous bar, all you have to do is to check for the condition by specifying the number of bars ago. Say in this case, you are interested in the condition of 10 bars ago, you can write the following code to do the task.

In Delphi Script and Borland Delphi:

```
if itself.indicator ('MyPattern').value [10] <> 0 then
```

In VBScript and Visual Basic:

```
if itself.indicator ("MyPattern").value (10) <> 0 then
```

Quick Combination of Multiple Indicators

Another useful aspect of the formula indicator is its ability to combine multiple indicators easily. For example, if you were interested in the smoothed difference between a simple moving average and an exponential moving average of your data series, without formula, you would normally write the following code.

In Delphi Script and Borland Delphi:

```
ItSelf.MakeIndicator ('sma', 'average', ['1'], ['10']);
ItSelf.MakeIndicator ('ema', 'xaverage', ['1'], ['10']);
ItSelf.MakeIndicator ('diff', 'diff', ['sma', 'ema'], ['']);
ItSelf.MakeIndicator ('myavg', 'average', ['diff'], ['10']);
```

In VBScript and Visual Basic:

```
ItSelf.MakeIndicator "sma", "average", Array ("1"), Array
("10")
ItSelf.MakeIndicator "ema", "xaverage", Array ("1"), Array
("10")
ItSelf.MakeIndicator "diff", "diff", Array ("sma", "ema"),
Array ("")
ItSelf.MakeIndicator "myavg", "average", Array ("diff"), Array
("10")
```

For each step in the combination you will need to write one line of MakeIndicator to construct the necessary intermediate step until you arrive at the final answer. With indicator “fml”, you can do the same in one expression.

In Delphi Script and Borland Delphi:

```
ItSelf.MakeIndicator ('myavg', 'fml', ['1'],
    ['average (average (data1, 10) - xaverage (data1, 10),
2)']);
```

In VBScript and Visual Basic:

```
ItSelf.MakeIndicator "myavg", "fml", Array ("1"), _
Array ("average (average (data1, 10) - xaverage (data1, 10),
2)")
```

Later in your code, you can access the values by using the Indicator method.

In Delphi Script and Borland Delphi:

```
ItSelf.Indicator ('myavg').value [0]
```

In VBScript and Visual Basic:

```
ItSelf.Indicator ("myavg").value (0)
```

Tutorial: Writing a Breakout Trading System

This tutorial will show you how to write a trading system using VBScript.

Using a scripting or programming language is one way of implementing trading system. If you prefer not to program, you can use Backtest EZ as an alternative. For more information, see ***Backtest EZ*** (on page 1275).

Overview

Writing a trading system is not a complex task. It is no more than expressing your ideas in terms of the object model within your favorite programming environment. We will present a case study on designing a simple trading system. Most of the time, trading system research is an interactive task that requires you to repeat the cycle of design, implement, and then test run your system against real data many times before something useful can be created for real-time usage.

It is required that you have some basic understanding on how to write indicators. For information about indicator writing, see ***Indicator Programming Topics*** (on page 1443).

You may also want to gain some understanding on how trading system works in NeoTicker®. For more information, see ***Trading System Programming Topics*** (on page 1491).

NeoTicker® uses an object model for programming. Object reference can be found in ***Objects Reference*** (on page 1517).

Idea

There is a classic concept in trading called breakout. Breakout happens when a market is trading within a tight range for a period of time and then starts to move out of this tight region. It is believed that when the price moves out of the tight region, the momentum of the move is very powerful and the move has continuation strength due to the sheer number of traders being trapped at the wrong side during the tight range period and the additional forces of traders jump into the boat who was staying on the sideline during the period of tight range.

To implement this idea we will have to identify the components of the trading system. First, there is a setup of tight range period. We need a clear definition of what constitute a tight range and a clear definition for the length of the time period involved that can qualify. Second, the signal to entry is the move away from the tight range. We will need a clear definition of what constitute a move away from the tight range as well.

When we talk about clear definition, it must be expressed in exact terms, which can then be translated into rules that can be recognized mechanically through the trading system.

First Attempt

Defining the Rules

There are several common trend identification techniques. Since we are not sure which is more useful in our system, we can choose to use a formula parameter to hold the condition for identifying the tight range or congestion situation.

At the moment, we can use one of the standard trend indicator ADX to put into the formula as the default formula condition. If we like to change that, it can be done during run time.

It is well known that when ADX is greater than 30 and is rising, the trend is strong. When ADX is below 30 the trend is not well developed, and when ADX is below 15, there is no trend in the market.

We will use this property of ADX < 15 as our tight range condition.

For the duration of tight range, it is also an user adjustable parameter.

For the breakout signal, we can simply use highest high and lowest low values during the tight range period as the breakout price reference. Then a parameter can be used as the offset from these reference prices to define the break out entry price.

Basic System

Here is the trading system written in VBScript.

```
const hHHV  = 0
const hLLV  = 1

function BreakoutSystem
dim tightrange, tightrangec, hhv, llv
dim prevorder

   tightrange = itself.makeindicator ("tr", "fml", Array ("1"),
Array (param1.str))
   tightrangec = itself.makeindicator ("trc", "summation", Array
("tr"), Array (param2.str))

   hhv = itself.makeindicator ("hhv", "hhv", Array ("1.h"), Array
(param2.str))
   llv = itself.makeindicator ("llv", "llv", Array ("1.l"), Array
(param2.str))

   if not tightrangec.valid (0) then
      itself.successall = false
      exit function
   end if

   if heap.size = 0 then
      heap.allocate 2
      heap.fill 0, 1, 0
   end if
   if tightrangec.value (0) >= param2.int then

      if tightrangec.value (1) < param2.int then ' was not in
tight range, new order
         heap.value (hHHV) = hhv.value (0)
         heap.value (hLLV) = llv.value (0)
         trade.cancelallopenorders
      else ' order was placed
         if hhv.value (0) > heap.value (hHHV) then heap.value
(hHHV) = hhv.value (0)
         if llv.value (0) < heap.value (hLLV) then heap.value
(hLLV) = llv.value (0)
      end if

      trade.longstop heap.value (hHHV) + param3.real, param4.int,
otffillorkill, "L"
      trade.shortstop heap.value (hLLV) - param3.real, param4.int,
otffillorkill, "S"

   end if

   BreakoutSystem = trade.currentequity
end function
```

You do not have to type the script in. The code can be found in the `Samples` folder, under the `Breakout Trading System` folder. The file is called `01_Breakoutsystem.VBS`, e.g.

```
C:\Program Files\TickQuest\NeoTicker3\Samples\Breakout Trading
System\01_Breakoutsystem.VBS
```

To open and install this trading system:

1 Open script editor by choosing from the the main window, **Program>Script Editor>Open Indicator**, then choose the script file.

2 In the script editor, choose **Indicator>Install**. This will automatically copy the script to your indicator directory and install the trading system. The trading system is installed as Breakout System.

Dissecting the System

The system starts by creating a formula indicator:

```
tightrange = itself.makeindicator ("tr", "fml", Array ("1"),
Array (param1.str))
```

The first parameter, `param1`, is where we are going to put in trend. We do not hard code the ADX condition here but rely on the formula indicator to evaluate the trend. This provides maximum flexibility when you use the trading system. Basically, `tightrange` will evaluate to 0 or 1 (because it is an ADX comparision).

The next step is accumulation of `tightrange` over the user specified period (`param2`).

```
tightrangec = itself.makeindicator ("trc", "summation", Array
("tr"), Array (param2.str))
```

Basically, `tightrangec` is number of times the tight range condition is satisfied within the given period. Specially, `tightrangec` will be the number of times the ADX comparision is true.

The next step is creation of the highest high and lowest low value as the price reference for breakout:

```
hhv = itself.makeindicator ("hhv", "hhv", Array ("1.h"), Array
(param2.str))
llv = itself.makeindicator ("llv", "llv", Array ("1.l"), Array
(param2.str))
```

Highest high and lowest low are indicators. Note that they take the second user parameter as the period.

The next step is a safety check. Most trend indicator including ADX needs several bars before the indicator value becomes meaningful. We do not want to proceed to trading until the trend is ready:

```
if not tightrangec.valid (0) then
    itself.successall = false
    exit function
end if
```

The next step allocates memory for storing the strongest highest high and lowest low values over the tight range period. Trading system uses at the point evaluation. Any persistent values over multiple points must be stored in the `Heap` object. The trading system will eventually index the heap values with the index `hHHV` and `hLLV`.

```
if heap.size = 0 then
   heap.allocate 2
   heap.fill 0, 1, 0
end if
```

Recall that we want the tight range condition to hold for some period before we consider it really is a tight range. The next step in the code checks exactly that:

```
if tightrangec.value (0) >= param2.int then
```

Recall that `tightrangec` is the number of times the tight range condition is true. By comparing the value to the period (`param2`), the trading system proceed to order placement only when the tight range condition has been true for the specified period.

The next step checks whether we are just entering a tight range. Note that we are just entering a tight range, the previous value of `tightrangec` will be less the period specified.

```
if tightrangec.value (1) < param2.int then ' was not in tight
range, new order
```

When we just enter a tight range, we will record the initial highest high and initial lowest low values. These values are later used to check for breakout:

```
heap.value (hHHV) = hhv.value (0)
heap.value (hLLV) = llv.value (0)
```

We put in an optional safety check to cancel all hanging open orders from previous bars.

```
trade.cancelallopenorders
```

Suppose we are not just entering the tight range, but the tight range has been there for a while. This situation is handled by the code blow:

```
else ' order was placed
   if hhv.value (0) > heap.value (hHHV) then heap.value (hHHV)
= hhv.value (0)
   if llv.value (0) < heap.value (hLLV) then heap.value (hLLV)
= llv.value (0)
end if
```

What the code above does is evaluating the highest high and lowest low values, and storing the strongest values in the heap within the tight range.

The next step is actual order placement:

```
trade.longstop heap.value (hHHV) + param3.real, param4.int,
otffillorkill, "L"
trade.shortstop heap.value (hLLV) - param3.real, param4.int,
otffillorkill, "S"
```

There is no need to explicitly code the breakout logic. The breakout can be nicely handled by stop orders.

The long side of the system works like this: when the price of the security is above the highest high value within the period (`heap.value (hHHV)`) plus some leeway (`param3`), then long stop order of specific size (`param4`) will be triggered and becomes a market order. In case the condition is not met, we specify that the order should be killed (otffillorkill).

The short side of the system works similarly.

For our convenience, we have assigned the system equity (`CurrentEquity`) to plot1 of the indicator so that we can spot the performance quickly:

```
BreakoutSystem = trade.currentequity
```

This system is pretty simple. As it does not have any special rules for money management.

Preparing the Data

You can follow this tutorial with your own data, but if you want to reproduce the exact result seen here, a data file is provided. The data file is:

```
C:\Program Files\TickQuest\NeoTicker3\data\ES-F.CSV
```

This is the S&P e-mini minute data from 2003/01/27 to 2003/04/17.

Test Run

The following are the steps to apply the trading system using the sample data provided.

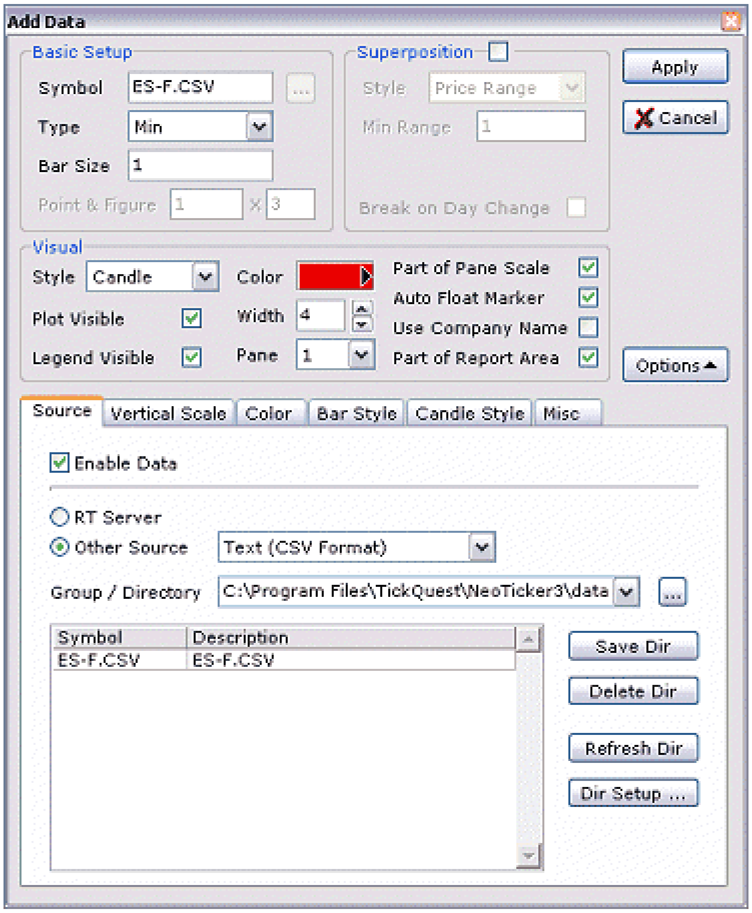
If you are using sample data, follow from step 1.

If you are using your own data, open a chart and load your data, and start from step 6.

1 Open a time chart, open Chart Manager, in the **Data Settings** tab, set the trading time to 9:30 to 16:15.

2 In the time chart. choose **Add Data** from pop up menu.

3 In the add data dialog, under the **Source** tab, set **Other Source** to **Text (CSV)**. Make sure you points the directory to the `Data` folder and click on **ES-F.CSV**. Set it to 1 minute.



Add Data

Basic Setup
Symbol: ES-F.CSV
Type: Min
Bar Size: 1
Point & Figure: 1 X 3

Superposition □
Style: Price Range
Min Range: 1
Break on Day Change □

Visual
Style: Candle
Color: 🟥
Width: 4
Pane: 1
Plot Visible ☑
Legend Visible ☑
Part of Pane Scale ☑
Auto Float Marker ☑
Use Company Name □
Part of Report Area ☑

☑ Enable Data
○ RT Server
● Other Source: Text (CSV Format)
Group / Directory: C:\Program Files\TickQuest\NeoTicker3\data

Symbol	Description
ES-F.CSV	ES-F.CSV

4 Press the **Apply** button to add data.

5 The data covers the 2003/01/27 to 2003/04/17 time period. In the chart, set a long days to load to cover all the data. You can press the HOME and END key on the keyboard to verify the chart indeed covers this period.

6 Select the data series, and from pop up menu, choose **Add Indicator**. Choose **Breakout System**.

7 Under the **System** tab, set the following trading system parameters:

Price Multiple set to 50

Min Tick Size to 0.25

Enable **Single Entry Per Direction**

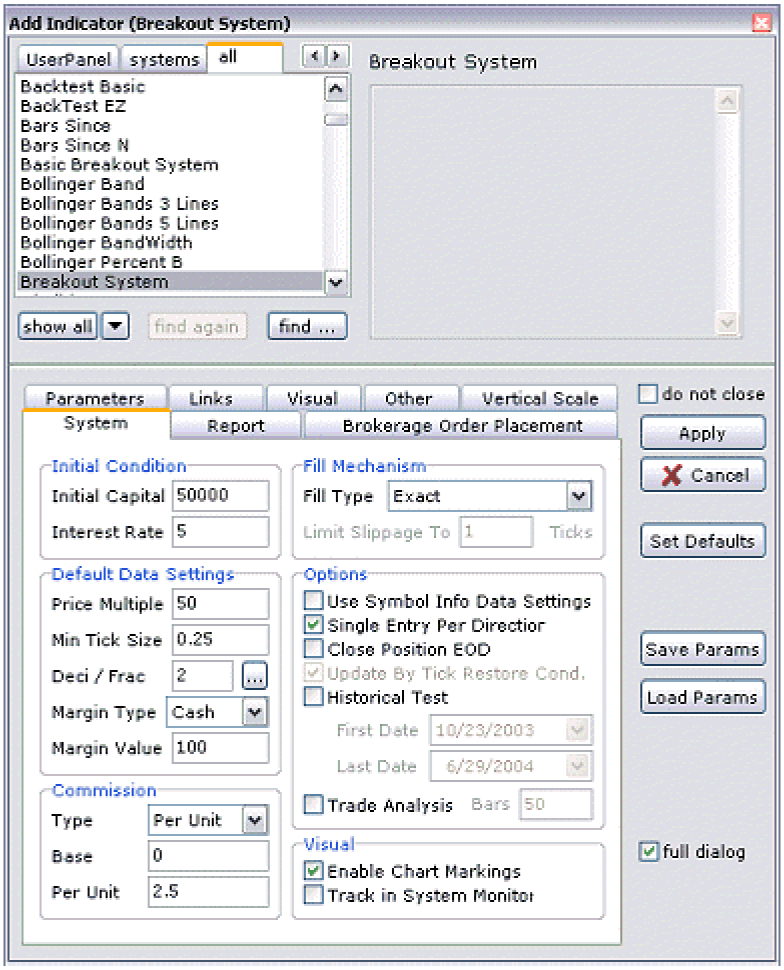
Fill Type to **Exact**

Commission Type to **Per Unit**

Commission Base to 0

Commission Per Unit to 2.5

These parameters are for trading S&P 500 e-mini contract. If you are testing against stocks, you will need to make the appropriate adjustments, e.g. **Price Multiple** to 1, **Min Tick Size** to 0.01, and a compatible commission.



Add Indicator (Breakout System)

UserPanel | systems | all

Backtest Basic
BackTest EZ
Bars Since
Bars Since N
Basic Breakout System
Bollinger Band
Bollinger Bands 3 Lines
Bollinger Bands 5 Lines
Bollinger BandWidth
Bollinger Percent B
Breakout System

Breakout System

Parameters | Links | Visual | Other | Vertical Scale
System | Report | Brokerage Order Placement

Initial Condition
Initial Capital 50000
Interest Rate 5

Default Data Settings
Price Multiple 50
Min Tick Size 0.25
Deci / Frac 2
Margin Type Cash
Margin Value 100

Commission
Type Per Unit
Base 0
Per Unit 2.5

Fill Mechanism
Fill Type Exact
Limit Slippage To 1 Ticks

Options
□ Use Symbol Info Data Settings
☑ Single Entry Per Direction
□ Close Position EOD
☑ Update By Tick Restore Cond.
□ Historical Test
First Date 10/23/2003
Last Date 6/29/2004
□ Trade Analysis Bars 50

Visual
☑ Enable Chart Markings
□ Track in System Monitor

□ do not close
☑ full dialog

8 Under the **Parameters** tab, set the following parameters for the trading system:

Tight Range to `ADX (data1, 14) < 15`

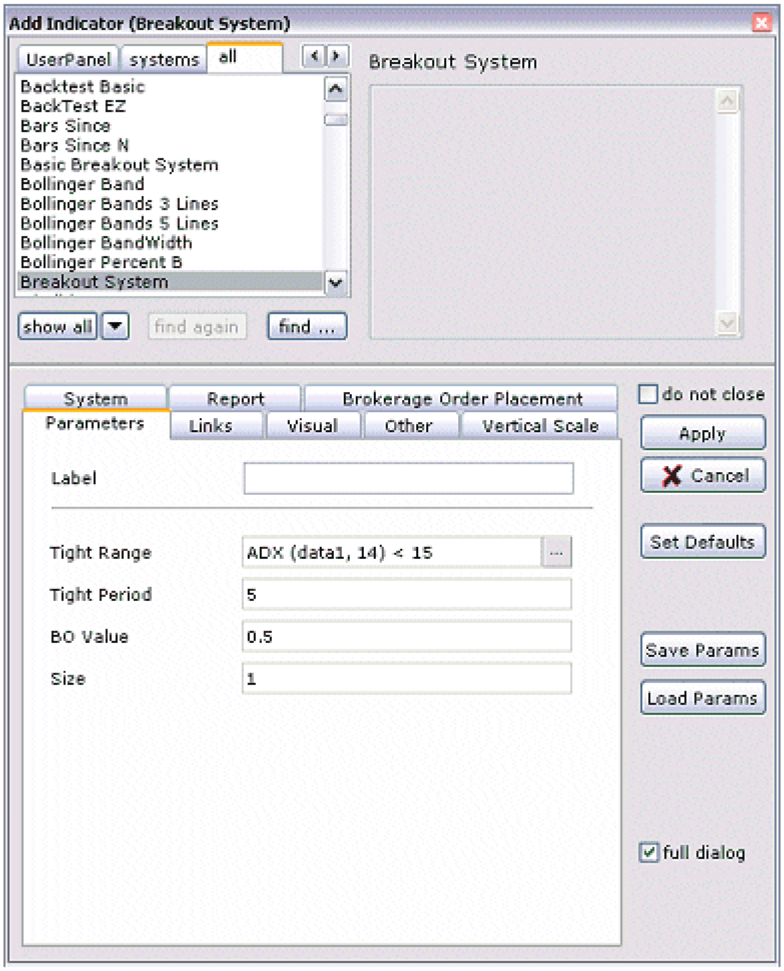
Tight Period to 5

BO Value to 0.5

Size to 1

Tight Range is the condition when a tight range is hold. **Tight Period** is the period for the tight condition to hold for the range to be considered as tight. **BO Value** is the breakout value. **Size** is the order size.

If you are testing against stocks, you should adjust **BO Value** to match the stock's price movement and increase **Size** to number of shares, e.g. **BO Value** to 0.02, **Size** to 100.



Add Indicator (Breakout System)

Backtest Basic
BackTest EZ
Bars Since
Bars Since N
Basic Breakout System
Bollinger Band
Bollinger Bands 3 Lines
Bollinger Bands 5 Lines
Bollinger BandWidth
Bollinger Percent B
Breakout System

Breakout System

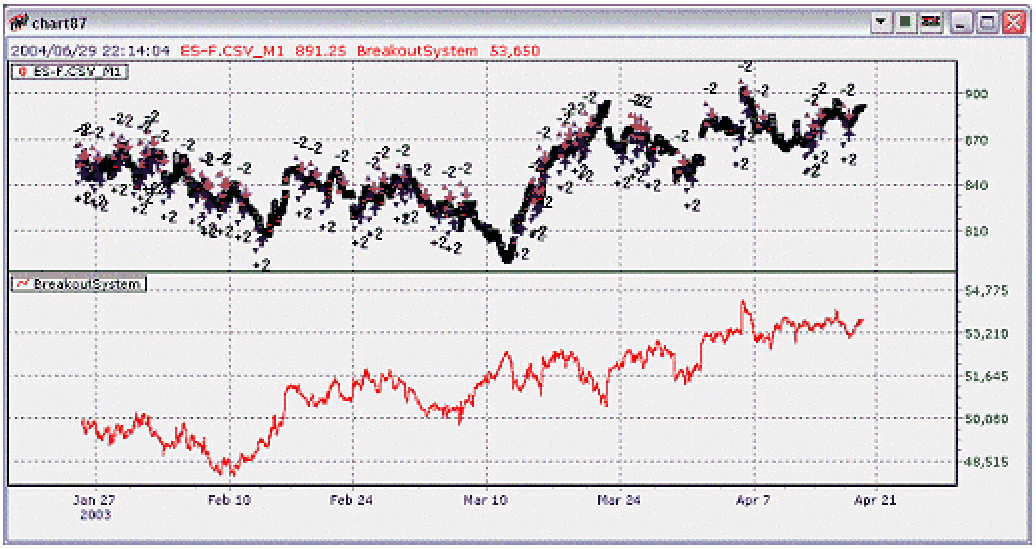
Label	
Tight Range	ADX (data1, 14) < 15
Tight Period	5
BO Value	0.5
Size	1

□ do not close

☑ full dialog

9 Press the **Apply** button. You may want to move the trading system to another pane as the trading system shows the equity curve in plot1.

The following figure shows the trading system over the two month period.



We have now completed the basic system and can proceed to the next step in our research of the breakout model.

Review Performance

Using System Performance Viewer

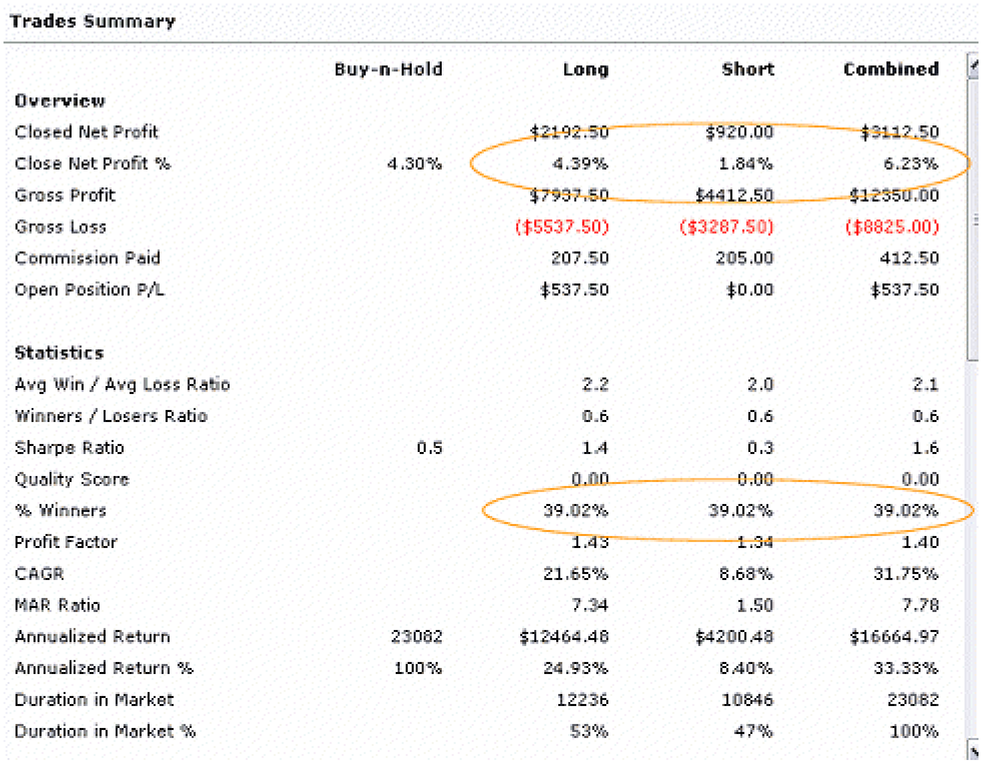
To analyze the system we have just created, we can open the system performance viewer to review its performance.

To open the system performance viewer, click on the legend BreakoutSystem. Right-click to open the indicator pop up menu and choose **Trading System>Open Performance Viewe**r, a new performance viewer window will open.

You can review the various reports of the trading system here. You can also send these reports to the printer or to Excel.

Identifying Possible Improvements

Switch to the Trades Summary page in the performance viewer, you will find that the system we have created is profitable but its percentage winners is low. The amount of profits is not that great either.

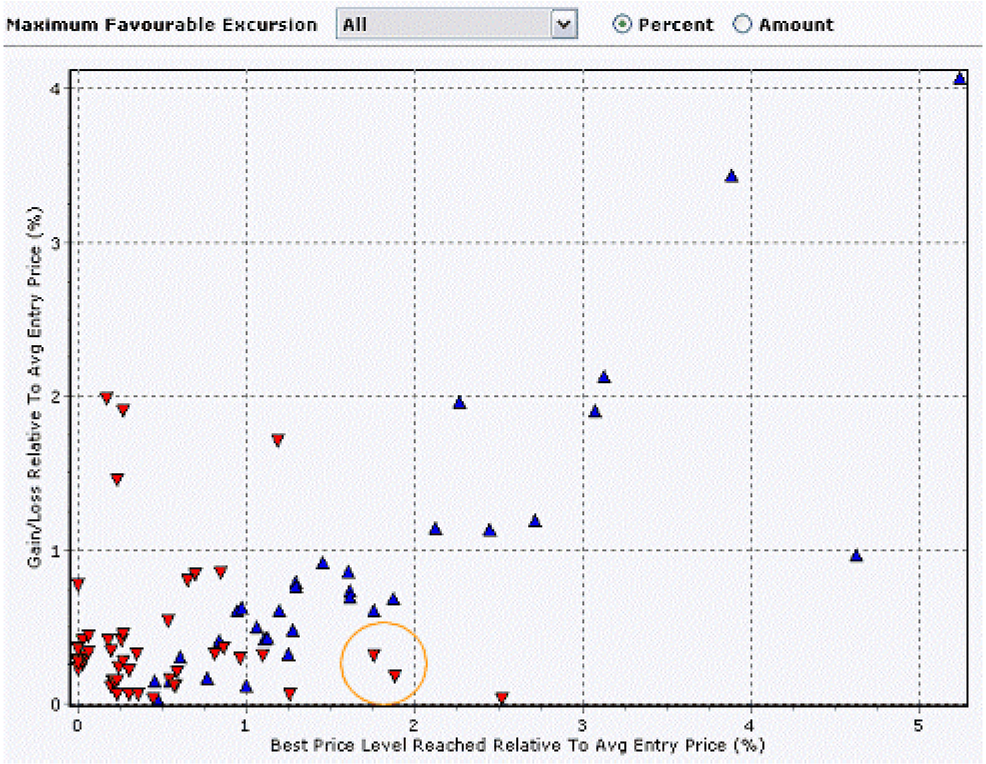


Trades Summary

	Buy-n-Hold	Long	Short	Combined
Overview				
Closed Net Profit		$2192.50	$920.00	$3112.50
Close Net Profit %	4.30%	4.39%	1.84%	6.23%
Gross Profit		$7937.50	$4412.50	$12350.00
Gross Loss		($5537.50)	($3287.50)	($8825.00)
Commission Paid		207.50	205.00	412.50
Open Position P/L		$537.50	$0.00	$537.50
Statistics				
Avg Win / Avg Loss Ratio		2.2	2.0	2.1
Winners / Losers Ratio		0.6	0.6	0.6
Sharpe Ratio	0.5	1.4	0.3	1.6
Quality Score		0.00	0.00	0.00
% Winners		39.02%	39.02%	39.02%
Profit Factor		1.43	1.34	1.40
CAGR		21.65%	8.68%	31.75%
MAR Ratio		7.34	1.50	7.78
Annualized Return	23082	$12464.48	$4200.48	$16664.97
Annualized Return %	100%	24.93%	8.40%	33.33%
Duration in Market		12236	10846	23082
Duration in Market %		53%	47%	100%

We can try to figure out the reason through various position analysis reports to see if we can improve the performance of the system.

One obvious reason can be seen in the Maximum Favorable Excursion report.



Notice the system is taking huge losses that can be avoided if proper money management is used. Take the trades that have been circled for example. They have been making more than 15 points in profit but ends up losing 2 to 3 points.

At this point, it is time to utilize the information we have found here.

Second Attempt

Idea Refinement - Apply Absolute Stop Loss

What we have learnt from the performance report is that there are outrageous losses that should not have happened. If we have straight stop loss for each position, once certain profitability is reached, we can avoid such losses. Thus we will add a new parameter to the system as the stop loss to see if that will improve the system's performance.

Improved System

Here is the improved system.

Two new parameters are added, one for the triggering of the new money management rule and one for the actual protection of profit. New code is added for money management.

```
const hHHV  = 0
const hLLV  = 1

function BreakoutSystem2
dim tightrange, tightrangec, hhv, llv
dim prevorder

   tightrange = itself.makeindicator ("tr", "fml", Array ("1"), 
Array (param1.str))
   tightrangec = itself.makeindicator ("trc", "summation", Array 
("tr"), Array (param2.str))

   hhv = itself.makeindicator ("hhv", "hhv", Array ("1.h"), Array 
(param2.str))
   llv = itself.makeindicator ("llv", "llv", Array ("1.l"), Array 
(param2.str))

   if not tightrangec.valid (0) then
      itself.successall = false
      exit function
   end if

   if heap.size = 0 then
      heap.allocate 2
      heap.fill 0, 1, 0
   end if

   if tightrangec.value (0) >= param2.int then

      if tightrangec.value (1) < param2.int then ' was not in 
tight range, new order
         heap.value (hHHV) = hhv.value (0)
         heap.value (hLLV) = llv.value (0)
         trade.cancelallopenorders
      else ' order was placed
         if hhv.value (0) > heap.value (hHHV) then heap.value 
(hHHV) = hhv.value (0)
         if llv.value (0) < heap.value (hLLV) then heap.value 
(hLLV) = llv.value (0)
      end if

      trade.longstop heap.value (hHHV) + param3.real, param4.int, 
otffillorkill, "L"
      trade.shortstop heap.value (hLLV) - param3.real, param4.int, 
otffillorkill, "S"

   end if

   if trade.openpositionlong and _
      ((trade.openpositionbestpricelevel - 
trade.openpositionentryprice) >= param5.real) then

       trade.longexitstop _
          trade.openpositionentryprice + param6.real, _
          trade.openpositionabssize, otffillorkill, "LX"

   elseif trade.openpositionshort and _
```

```
      ((trade.openpositionentryprice -
trade.openpositionbestpricelevel) >= param5.real) then

       trade.shortexitstop _
           trade.openpositionentryprice - param6.real, _
           trade.openpositionabssize, otffillorkill, "SX"
   end if

   BreakoutSystem2 = trade.currentequity
end function
```

Dissecting the Improved System

The logic of the improved system is largely the same as the original system. The code below is added for long and short exit.

```
   if trade.openpositionlong and _
       ((trade.openpositionbestpricelevel -
   trade.openpositionentryprice) >= param5.real) then

       trade.longexitstop _
           trade.openpositionentryprice + param6.real, _
           trade.openpositionabssize, otffillorkill, "LX"

   elseif trade.openpositionshort and _
       ((trade.openpositionentryprice -
   trade.openpositionbestpricelevel) >= param5.real) then

       trade.shortexitstop _
          trade.openpositionentryprice - param6.real, _
          trade.openpositionabssize, otffillorkill, "SX"
   end if
```

Consider the long side of the system. The exit stop order is enter when the following conditions are true:

- There is open position in the long side
- The difference between the best price of the open position and the entry price is larger than the user specified threshold (`param5`).

Basically, it means that the trading system will issue a stop exit order only after certain profit goal has been reached.

The actual stop exit order will be placed at the entry price plus a protection value (`param6`). Should the price reverses and goes against the trading system, the stop will be hit and filled at market price.

The short side works similarly.

Installing the Improved System

The new system is called `02_BreakoutSystem.VBS`. in the `Samples` folder. Simply open the script and install it. The script will be installed as Breakout System 2.

Running the Improved System

We will apply the second system to the same chart. The parameters are largely the same, with the addition of the money management parameters. As before, if you are testing against your own data, you will need to make appropriate adjustments to the system.

1 Delete the original trading system from the chart.

2 Select the data series, and from pop up menu, choose **Add Indicator**. Choose **Breakout System 2**.

3 Under the **System** tab, set the following trading system parameters:

Price Multiple set to 50

Min Tick Size to 0.25

Enable **Single Entry Per Direction**

Fill Type to **Exact**

Commission Type to **Per Unit**

Commission Base to 0

Commission Per Unit to 2.5

4 Under the **Parameters** tab, set the following parameters for the trading system:

Tight Range to `ADX (data1, 14) < 15`

Tight Period to 5

BO Value to 0.5

Size to 1

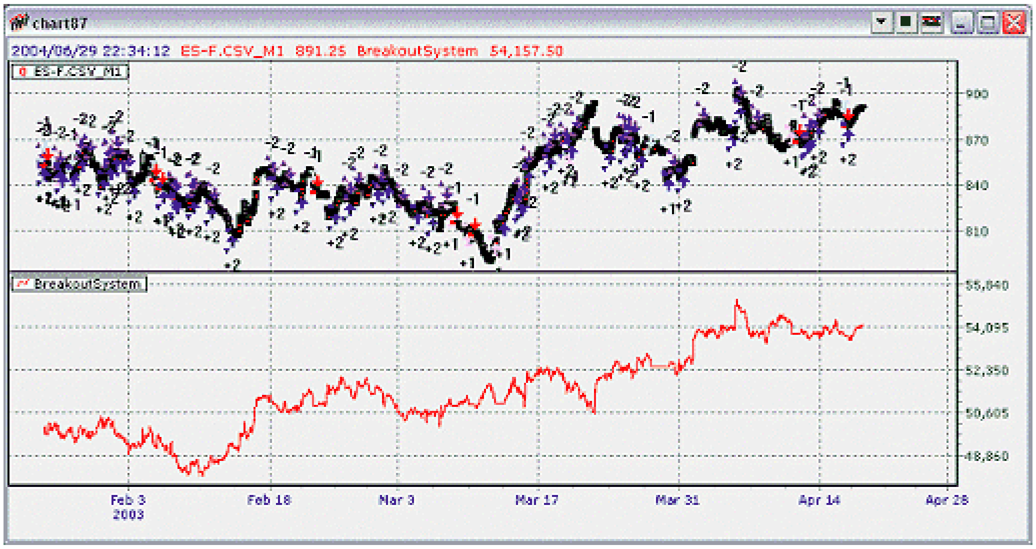
Raise Stop to 8

Stop to 1

Raise Stop is the profit threshold mentioned earlier. In this case, the stop exit order will only be placed if the system has made at least 8 points. **Stop** is profit protection value. In this case, the stop exit happens one point above (for long position) or below (for short position) the entry price.

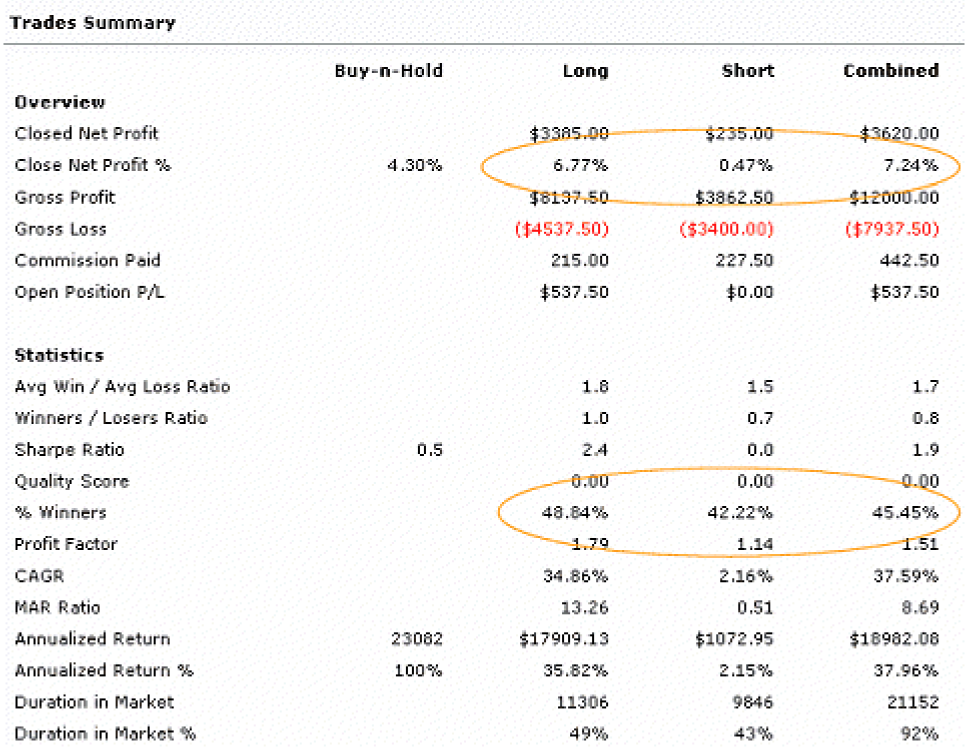
5 Press the **Apply** button to run the system.

The following figure shows the improved breakout system.



Review Performance Again

Select the improved system in the chart and choose **Trading System>Open Performance Viewer** from the pop up menu.



Trades Summary

	Buy-n-Hold	Long	Short	Combined
Overview				
Closed Net Profit		$3385.00	$235.00	$3620.00
Close Net Profit %	4.30%	6.77%	0.47%	7.24%
Gross Profit		$8137.50	$3862.50	$12000.00
Gross Loss		($4537.50)	($3400.00)	($7937.50)
Commission Paid		215.00	227.50	442.50
Open Position P/L		$537.50	$0.00	$537.50
Statistics				
Avg Win / Avg Loss Ratio		1.8	1.5	1.7
Winners / Losers Ratio		1.0	0.7	0.8
Sharpe Ratio	0.5	2.4	0.0	1.9
Quality Score		0.00	0.00	0.00
% Winners		48.84%	42.22%	45.45%
Profit Factor		1.79	1.14	1.51
CAGR		34.86%	2.16%	37.59%
MAR Ratio		13.26	0.51	8.69
Annualized Return	23082	$17909.13	$1072.95	$18982.08
Annualized Return %	100%	35.82%	2.15%	37.96%
Duration in Market		11306	9846	21152
Duration in Market %		49%	43%	92%

If you compared the report to that of the original system, you can see that the percentage winner has been improved by about 5% and Close Net Profit % has been improved by about 1%.

The improved system actually does worse in the short side. You can further improve the performance by fine tuning the short side of the system.

Summary

Creating trading systems is about the identification and utilization of repeated market behaviour. Signals and filters may not be trivial. Thus time spent on studying the charts and testing out trading ideas will certainly help improve your trading performance even though the research result (in this case a trading system) may not be used as is in your trading. In this case we have shown that you can certainly improve your trading if proper money management rule is added to your winning positions.

Tutorial: Writing a Portfolio Trading System

This tutorial shows you how to write a portfolio trading system.

You may want to look at the tutorial on writing a single instrument trading system if you are not already familiar with how NeoTicker®'s trading system model works. (See ***Tutorial: Writing a Breakout Trading System*** (on page 1397)).

Overview

Consider the items in a typical single instrument trading system:

- A chart with the data to be tested.
- A trading system that implements the trading strategy, applied on the data.
- Performance viewer for analyzing trading system performance.

A portfolio trading system is not much different. It is a generalization of the single instrument model:

- A chart with the data to be tested. In portfolio testing, multiple data series are added to the chart.
- A trading system that implements the trading strategy, applied on the data. The trading strategy analyzes multiple data series.
- Performance viewer for analyzing portfolio system performance.

NeoTicker®'s portfolio trading system architecture has the following properties:

- All instruments are available for analysis all the time within the trading system. There is no need to switch instrument.
- Order mechanism is transaction based. Positions can be tracked for individual instruments.
- Portfolio concept is encapsulated in a chart. You can have multiple symbols and indicators in multiple charts. So real-time setup for deploying multiple portfolio trading systems is highly flexible.

Idea

In this tutorial, we will create a portfolio moving average crossover system.

In a single instrument moving average crossover system, the instrument is traded when a crossover happens, and the position is reversed when the crossover reversed.

The portfolio version will look at moving average crossover for all symbols in the portfolio and make trading decisions based on crossovers. The objectives are:

- A portfolio of 100 stocks (Nasdaq 100)
- Go into position when a crossover happens with a symbol in the portfolio
- We want to limit the number of open positions to 1, i.e. the system does not simultaneous longi/short more than 1 symbol.

When you design a portfolio trading system, you will face the following challenges:

- Accessing arbitrary data series, and applying analysis to the data series
- Loading and constraining data for testing

This tutorial will start with writing a simple script, and evolving the script into a portfolio trading system. You will learn how to tackle these problems along the way.

The trading system here is written in Delphi Script. Because NeoTicker® provides a uniform interface to all scripting languages and IDL interface. The same concept can be easily translated to other languages.

To save you some typing, all example scripts used in this tutorial is available in the NeoTicker® installation under the `Sample\Portfolio Trading System` folder, e.g. `C:\Program Files\TickQuest\NeoTicker3\Samples\Portfolio Trading System`..

Moving Average Prototype

We start by writing a script that calculates two simple moving averages for a single symbol. The goal here is to write it in a such a way that the code can be easily reused with a portfolio of 100 symbols.

Referencing Arbitrary Indicator

When you create an indicator such as moving average in script, you will need to supply a name. This name is later used for referencing the indicator. For single symbol analysis, any arbitrary name will suffice.

For a portfolio, what we need is a systematic way of creating indicator names so that large number of indicators can be easily referenced. We can achieve this goal by generating indicator name using a function.

Notice that any indicator can be uniquely identified by three attributes:

1. A name that represents the type of the indicator (e.g. moving average).
2. Data source. In a portfolio, this can be a index referencing the symbol in the portfolio.
3. Parameters. In the case of simple moving average, there is only one parameter (period).

Here is the function for creating a unique name for an indicator in a portfolio:

```
// indicator_name constructs a name that references
// a created indicator, based an index value (idx)
// and a period.
//
// We want to created the name programmatically because
// this system is going to be generalized into
// a portfolio system. It is not desirable to
// hard code the indicator name
function indicator_name(suffix : String; idx : integer; period :
integer) : String;
begin
    result := suffix + '_' +
              NTLib.integer2str(idx) + '_' +
              NTLib.integer2str(period);
end;
```

DataSeries Objects

Consider the data source for the moving average indicators.

When you are dealing with a few symbols, it is a good idea to use the Data1, Data2, or LinkSeries objects to reference the data series. These objects have advantage of good tick updates. When you use these objects, data series must be linked explicitly to the indicator. Imagine a portfolio of 100 stocks, explicitly linking 100 stocks to the indicator is a hassle.

Instead, a better solution is to use the DataSeries object. DataSeries can access arbitrary data series in a chart without explicit linkage.

Note that when you make indicators for DataSeries object, you need to use # notation, e.g. #1 refers to the first data series, #2 refers to the second data series, and so on. It is a good idea to have a function that constructs the # notation:

```
// Data series index creates a string index into a
// data series (not necessarily linked). This index
// is used for creating indicator for the data series
//
// We want to use non-linked series because this system
// is going to address many symbols in a portfolio. It
// is not desirable to use a linked series in this case.
function dataseries_index(idx : integer) : String;
begin
    result := '#' + NTLib.integer2str(idx);
end;
```

For more information about DataSeries object, refer to ***Accessing Non-Linked Data Series in the Chart*** (on page 1448).

Indicator

Here is the complete indicator code for moving average indicators (portsystem_ma_prototype.pas in the sample folder).

Notice that the code uses the following techniques to access arbitrary data and indicators:

- the variable idx for referencing arbitrary data series.
- dataseries_index to generate # notation when creating indicator.
- indicator_name for referencing arbitrary indicator.

The main entry point for the indicator is the portsystem function.

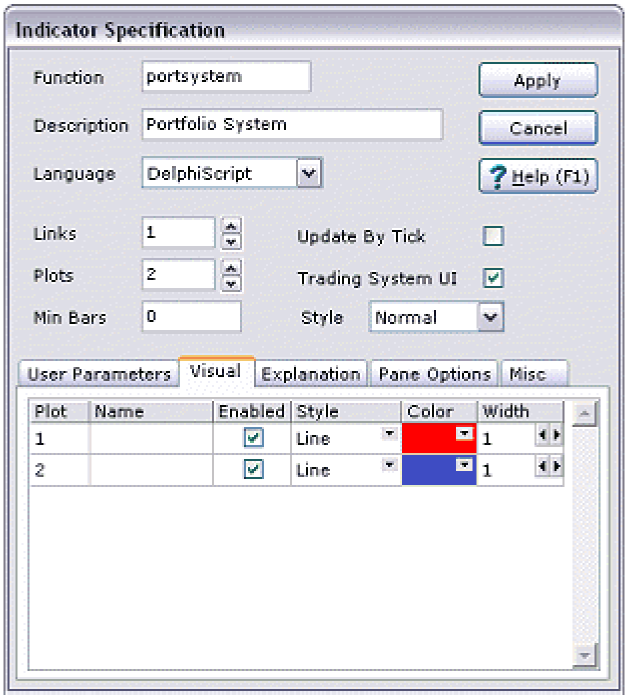
```
// indicator_name constructs a name that references
// a created indicator, based an index value (idx)
// and a period.
//
// We want to created the name programmatically because
// this system is going to be generalized into
// a portfolio system. It is not desirable to
// hard code the indicator name
function indicator_name(suffix : String; idx : integer; period :
```

```
integer) : String;
begin
    result := suffix + '_' +
              NTLib.integer2str(idx) + '_' +
              NTLib.integer2str(period);
end;

// Data series index creates a string index into a
// data series (not necessarily linked). This index
// is used for creating indicator for the data series
//
// We want to use non-linked series because this system
// is going to address many symbols in a portfolio. It
// is not desirable to use a linked series in this case.
function dataseries_index(idx : integer) : String;
begin
    result := '#' + NTLib.integer2str(idx);
end;

function portsystem : double;
var
    idx : integer;
    period1, period2 : integer;
    name1, name2 : String;
begin
   // Index setup
   idx := 1;

   // First moving average, 10-period
   period1 := 10;
   name1 := indicator_name('MA', idx, period1);
   ItSelf.MakeIndicator(
       name1, 'average',
       [dataseries_index(idx)],
       [NTLib.integer2str(period1)]);

   // Second moving average, 30-period
   period2 := 30;
   name2 := indicator_name('MA', idx, period2);
   ItSelf.MakeIndicator(
       name2, 'average',
       [dataseries_index(idx)],
       [NTLib.integer2str(period2)]);

   // In this prototype, we will use plot1, plot2 to
   // verify moving average creation.

   // Check validity of first moving average, if valid, return it
   if ItSelf.Indicator(name1).valid[0] then
   begin
       ItSelf.Answer[1] := ItSelf.indicator(name1).value[0];
   end
   else
   begin
       ItSelf.SuccessEx[1] := false;
   end;
   // Check validity of second moving average, if valid, return it
   if ItSelf.Indicator(name2).valid[0] then
   begin
       ItSelf.Answer[2] := ItSelf.indicator(name2).value[0];
   end
   else
   begin
       ItSelf.SuccessEx[2] := false;
   end;
end;
```

Installing

If you are going to install this indicator, here are the settings when you create the indicator:

- Function name set to `portsystem`
- Language set to `Delphi Script`
- Enable `Trading System UI`
- Make this a 2-plot indicator



Indicator Specification

Function: portsystem

Description: Portfolio System

Language: DelphiScript

Links: 1

Plots: 2

Min Bars: 0

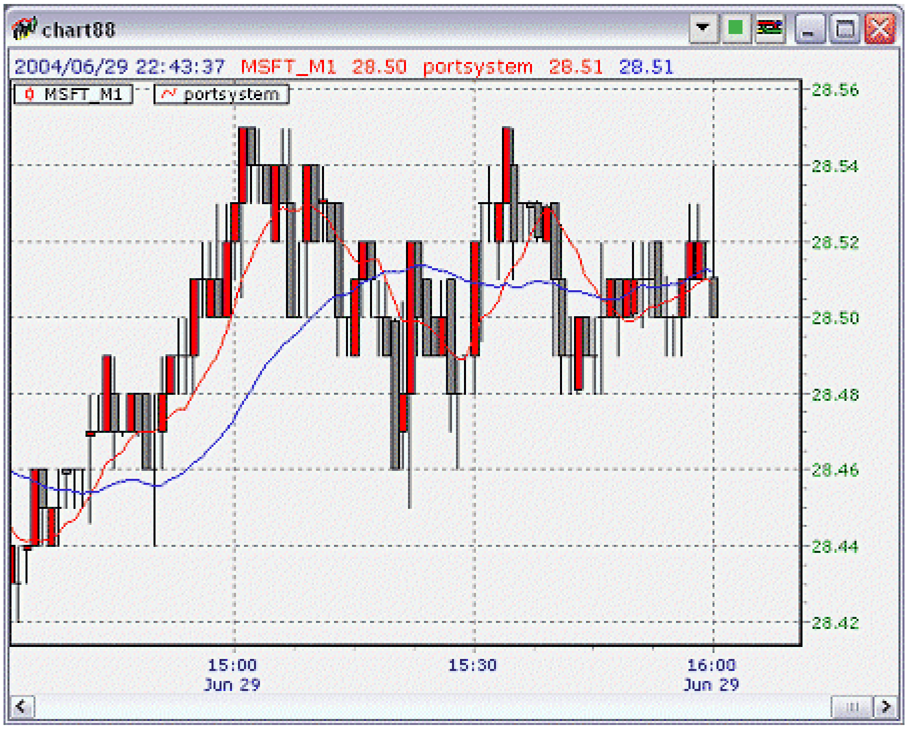
Update By Tick □

Trading System UI ☑

Style: Normal

Plot	Name	Enabled	Style	Color	Width
1		☑	Line	🟥	1
2		☑	Line	🟦	1

This indicator outputs 2 simple moving averages (red line is 10-period, blue line is 30 period). The script is a rather cumbersome way for creating moving average, but remember the goal here to lay down a framework for a portfolio trading system.



Crossover Prototype

We will extend the moving average prototype for detecting moving average crossover. Our goal is to write the code such that the crossover detection can be reused with portfolio.

Indicator-on-Indicator

To compute crossover for the moving averages, we can apply cross above and cross below indicators on the moving averages.

Recall that in the moving average prototype, the moving average indicator names are created using `indicator_name` function, then the names are assigned to the variables `name1` and `name2`. To apply crossover on the moving averages, we can simply specify `name1` and `name2` as the data source for crossover:

```
   // create crossabove and crossbelow indicators, based on the
   // moving averages
   crossabovename := indicator_name('CA', idx, 0);
   ItSelf.MakeIndicator(
       crossabovename, 'xabove',
       [name1, name2],
       ['']);

   crossbelowname := indicator_name('CB', idx, 0);
   ItSelf.MakeIndicator(
       crossbelowname, 'xbelow',
       [name1, name2],
       ['']);
```

Consider in a portfolio, we will create many crossover indicators. To reference the crossover indicators, `indicator_name` function is reused to construct the indicator names for the crossover.

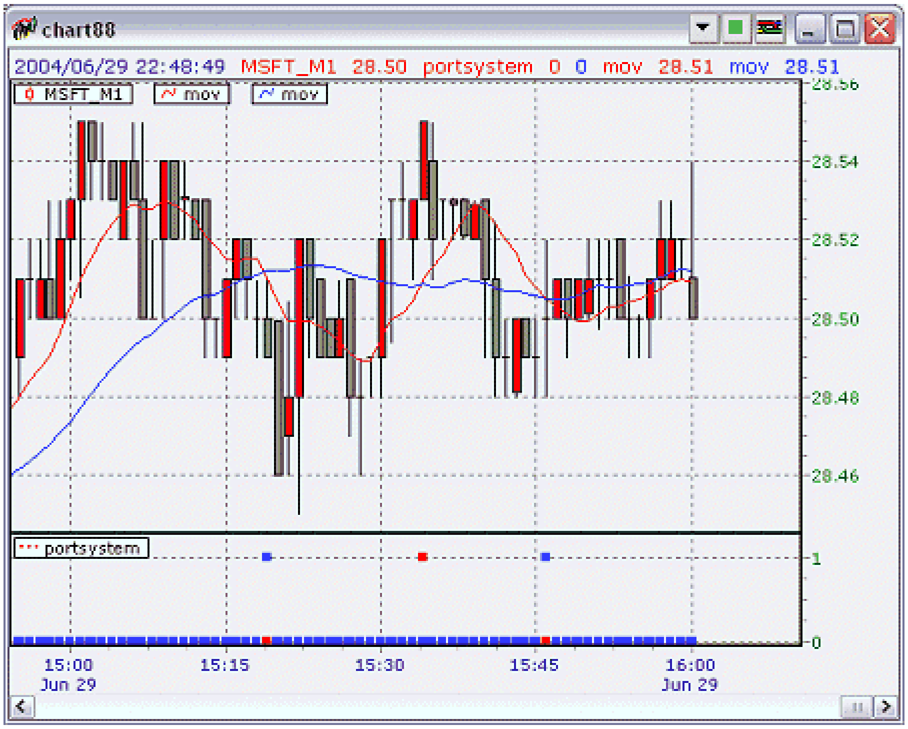
Indicator

Here is the complete script (`portsystem_crossover_prototype.pas` in the sample folder):

```
// indicator_name constructs a name that references
// a created indicator, based an index value (idx)
// and a period.
//
// We want to created the name programmatically because
// this system is going to be generalized into
// a portfolio system. It is not desirable to
// hard code the indicator name
function indicator_name(suffix : String; idx : integer; period :
integer) : String;
begin
    result := suffix + '_' +
              NTLib.integer2str(idx) + '_' +
              NTLib.integer2str(period);
end;

// Data series index creates a string index into a
// data series (not necessarily linked). This index
// is used for creating indicator for the data series
//
// We want to use non-linked series because this system
// is going to address many symbols in a portfolio. It
// is not desirable to use a linked series in this case.
function dataseries_index(idx : integer) : String;
begin
    result := '#' + NTLib.integer2str(idx);
end;

function portsystem : double;
var
    idx : integer;
    period1, period2 : integer;
    name1, name2 : String;
    crossabovename, crossbelowname : String;
begin
   // Index setup
   idx := 1;

   // First moving average, 10-period
   period1 := 10;
   name1 := indicator_name('MA', idx, period1);
   ItSelf.MakeIndicator(
       name1, 'average',
       [dataseries_index(idx)],
       [NTLib.integer2str(period1)]);

   // Second moving average, 30-period
   period2 := 30;
   name2 := indicator_name('MA', idx, period2);
   ItSelf.MakeIndicator(
       name2, 'average',
       [dataseries_index(idx)],
       [NTLib.integer2str(period2)]);

   // create crossabove and crossbelow indicators, based on the
   // moving averages
   crossabovename := indicator_name('CA', idx, 0);
   ItSelf.MakeIndicator(
       crossabovename, 'xabove',
       [name1, name2],
       ['']);

   crossbelowname := indicator_name('CB', idx, 0);
   ItSelf.MakeIndicator(
       crossbelowname, 'xbelow',
       [name1, name2],
       ['']);

   // In this prototype, we will use plot1, plot2 to
   // verify crossover

   // Check validity of crossabove
   if ItSelf.Indicator(crossabovename).valid[0] then
   begin
       ItSelf.Answer[1] :=
ItSelf.indicator(crossabovename).value[0];
   end
   else
   begin
       ItSelf.SuccessEx[1] := false;
   end;

   // Check validity of crossbelow
   if ItSelf.Indicator(crossbelowname).valid[0] then
   begin
       ItSelf.Answer[2] :=
ItSelf.indicator(crossbelowname).value[0];
   end
   else
   begin
       ItSelf.SuccessEx[2] := false;
   end;
end;
```

Below is a screen shot. The lines in the top pane are moving averages added by hand for reference. They are not part of the script.

The script resides in the lower pane. We change the appearance of the indicator from lines to dots for better viewing. Red dots at 1 are a cross above signals and blue dots at 1 are cross below signals.



Single Instrument Prototype

We will extend the crossover prototype to trade single instrument. The system longs when a cross above is detected, and shorts when a cross below is detected. The goal is to use generalized trading methods that can be reused to trade arbitrary data series in a portfolio.

Trading Methods

Below is the code that performs the trading:

```
  if ItSelf.Indicator(crossabovename).valid[0] and
     (ItSelf.Indicator(crossabovename).value[0] > 0) then
  begin
      // Long
      Trade.LongAtMarketEx(idx, 100, '');
  end
  else if ItSelf.Indicator(crossbelowname).valid[0] and
     (ItSelf.Indicator(crossbelowname).value[0] > 0) then
```

```
   begin
       // Short
       Trade.ShortAtMarketEx(idx, 100, '');
   end;

   // Plot1 reports equity curve
   ItSelf.Answer[1] := Trade.currentequity;
```

The code uses `LongAtMarketEx` and `ShortAtMarketEx` for entering position. These methods behaves like their single instrument counterpart `LongAtMarket` and `ShortAtMarket`. The difference is `LongAtMarketEx` and `ShortAtMarketEx` accepts an extra parameter that specifies the instrument to trade. Here you can see that the system uses the `idx` variable to identify the instrument.

There is no generalize version for the `currentequity` property. `Currentequity` automatically detects a portfolio trading system and reports equity accordingly.

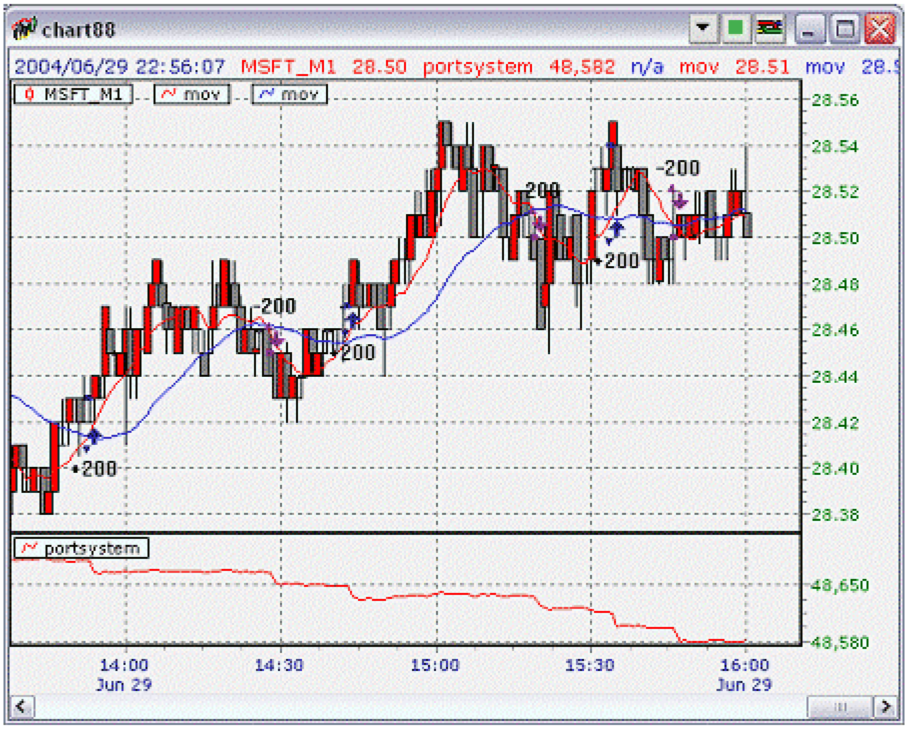
System

Here is the complete script for the system (`portsystem_single_instrument.pas` in the sample folder):

```
// indicator_name constructs a name that references
// a created indicator, based an index value (idx)
// and a period.
//
// We want to created the name programmatically because
// this system is going to be generalized into
// a portfolio system. It is not desirable to
// hard code the indicator name
function indicator_name(suffix : String; idx : integer; period :
integer) : String;
begin
    result := suffix + '_' +
              NTLib.integer2str(idx) + '_' +
              NTLib.integer2str(period);
end;

// Data series index creates a string index into a
// data series (not necessarily linked). This index
// is used for creating indicator for the data series
//
// We want to use non-linked series because this system
// is going to address many symbols in a portfolio. It
// is not desirable to use a linked series in this case.
function dataseries_index(idx : integer) : String;
begin
    result := '#' + NTLib.integer2str(idx);
end;

function portsystem : double;
var
    idx : integer;
    period1, period2 : integer;
    name1, name2 : String;
    crossabovename, crossbelowname : String;
begin
   // Index setup
   idx := 1;

   // First moving average, 10-period
   period1 := 10;
   name1 := indicator_name('MA', idx, period1);
   ItSelf.MakeIndicator(
       name1, 'average',
       [dataseries_index(idx)],
       [NTLib.integer2str(period1)]);

   // Second moving average, 30-period
   period2 := 30;
   name2 := indicator_name('MA', idx, period2);
   ItSelf.MakeIndicator(
       name2, 'average',
       [dataseries_index(idx)],
       [NTLib.integer2str(period2)]);

   // create crossabove and crossbelow indicators, based on the
   // moving averages
   crossabovename := indicator_name('CA', idx, 0);
   ItSelf.MakeIndicator(
       crossabovename, 'xabove',
       [name1, name2],
       ['']);

   crossbelowname := indicator_name('CB', idx, 0);
   ItSelf.MakeIndicator(
       crossbelowname, 'xbelow',
       [name1, name2],
       ['']);

   // In this prototype, we will use plot1 to display
   // equity curve, plot2 is not used

   if ItSelf.Indicator(crossabovename).valid[0] and
      (ItSelf.Indicator(crossabovename).value[0] > 0) then
   begin
       // Long
       Trade.LongAtMarketEx(idx, 100, '');
   end
   else if ItSelf.Indicator(crossbelowname).valid[0] and
      (ItSelf.Indicator(crossbelowname).value[0] > 0) then
   begin
       // Short
       Trade.ShortAtMarketEx(idx, 100, '');
   end;

   // Plot1 reports equity curve
   ItSelf.Answer[1] := Trade.currentequity;

   // Plot2 is not used
   ItSelf.SuccessEx[2] := false;
```

```
end;
```

Below is a screen shot. The equity curve is shown in the bottom pane. The moving average lines are added manually for reference only. They are not part of the trading system.



Handling Portfolio Data

In this step, we will show you how to quickly create a chart and load it up with portfolio data.

Creating a Single Data Series as a Template

We will create a single data series in a chart. This data series will act as a template later to load all the symbols in the portfolio. The goal is to setup the chart such that it is easy to control date range for the portfolio.

1 Starts with a blank chart. Then add a single data series to the chart.

2 Select the data series, then press space bar to edit the data series. Under the **Misc** tab, make sure the **Use Chart Default** option is chosen for **Days to Load** option. Then close the data editor. Reason: we want to make sure individual item in the portfolio does not override the chart's days to load setting.

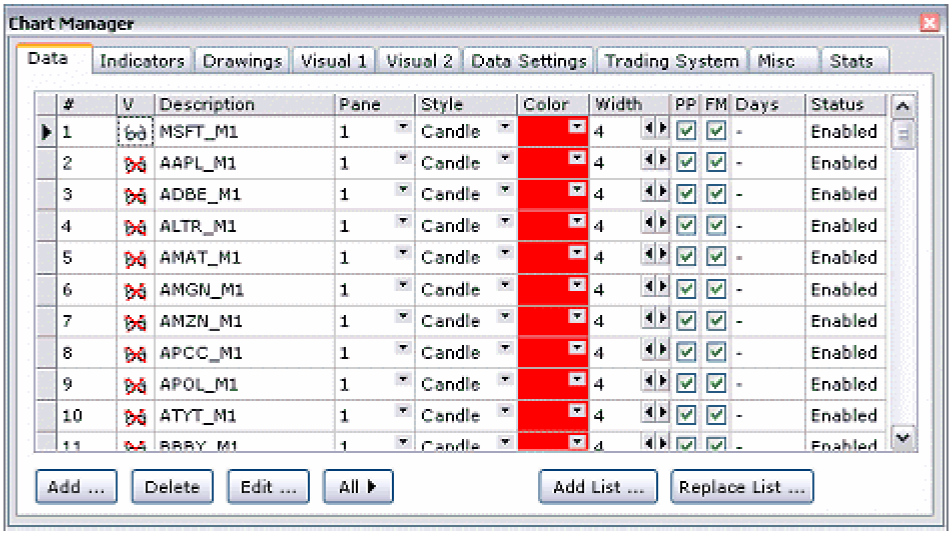
3 Open Chart Manager, under **Data Settings** tab, set **Last Day** option to **Specific Day**. Reason: use this setting for historical testing. It allows precise date control for the testing. For real-time deployment of a system, use **Most Recent Data**.

Loading Portfolio Data

We will use the add list feature in Chart Manager to load a portfolio into a chart. This feature adds the symbols from a symbol list to a chart. In this case the symbol list contains the symbols of the portfolio. We will be using the built-in Nasdaq 100 symbol list, but if you want to create your own symbol list, you can consult ***Tutorial: Creating a Symbol List*** (on page 155).

1. Open Chart Manager, press **Data** tab. You should have only one data series in the table.
2. Press the **Add List** button.
3. Choose the Nasdaq 100 symbol list. This will take a few minutes to load for the first time. Subsequent loading will be much faster because the data will be cached.

At this stage, you chart will have 101 data series (first data series + Nasdaq 100 symbols) ready for portfolio testing. The chart will look pretty bad with so many data series in it. You can use the **All>Hide** button in Chart Manager to first turn off the visibility of all data series, then turn on the visibility of only the first data series.



#	V	Description	Pane	Style	Color	Width	PP	FM	Days	Status
1	👓	MSFT_M1	1	Candle	🟥	4	☑	☑	-	Enabled
2	✗	AAPL_M1	1	Candle	🟥	4	☑	☑	-	Enabled
3	✗	ADBE_M1	1	Candle	🟥	4	☑	☑	-	Enabled
4	✗	ALTR_M1	1	Candle	🟥	4	☑	☑	-	Enabled
5	✗	AMAT_M1	1	Candle	🟥	4	☑	☑	-	Enabled
6	✗	AMGN_M1	1	Candle	🟥	4	☑	☑	-	Enabled
7	✗	AMZN_M1	1	Candle	🟥	4	☑	☑	-	Enabled
8	✗	APCC_M1	1	Candle	🟥	4	☑	☑	-	Enabled
9	✗	APOL_M1	1	Candle	🟥	4	☑	☑	-	Enabled
10	✗	ATYT_M1	1	Candle	🟥	4	☑	☑	-	Enabled
11	✗	BBBY_M1	1	Candle	🟥	4	☑	☑	-	Enabled

Final Portfolio System

In this step, we will extend the script created in ***Single Instrument Prototype*** (on page 1422) to a portfolio trading system, and apply the system to the chart that contains portfolio data created in ***Handling Portfolio Data*** (on page 1425).

Indexing

Recall that the chart we created has 101 symbols. The first symbol is the template for loading the portfolio. So the first symbol should not be included in the portfolio analysis. When you work on a portfolio, it is a nice practice to define the range of data series that constitute the portfolio:

```
  // Define the data series that is within the portfolio.
  // Indicators are created for all symbols in the portfolio.
```

```
   // We skip the first data series because it is used mainly
   // as a template for loading data.
   // All other data series are part of the portfolio

   portfolio_start := 2;
   portfolio_end := DataSeries.Count; // You can lower this number
                                      // when you are debugging

   // Create indicators for all symbols in the chart.
   for idx := portfolio_start to portfolio_end do
```

The `idx` variable is now part of the for loop. Thus when the system creates indicators, it will create indicators for all symbols in the portfolio.

Making Indicators

The `MakeIndicator` calls are identical to those in earlier prototypes. For example, below is the code for creating the 10-period moving average:

```
       // First moving average, 10-period
       period1 := 10;
       name1 := indicator_name('MA', idx, period1);
       ItSelf.MakeIndicator(
           name1, 'average',
           [dataseries_index(idx)],
           [NTLib.integer2str(period1)]);
```

Detecting Crossovers

The system uses a for loop over the portfolio to detect crossover. The system does not trade right away when a crossover is detected. Recall that one of the portfolio system's objective is trading one symbol at a time. So once a crossover is detected, it is recorded. We will only go into position after we close an old position first.

```
   // Scan for a crossover in the portfolio.  We will only pickup
the first
   // crossover within this bar.  If there are more than 1, the
2nd, 3rd
   // crossovers that happen in the current bar will be ignored

   current_bar_crossover_count := 0;

   for idx := portfolio_start to portfolio_end do
   begin
       crossabovename := indicator_name('CA', idx, 0);
       crossbelowname := indicator_name('CB', idx, 0);

       if ItSelf.Indicator(crossabovename).valid[0] and
          (ItSelf.Indicator(crossabovename).value[0] > 0) then
       begin
           // Record idx for entering position
           current_bar_crossover_count :=
current_bar_crossover_count + 1;
           current_bar_crossover_idx := idx;
       end
       else if ItSelf.Indicator(crossbelowname).valid[0] and
               (ItSelf.Indicator(crossbelowname).value[0] > 0)
```

```
then
       begin
           // Record idx for entering position
           current_bar_crossover_count :=
current_bar_crossover_count + 1;
           current_bar_crossover_idx := idx;
       end;

       // We've found all we need
       if current_bar_crossover_count >= 1 then break;
   end;
```

Closing Old Position

Once a crossover is detected, the system scans for an old position.

```
       // Now we have candidates for new position, check our
position history.
       // Close the open positions if necessary to make places for
new positions

       for idx := portfolio_start to portfolio_end do
       begin
           // Do we have a position for this item in the portfolio
           if not Trade.OpenPositionFlatEx[idx] then
           begin
               Trade.ExitCurrentPositionEx(idx, 'Covers previous
position');
               break;
           end;
       end;
```

Here the system uses `OpenPositionFlatEx` to detect open positions for a data series. When detected, the system uses `ExitCurrentPositionEx` to close out the position. Both methods are generalization of single instrument trading counterparts.

Trading

The trading code is virtually identical to the single instrument prototype, except the data series now points to the data series where the crossover happens (`current_bar_crossover_idx`).

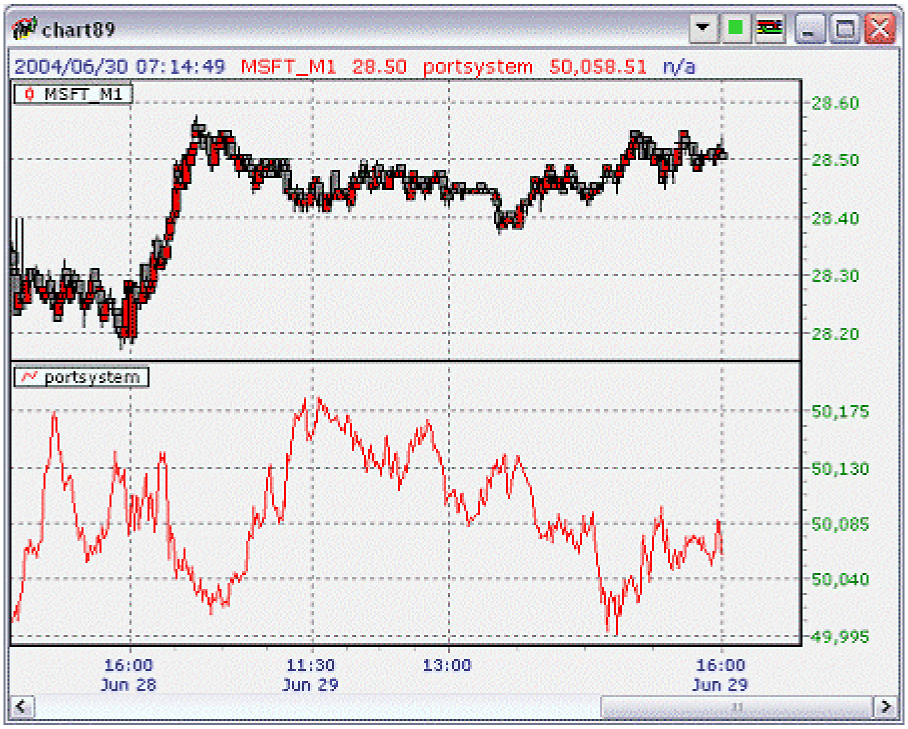
```
       // Make trade
       crossabovename := indicator_name('CA',
current_bar_crossover_idx, 0);
       crossbelowname := indicator_name('CB',
current_bar_crossover_idx, 0);

       if ItSelf.Indicator(crossabovename).valid[0] and
           (ItSelf.Indicator(crossabovename).value[0] > 0) then
       begin
           // Long
           Trade.LongAtMarketEx(current_bar_crossover_idx, 100,
'Crossabove long');
       end
       else if ItSelf.Indicator(crossbelowname).valid[0] and
           (ItSelf.Indicator(crossbelowname).value[0] > 0) then
       begin
           // Short
           Trade.ShortAtMarketEx(current_bar_crossover_idx, 100,
'Crosbelow short');
       end;
```

System

Here is the complete system (`portsystem_multiple_instrument.pas` in sample folder).

```
// indicator_name constructs a name that references
// a created indicator, based an index value (idx)
// and a period.
//
// We want to created the name programmatically because
// this system is going to be generalized into
// a portfolio system. It is not desirable to
// hard code the indicator name
function indicator_name(suffix : String; idx : integer; period :
integer) : String;
begin
    result := suffix + '_' +
              NTLib.integer2str(idx) + '_' +
              NTLib.integer2str(period);
end;

// Data series index creates a string index into a
// data series (not necessarily linked). This index
// is used for creating indicator for the data series
//
// We want to use non-linked series because this system
// is going to address many symbols in a portfolio. It
// is not desirable to use a linked series in this case.
function dataseries_index(idx : integer) : String;
begin
    result := '#' + NTLib.integer2str(idx);
end;

function portsystem : double;
var
    idx : integer;
    period1, period2 : integer;
    name1, name2 : String;
    crossabovename, crossbelowname : String;

    current_bar_crossover_count : integer;
    current_bar_crossover_idx : integer;

    portfolio_start, portfolio_end : integer;
begin
   // Define the data series that is within the portfolio.
   // Indicators are created for all symbols in the portfolio.
   // We skip the first data series because it is used mainly
   // as a template for loading data.
   // All other data series are part of the portfolio
   portfolio_start := 2;
   portfolio_end := DataSeries.Count; // You can lower this number
                                      // when you are debugging

   // Create indicators for all symbols in the chart.
   for idx := portfolio_start to portfolio_end do
   begin
       // First moving average, 10-period
       period1 := 10;
       name1 := indicator_name('MA', idx, period1);
       ItSelf.MakeIndicator(
           name1, 'average',
           [dataseries_index(idx)],
           [NTLib.integer2str(period1)]);

       // Second moving average, 30-period
       period2 := 30;
       name2 := indicator_name('MA', idx, period2);
       ItSelf.MakeIndicator(
           name2, 'average',
           [dataseries_index(idx)],
           [NTLib.integer2str(period2)]);

       // create crossabove and crossbelow indicators, based on
the
       // moving averages
       crossabovename := indicator_name('CA', idx, 0);
       ItSelf.MakeIndicator(
           crossabovename, 'xabove',
           [name1, name2],
           ['']);

       crossbelowname := indicator_name('CB', idx, 0);
       ItSelf.MakeIndicator(
           crossbelowname, 'xbelow',
           [name1, name2],
           ['']);
   end;

   // Scan for a crossover in the portfolio.  We will only pickup
the first
   // crossover within this bar.  If there are more than 1, the
2nd, 3rd
   // crossovers that happen in the current bar will be ignored

   current_bar_crossover_count := 0;

   for idx := portfolio_start to portfolio_end do
   begin
       crossabovename := indicator_name('CA', idx, 0);
       crossbelowname := indicator_name('CB', idx, 0);

       if ItSelf.Indicator(crossabovename).valid[0] and
          (ItSelf.Indicator(crossabovename).value[0] > 0) then
       begin
           // Record idx for entering position
           current_bar_crossover_count :=
current_bar_crossover_count + 1;
           current_bar_crossover_idx := idx;
       end
       else if ItSelf.Indicator(crossbelowname).valid[0] and
               (ItSelf.Indicator(crossbelowname).value[0] > 0)
then
       begin
           // Record idx for entering position
           current_bar_crossover_count :=
current_bar_crossover_count + 1;
           current_bar_crossover_idx := idx;
       end;

       // We've found all we need
       if current_bar_crossover_count >= 1 then break;
   end;

   if current_bar_crossover_count > 0 then
   begin
       // Now we have candidates for new position, check our
position history.
       // Close the open positions if necessary to make places for
new positions

       for idx := portfolio_start to portfolio_end do
       begin
           // Do we have a position for this item in the portfolio
           if not Trade.OpenPositionFlatEx[idx] then
           begin
               Trade.ExitCurrentPositionEx(idx, 'Covers previous
position');
               break;
           end;
       end;

       // Make trade
       crossabovename := indicator_name('CA',
current_bar_crossover_idx, 0);
       crossbelowname := indicator_name('CB',
current_bar_crossover_idx, 0);

       if ItSelf.Indicator(crossabovename).valid[0] and
           (ItSelf.Indicator(crossabovename).value[0] > 0) then
       begin
           // Long
           Trade.LongAtMarketEx(current_bar_crossover_idx, 100,
'Crossabove long');
       end
       else if ItSelf.Indicator(crossbelowname).valid[0] and
           (ItSelf.Indicator(crossbelowname).value[0] > 0) then
       begin
           // Short
           Trade.ShortAtMarketEx(current_bar_crossover_idx, 100,
'Crosbelow short');
       end;
   end;

   // Plot1 reports equity curve
   ItSelf.Answer[1] := Trade.currentequity;

   // Plot2 is not used
   ItSelf.SuccessEx[2] := false;
end;
```

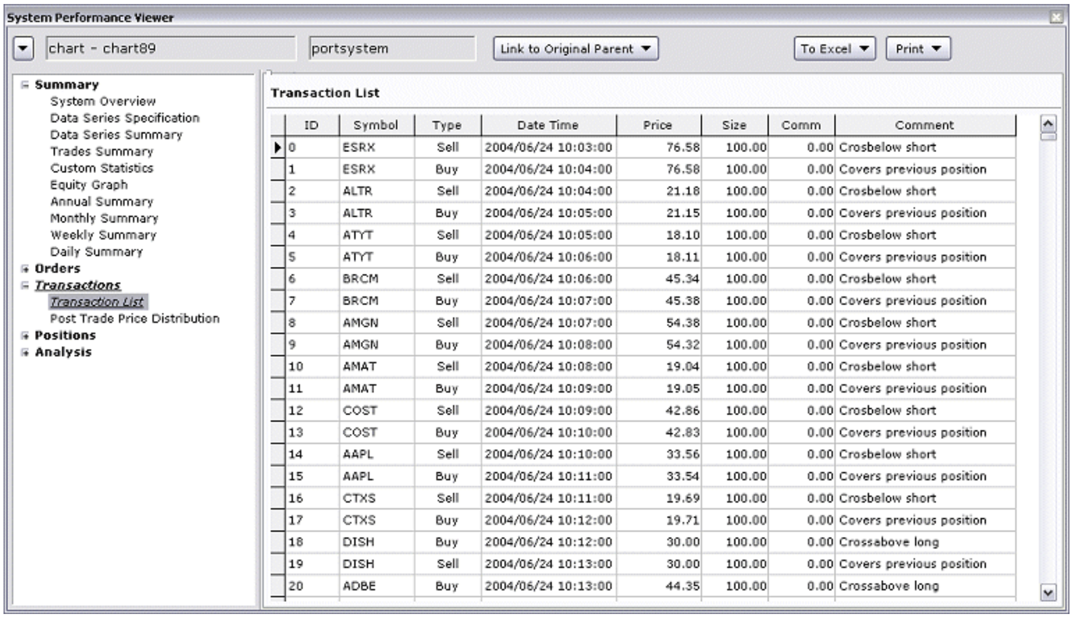
When you run this system, you can simply apply the system on the first data series in the chart. The following chart shows the portfolio trading system. You need to set a low commission as this system trades actively.



Performance Viewer

Select the portfolio system in the chart, right click to open pop up menu, and choose **Trading System>Open Performance Viewer**.

Performance Viewer is compatible with portfolio trading system. For example, in the position list report, you can see how each individual trades happen.



Transaction List

ID	Symbol	Type	Date Time	Price	Size	Comm	Comment
0	ESRX	Sell	2004/06/24 10:03:00	76.58	100.00	0.00	Crosbelow short
1	ESRX	Buy	2004/06/24 10:04:00	76.58	100.00	0.00	Covers previous position
2	ALTR	Sell	2004/06/24 10:04:00	21.18	100.00	0.00	Crosbelow short
3	ALTR	Buy	2004/06/24 10:05:00	21.15	100.00	0.00	Covers previous position
4	ATYT	Sell	2004/06/24 10:05:00	18.10	100.00	0.00	Crosbelow short
5	ATYT	Buy	2004/06/24 10:06:00	18.11	100.00	0.00	Covers previous position
6	BRCM	Sell	2004/06/24 10:06:00	45.34	100.00	0.00	Crosbelow short
7	BRCM	Buy	2004/06/24 10:07:00	45.38	100.00	0.00	Covers previous position
8	AMGN	Sell	2004/06/24 10:07:00	54.38	100.00	0.00	Crosbelow short
9	AMGN	Buy	2004/06/24 10:08:00	54.32	100.00	0.00	Covers previous position
10	AMAT	Sell	2004/06/24 10:08:00	19.04	100.00	0.00	Crosbelow short
11	AMAT	Buy	2004/06/24 10:09:00	19.05	100.00	0.00	Covers previous position
12	COST	Sell	2004/06/24 10:09:00	42.86	100.00	0.00	Crosbelow short
13	COST	Buy	2004/06/24 10:10:00	42.83	100.00	0.00	Covers previous position
14	AAPL	Sell	2004/06/24 10:10:00	33.56	100.00	0.00	Crosbelow short
15	AAPL	Buy	2004/06/24 10:11:00	33.54	100.00	0.00	Covers previous position
16	CTXS	Sell	2004/06/24 10:11:00	19.69	100.00	0.00	Crosbelow short
17	CTXS	Buy	2004/06/24 10:12:00	19.71	100.00	0.00	Covers previous position
18	DISH	Buy	2004/06/24 10:12:00	30.00	100.00	0.00	Crossabove long
19	DISH	Sell	2004/06/24 10:13:00	30.00	100.00	0.00	Covers previous position
20	ADBE	Buy	2004/06/24 10:13:00	44.35	100.00	0.00	Crossabove long

Tutorial: Creating a Custom Drawing Tool

Custom drawing tool in NeoTicker® allows you to create any drawing tools you want. A custom drawing tool script tells NeoTicker® how the drawing tool behaves.

In this tutorial we will use a simple drawing tool script call "Time Cycle", to give a detail explanation and demonstrate how to create this drawing tool.

Creating Script

To create a custom drawing tool, the best way to start is to use script editor to open an existing drawing tool script (e.g. `timecycle.vbs)`:

1. Open a new script script editor.
2. In Script Editor, choose **File>Open Custom Drawing Tool**
3. Choose a script (e.g. `timecycle.vbs).`
4. Make modifications to the script.
5. Verify the script by choosing **Drawing Tool>Verify.**
6. Choose **File>Save As** and choose a new name. Make sure you are saving to the `CustomDrawingTool` folder.

The custom drawing tools are stored under the `CustomDrawingTool` folder in the NeoTicker® installation, e.g.

```
C:\Program
Files\TickQuest\NeoTicker3\CustomDrawingTool\template.vbs
```

Drawing Tool Script Explained

We will use `timecycle.vbs` as an example.

After you open the template.vbs you should see seven functions. They are:

```
toolname
numpoints
usevmult
usehmult
usefmult
clicked
draw
```

These functions define the behavior of the custom drawing tool.

toolname

This function assigns the name of your drawing tool, the name is a string and this name will show up as a tool-tip in the tool bar. Since the drawing tool we are working on is call time cycle therefore we will change the code to assign the string `"Time Cycle"` to the result of the `toolname` function

```
function toolname ()
    Toolname = "Time Cycle"
end function
```

numpoints

This function sets the number of control points you will need for your drawing tool. Since time cycle is essentially an infinite number of lines parallel to the price axis, we have to specify the interval between the two initial parallel lines, so the `numpoints` should be set to 2, one to tell NeoTicker® where the starting point of the drawing is, the other to tell where the next line is.

```
function numpoints ()
    numpoints = 2
end function
```

usevmult

This function sets the availability of the vertical multiples user interface (the **Vert** tab in the drawing tool's edit dialog). You can specify drawing of lines parallel to the time axis in between the one you have drawn. We will set this to false, since time cycle are only lines parallel to the price axis.

```
Function usevmult ()
    usevmult = false
end function
```

usehmult

This function sets the availability of the horizontal multiple user interface (the **Horz** tab in the drawing tool's edit dialog). You can specify drawing of lines parallel to the price axis in between the one you have drawn. We will set this to false because we don’t want more lines in between the time cycle.

```
Function usehmult ()
    usehmult = false
end function
```

usefmult

This function sets the availability of the fan multiple user interface (the **Fan** tab in the drawing tool's edit dialog). You can specify drawing of multiple diagonal lines. We are setting this to false.

```
Function usefmult()
    usefmult = false
end function
```

clicked

This function sets the rules to capture movement of the mouse pointer. This function passes in to you the boundaries of the pane, the control points as arrays and the list of multiples if available. This function returns an array of numbers as instructions to NeoTicker® on how to recognize the lines. For the time cycle we will allow the user to click on any line from top to bottom to select this drawing tool.

```
Function clicked (bounds, xlist, ylist, prices, hlist, vlist,
flist)
    dim ResultArray()
    dim delta, arraysize
    dim i, x1, x2, y1, y2, x3

    x1 = xlist  (0)
    x2 = xlist  (1)

    ' Height of the line fill the whole pane therefore
    ' Bounds are used as y coordinates
    y1 = bounds (1)
    y2 = bounds (3)
    i = 0 'number of coordinate values
    arraysize = 4 'size of the array
    redim preserve ResultArray (arraysize) 'declare a array size
of 4
    delta = x2-x1 ' Distance from first line draw to the second
line
    x3 = x1 ' Value for looping
    if delta = 0 then
        ' There is only one line so send one line
        call enable_line ( x3, y1, x3, y2, ResultArray, i)
    elseif delta > 0 then
        ' If lines are drawn to the right side. Delta is positive
        ' and enable all lines to the left side that are within
        ' the boundry of the current pane
        while x3 < bounds(2)
            ' Enable the lines within the pane only
            if x3 > bounds(0) then
                call enable_line (x3, y1, x3, y2, ResultArray, i)
            end if

            x3 = x3 + delta ' go to next line

            arraysize = arraysize + 4
            redim preserve ResultArray (arraysize)
        wend
    else
        ' Delta is negative lines are drawn to left.
        while x3 > bounds(0)
           if x3 < bounds(2) then
               call enable_line (x3, y1, x3, y2, ResultArray, i)
           end if
           x3 = x3 + delta
           arraysize = arraysize + 4
           redim preserve ResultArray (arraysize)
        wend
    end if

    redim preserve ResultArray(i)
    clicked = ResultArray ' return result
end function
```

In our code we have added another subroutine to support the line assignment process, because we have to repetitively assign the coordinate of lines, therefore we have created a subroutine called enable_line to repeatedly assign the coordinate to an array.

```
   sub enable_line (x1, y1, x2, y2, ByRef RArray, ByRef i)
      RArray (i)     = x1
      RArray (i + 1) = y1
      RArray (i + 2) = x2
      RArray (i + 3) = y2

      i = i + 4
   end sub
```

draw

This function returns drawing instructions for the drawing tool. It passes in to you the boundaries of the pane, the coordinate of the control points and the list of multiple lines. This function returns an array of numbers as instructions to how to draw the drawing tool. For Time Cycle we need only the x coordinate of the control points to get the interval. The result tells NeoTicker® to draw a series of lines with height equal to that of the pane and parallel to price axis.

```
' Draw parallel lines with height equal to from x1 to end of pane
function draw (bounds, xlist, ylist, prices, hlist, vlist, flist)
   dim i, j, x1, x2, y1, y2, x3
   dim delta, arraysize
   dim result()

   x1 = xlist (0)
   x2 = xlist (1)
   y1 = ylist (0)
   y2 = ylist (1)

   arraysize = 5
   redim preserve result (arraysize)
   i = 0
   delta = x2-x1
   x3 = x1
   if delta = 0 then

      ' Draw one single line
      call add_next_line (6, x3, y1, x3, y2, result, i)
   elseif delta > 0 then
      ' Draw lines until the right side boundry
      while x3 < bounds(2)
         ' don't draw if lines are not within display boundary
         if x3 > bounds(0) then
            call add_next_line (6, x3, y1, x3, y2, result,i)
         end if

         x3 = x3 + delta

         arraysize = arraysize + 5
         redim preserve result (arraysize)
      wend
   else
      ' Draw lines until the left side boundry
      while x3 > bounds(0)
          ' Don't draw if lines are not within display boundary
          if x3 < bounds(2) then
             call add_next_line (6, x3, y1, x3, y2, result, i)
          end if

          x3 = x3 + delta

          arraysize = arraysize + 5
          redim preserve result (arraysize)
       wend
    end if

    redim preserve result(i)
    draw = result
end function
```

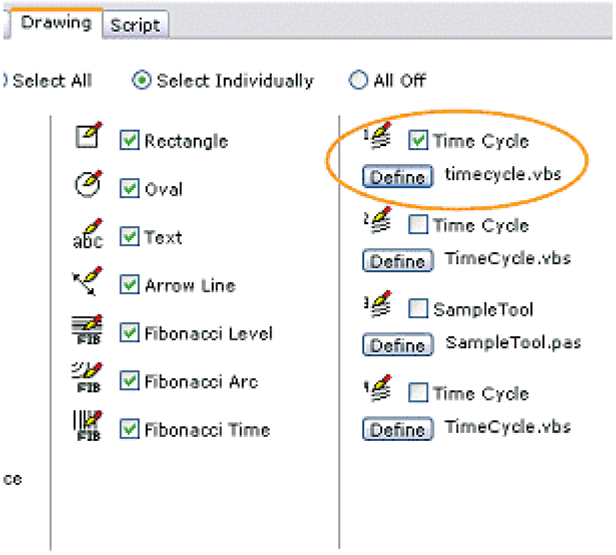
There is also a subroutine as a shortcut in assigning coordinates into the result array.

```
   sub add_next_line (c, x1, y1, x2, y2, ByRef list, ByRef i)

       list (i) = c

       list (i + 1) = x1
       list (i + 2) = y1
       list (i + 3) = x2
       list (i + 4) = y2
       i = i + 5
   end sub
```

Installing Script

There are two ways you can install a drawing tool. You can use the **Drawing Tool** menu in script editor, select **Drawing Tool>Install** as Custom1 to install the script.

The other way is the use the tool bar setup window, to open this window right click on the tool bar select **Setup** from the pop up menu. Press the **define** button to locate the script.



Drawing | Script

) Select All ● Select Individually ○ All Off

☑ Rectangle
☑ Oval
☑ Text
☑ Arrow Line
☑ Fibonacci Level
☑ Fibonacci Arc
☑ Fibonacci Time

ce

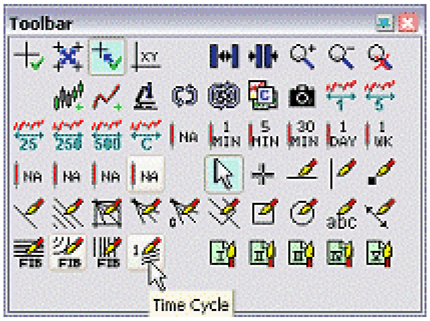
☑ Time Cycle
Define timecycle.vbs

□ Time Cycle
Define TimeCycle.vbs

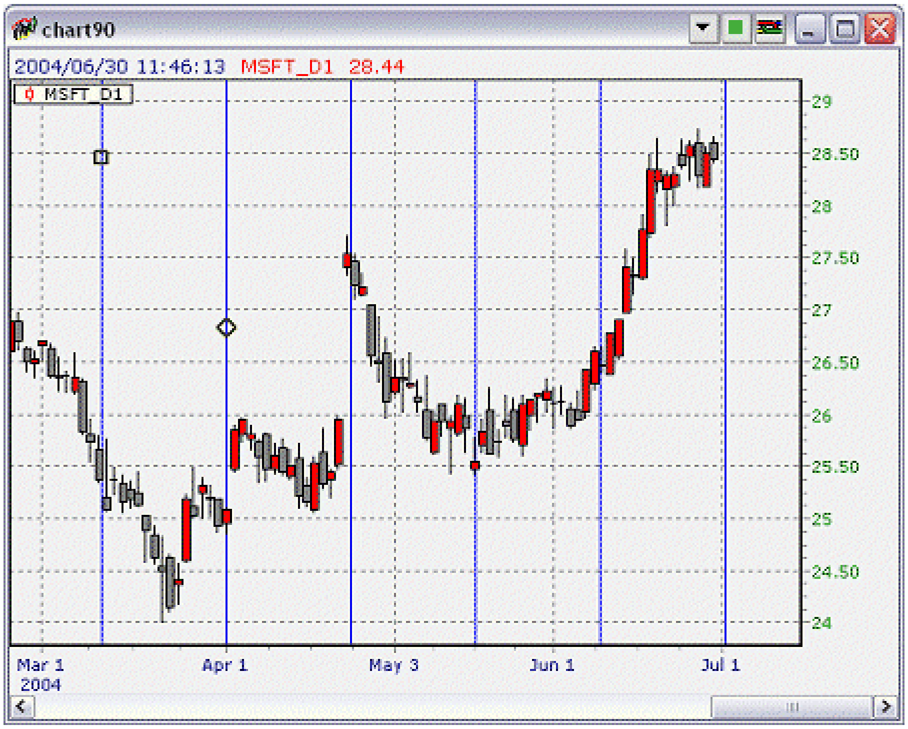
□ SampleTool
Define SampleTool.pas

□ Time Cycle
Define TimeCycle.vbs

After you've completed the installation you can check the custom drawing tool by moving you mouse pointer over the tool bar button. You should see the word Time Cycle show up when you pointer move over the button.



You can try out the time cycle drawing tool you've created by drawing it on a chart. Click on the first custom drawing tool button and you can drag the two control points to adjust the interval between the lines, when you release the mouse button you will get a series of parallel lines with equal interval in between them.



Indicator Programming Topics

This section covers the essentials of indicator programming. You should this section to get an overall impression of how NeoTicker® indicators work.

Interoperability

Once a indicator or trading system is installed, it is treated no different from internal indicators and trading system. You can use it in charts, pattern scanner, quote window and other scripts and expect it to behave the same way as if it is internally defined.

For example, you can program an indicator in VBScript, then reuse the calculation within your trading system written in Delphi.

Modular Design

Each indicator can be easily replaced. If you do not like the behavior of an indicator, you can write your own version and replace the build-in one. You can replace both internal and external indicators.

All you need to do is install an indicator of the same name to NeoTicker®

At the Point Evaluation

NeoTicker® uses at the point evaluation for indicators and trading systems. This means from the indicator or trading system's point of view, it always focuses on the current value to be evaluated.

For example, when you construct an indicator to calculate a simple moving average, which is the average of the current N values, your script specifies to sum up the current N data points and divide that by N. You do not need to know where you are within the data series. All you need to do is to specify the abstract behavior of your calculation relative to the current bar position.

One advantage of at the point evaluation it forces the code to be consistent for different bars. Your code will focus on the trading related information in the data, rather than how the data is stored within NeoTicker®.

Another advantage is it safeguard against accidental data look ahead. With at the point evaluation, historical data is available, but it is programmatically impossible to access data in the future. When the indicator or trading system is evaluating against historical data, you have the assurance that the indicator or trading system does not access data in the future due to programming mistakes, and makes physically impossible predictions and trades. This assurance is important for indicators and trading systems that are expected for real-life deployment when real money is at stake.

Object Driven

Objects are interfaces to NeoTicker®'s data and functionality. No matter what the environment you work in (e.g. VBScript, DelphiScript, IDL, etc), you can access NeoTicker®'s data and functionality through the objects.

Objects provides a uniform interface across the different environments. Aside from minor syntactic differences, the same feature is accessed the same way in different environment. For example, the way to access data is the same in Delphi Script and VBScript. You can easily understand and translate indicators and trading systems that are written in a language different from your preferred language.

Objects are logical divisions between features. From time to time, new objects are introduced for new features. Because the new objects are separate entities from existing objects. Existing code will work without complex rewrite.

Concept of Invalid Data Values

NeoTicker® supports the scientific concept of invalid data values.

That means if there is a holiday within a data series, and you intend to have that presents, such data point will then exists and its value will equal to invalid. In a microscopic scale, if a symbol is not traded in a bar, an invalid bar is also present.

You have the choice on how to process an invalid data value. Invalid data value presents the opportunity to analyze at a time when trading is not present. For example, it is very easy to measure the trading frequency in time with invalid bar information.

Separation of Data and Drawing

You are not directly controlling NeoTicker® to draw onto a chart with an indicator.

An indicator is series of values, similar to a data series. An indicator script or IDL contains the instructions to calculate the values for the indicator series. The indicator series is then used in charts, pattern scanner, quote window, etc.

Visual properties such as color and plot style are separated from the indicator values. The visual properties control how the indicator values are drawn on a chart, and they can be changed without triggering a recalculation of the indicator. In the case of pattern scanner and quote window, the visual properties are ignored as these windows do not draw indicators.

Links and Updates

Basic indicator programming involves access to an underlying data series. All indicators must link to at least one data series. This primary link determines the time frame and bar size of the indicator.

For historical data, the indicator will calculate once per bar. For real-time data, the indicator has two choices, whether to update by tick or not.

When update by tick, the indicator is recalculated for every tick that comes to the data. If the indicator is linked to multiple data, any tick that comes to any of the linked data will trigger a recalculation. Within the same bar, the calculation environment will remain constant for different ticks, until the bar is finished. For example, if your indicator performs some kind of bar counting operation, you will not over count because of update by tick.

If update by tick is off, the indicator is calculated once after all ticks has arrive for the bar. Turning off update by tick can save CPU usage for complex indicators that demand a lot of CPU power.

Calculation consistency is ensured for historical data, update by tick and update not by tick. In all three cases, the indicator value will be exactly identical for the same data.

Evaluation Order

Indicators are evaluated in chronological order. They are evaluated from the earliest data point to the latest.

Multiple Plot Per Indicator

Each indicator can draw an arbitrary number of plots. Each plot is actually an series by itself. For example, the build-in Bollinger Bands 5 indicator has 5 plots.

At each evaluation point, you need to calculate and assign the values for all the plots defined in an indicator.

For multiple plot indicators, you should put the most important information in the first plot.

Trading Systems

Trading system is a specialized type of indicator. The `Trade` object provides a realistic environment for trade execution. You can freely access this object to generate orders and use its internal methods to calculate various statistics on the performance of the system against any data you specified.

When no orders is generated through the trade object, the host indicator behaves just like a regular indicator. When orders are placed and trades are made through the trade object, the indicator will automatically be recognized as a trading system and can be tracked by the system monitor window.

There is a flag for turn on trading system UI in design time. Having the flag turn on will enable trading system UI (such as the menu item for launching performance viewer) to be available in run time. This is provided as a convenience. Trading system is fundamentally separated from trading system UI. You can have a perfectly functional trading system UI without any UI.

Because trading systems are specialized indicator, you can use it just like an indicator. Thus a time chart window can host multiple trading system and generate multiple real-time system signals.

Having the trading system object hosted by an indicator has the advantage of the ability to plot various system information onto a chart when you are developing the system. You are not bounded by the reported information from NeoTicker® to analysis the system and you can fine tune your model based on information you deem important.

About VBScript

VBScript is in many ways similar to Visual Basic and other programming languages. Here are some fine points you may want to be aware of:

- Unlike other languages VBScript does not take shortcut when evaluating an if statement. All conditions chained by boolean operators `AND` and `OR` are evaluated.
- When calling a method as a function, i.e. with assignment, you must have brackets around the parameters, e.g.

  ```
    a = ItSelf.MakeIndicator("sama", "ama", Array("1"),
  Array(param1.str))
  ```

  The same method, when calling as a procedure, i.e. without assignment, you must not have brackets around the parameters, e.g.

  ```
    ItSelf.MakeIndicator "sama", "ama", Array("1"),
  Array(param1.str)
  ```

Accessing Linked Data

To access the first two linked data series, use the `Data1` and `Data2` objects. For example,

- `Data1.Value(0)` will access the current bar value of the first linked series (VB syntax).
- `Data2.Value(1)` will access the value of one bar ago of the second linked series (VB syntax).

To access arbitrary link, use the `LinkSeries` object. For example,

- `LinkSeries(1).Value(0)` will access the current bar value of the first linked series (VB syntax).
- `LinkSeries(2).Value(1)` will access the value of one bar ago of the second linked series (VB syntax).

`LinkSeries` supports the `Count` property. Valid index for linked series are from `1` to `LinkSeries.Count`.

Accessing Non-Linked Data Series in the Chart

Normally an indicator receives input data from its data links. This is sufficient for many types of calculations.

In addition to data link, an indicator can access data through the `DataSeries` object. When data is accessed through the `DataSeries` object, the data does not have to be explicitly linked to the indicator.

Only data series can be accessed through the `DataSeries` object. Indicators can not be accessed this way.

Properties and Methods

To access the number of data series available, use the property Count. For example,

```
    total_issue = DataSeries.Count
```

returns the number of data series available and assigns the value to the variable total_issue.

To access a specific data series, you can simply indexed into the DataSeries object. The index ranges from 1 to DataSeries.Count. The result is identical to what you expect from Data1 or Data2. Methods such as Value and Valid are all available. For example,

```
   DataSeries(i).Value(0)
```

returns the value of the current bar of the data series i.

Note to Delphi Script Users: In Delphi Script, use `DataSeries.Items[i]` to access a data series.

Time Frame Issues

All indicators have at least one data link. The first data link, `Data1`, serves as the reference time frame for `DataSeries`. Bars from different time frames can be accessed. The values will be the currently available value.

Real-time Triggering

`DataSeries` does not trigger indicator calculation in real-time. The indicator is still relying on the links for triggering. To ensure the indicator is recalculated, you need to make sure indicators' link are assigned to data that receives enough real-time updates.

Time Chart vs. Pattern Scanner

In the current version, `DataSeries` is mainly for use in time charts. When the indicator is residing in a time chart, all the data series of the time chart are available to the `DataSeries` object.

If the indicator is residing in a pattern scanner, it can only access the available data series during the scan iteration, i.e. `Data1` if you are scanning one symbol per iteration, `Data1` and `Data2` if you are scanning two symbols per iteration.

MakeIndicator with DataSeries

You can pass `DataSeries` as input to `MakeIndicator`. The syntax is: `#N` and `#N.F` where `N` is the data series number and `F` is the field. The field is identical to those of a regular link series. If a field is not specified, it is defaulted to the close value of the data series.

For example,

```
itself.makeindicator "pdc", "prevDClose", Array("#3.C"),
Array()
```

creates a previous day close indicator using the closing values of the third data series.

Example: An Advance Decline Indicator

The following is a VBScript version of an advance decline indicator. When applied to a group of symbols in a chart, it calculates the advance and decline percentage of the total issues. To use this indicator, you simply load a group of symbols into a chart, and apply the indicator on the first data series. The data series must be of the same time frame.

Tips: if you want to use this indicator for a large group of symbols, turn the visibility of the data series to off in the chart manager. This will significantly reduce the CPU load.

If you paste the script below directly to a script editor, please enable the script header by choosing `Visual>Show Script Header` before pasting.

```
# NEOTICKER DATA BEGIN
ScriptType=Indicator
Description=Advance Decline
Name=advdec
Language=VBScript
Links=1
MinBars=0
Multiplot_num_plots=2
Multiplot_color_0=65280
Multiplot_style_0=Line
Multiplot_width_0=2
Multiplot_enabled_0=1
Multiplot_color_1=255
Multiplot_style_1=Line
Multiplot_width_1=2
Multiplot_enabled_1=1
UpdateByTick=0
```

```
Param_count=0
Explanation_Lines=4
Explanation0=Advance Decline plots the advance and decline
ratios.  The first
Explanation1=plot is the number of advancing issues divided by the
total number
Explanation2=of issues. The second plot is the number of declining
issues divided
Explanation3=by the total number of issues.
# NEOTICKER DATA END
function advdec
    total_issue = DataSeries.count
    ' If the number of issue is 0 or if
    ' the reference series (Data1) is not
    ' valid, don't calculate advance decline
    if (total_issue = 0) or (not Data1.Valid(0)) then
       ItSelf.SuccessEx(1) = false
       ItSelf.SuccessEx(1) = false
       exit function
    end if
    adv_total = 0
    dec_total = 0
    for i = 1 to total_issue
        ' Convert i to string type so the indicator name and index
can
        ' be constructed
        i_str = tq_integer2str(i)
        ' Create the previous day close indicator
        indname = "prevc" + i_str
        indindex = "#" + i_str + ".C"
        itself.makeindicator indname, "prevDClose",
Array(indindex), Array()
        ' Get the previous close and current value
        prevclose = itself.indicator(indname).Value(0)
        currvalue = DataSeries(i).Value(0)
        if currvalue > prevclose then
            adv_total = adv_total + 1
        elseif currvalue < prevclose then
            DEC_total = DEC_total + 1
        end if
    next
    ' Report advance decline in percentage
    ItSelf.Plot(1) = adv_total / total_issue * 100
    ItSelf.Plot(2) = DEC_total / total_issue * 100
end function
```

Accessing User Defined Parameters

To access user defined parameters (e.g. period parameter in a moving average indicator), use the `ParamN` or `Params` properties.

See ***Param and Params Object*** (on page 1519) for more information.

Creating Help for Indicator

If your indicator is used by users other than yourself, it is a good idea to include a help file for the indicator. NeoTicker® provides the facility to let your users quickly find your help file.

Help file is setup in design time:

1. In **Script Editor**, choose **Indicator>Setup**.
2. In the dialog, press the **Misc** tab.
3. In the **Help File** field, enter a local file path or a webpage.
4. Press the **Apply** button.

In run-time, the user can simply click on the **More Info** button when selecting the indicator to open your help file.

For security reason, only the following type of files are opened: .chm, .txt, .htm, .html.

Local Help File

If you specify a relative path, NeoTicker® will try to find the help file under the `Help` folder in the NeoTicker® installation, e.g. `C:\Program Files\TickQuest\NeoTicker4\Help`.

Webpage

The help file can reside on a webpage on your website. For example, if the help file is in `www.mycompany.com/Help/myindicator.html`, simply enter the URL with the `http://` prefix in front in the **Help File** field, i.e.

http://`www.mycompany.com/Help/myindicator.html`

Creating Higher Time Frame Series

The ability to monitor higher time frame information has always been a dream for technical analysis. With NeoTicker®, it is now a reality. You can freely monitor higher time frame information within an indicator or trading system and generate consistent historical and real time decision making information with confidence.

Creating a Compressed Series

The `ItSelf` object methods `CompressSeries` and `CompressSeriesEx` can take any series from within an indicator and compress that to a higher bar size and/or time frame based on your criteria. The resulted series will be updated real time with the indicator and you can freely access its data with the ItSelf object property Series.

The syntax for `CompressSeries` is as follows:

```
ItSelf.CompressSeries (SeriesName, Source, TimeFrame, BarSize)
```

The syntax for CompressSeriesEx is as follows:

```
ItSelf.CompressSeriesEx (SeriesName, UniqueID, Source,
TimeFrame, BarSize)
```

`CompressSeries` and `CompressSeriesEx` creates a new embedded data series with the name defined by `SeriesName`. With `CompressSeriesEx`, the new data series can be updated by UniqueID an integer parameter specified by the programmer. You only need to use `CompressSeriesEx` in IDL indicators for performance enhancement (see Performance Enhancement section below).

This new series is based on the `Source` (a string value) that can be one of the following:

- An Integer (e.g. '1', '2', '3', ...) that represents a linked series (`Data1`, `Data2`, `LinkSeries` instances).
- An integer with prefix "#" (e.g. '#1', '#2', '#3', ...) represents unlinked series (`DataSeries` instances).
- A name (e.g. 'myavg', 'sma20', ...) represents an embedded indicator series you have previously defined within the same indicator instance.
- A name with prefix "$" (e.g. '$myM5', '$myDaily', ...) represents another embedded compress series you have previously defined within the same indicator instance.

To define `TimeFrame`, you can use one of the following pre-defined constants.

```
ppTick
ppSec
ppMin
ppHour
ppDaily
ppWeekly
ppMonthly
ppQuarterly
```

```
ppYearly
```

Remember that you can only compress to the same or a higher time frame of the source series. If the time frame condition does not match, it will not generate the series you want.

For any `TimeFrame`, you will always need to accompany that with the specification of `BarSize`, which is an integer.

To find out about the time frame of the source series, you can use their respective properties `TimeFrameType` and `TimeFramePeriod` to determine how you want to compress the data.

A few special restrictions apply to compress series:

- You cannot compress a multiple tick series (e.g. 20-tick) to a higher time frame, but you can compress it to a multiple of the same `ppTick` time frame. For example, you can compress 20-tick to 40-tick or 60-tick but not to 5-min. The reason is that the 20-tick could cross the boundary of time that is needed to separate the data correctly.
- If the source series is of `ppTick` timeframe and `BarSize` is 1, then you can compress that to a higher time frame with no restriction.
- You cannot compress a source series to the same time frame with higher `BarSize` if the higher `BarSize` is not divisible by the source series. For example, you can compress 5-min bar to 15-min bar or 45-min bar, but not to 23-min bar. To solve this problem, you can load data with `BarSize` of 1 (say 1-min bar), then generate all the different higher `BarSize` compressed series from there.

Some examples of using the `CompressSeries`. The call:

```
ItSelf.CompressSeries ('a', '1', ppMin, 5)
```

compresses the first Link series of the indicator to 5-min bar.

The call:

```
ItSelf.CompressSeries ('a', '1', LinkSeries.items
[1].TimeFrameType, LinkSeries.items[1].TimeFramePeriod * 3)
```

compresses the primary link series of the indicator to 3 times its original bar size.

Accessing the Compressed Series

To access the data of the compressed series, you can use the `ItSelf.Series` property.

The syntax for Series is as follows.

```
ItSelf.Series (SeriesName)
```

`Series` returns the compressed series previously defined using the name `SeriesName`. The returned object is a data object that have all the properties and methods exactly like the `Data1` or `Data2` object.

Using a Compressed Series to Make Indicator

To apply indicator calculated on the compressed series, you can use the `MakeIndicator` method like the following example.

In DelphiScript,

```
ItSelf.CompressSeries ('a', '1', ppMin, 5);
ItSelf.MakeIndicator ('myavg', 'average', ['$a'], ['10']);
```

In VBScript,

```
ItSelf.CompressSeries "a", "1", ppMin, 5
ItSelf.MakeIndicator "myavg", "average", Array ("$a"), Array
("10")
```

The above example will create a simple moving average of the compressed series 'a'. You can then use this indicator whatever way you want just like the other embedded indicators.

Performance Enhancement

In IDL indicators, you can use `CompressSeriesEx` to enhance performance. The idea is to reduce unnecessary data processing between the IDL and NeoTicker®. It is similar to using unique id's in `MakeIndicatorEx` and `UpdateIndicatorEx` (see ***Using Other Indicators*** (on page 1479)). The following code illustrates the usage:

```
comp_series := Itself.UpdateCompressSeriesEx (35);
if comp_series = nil then
    comp_series := Itself.CompressSeriesEx ('a', 35, '1',
ppMin, 5);
// Use comp_series as a replacement for ItSelf.Series('a')
```

Creating Managed Series

If you want to create higher time frame series, compress series is easier to use. See ***Creating Higher Time Frame Series*** (on page 1452).

To be written.

Color Constants

Methods that expect a color input, e.g. drawing color for indicator created drawing tools, use integer value for the color. The following are pre-defined color constants you can use:

```
clAqua
clBlack
clBlue
clDkGray
clFuchsia
clGray
clGreen
clLime
clLtGray
clMaroon
clNavy
clOlive
clPurple
clRed
clSilver
clTeal
clWhite
clYellow
```

For custom color, the format is the decimal representation of the hex number BBGGRR, where BB is the blue color, GG is the green color, RR is the red color.

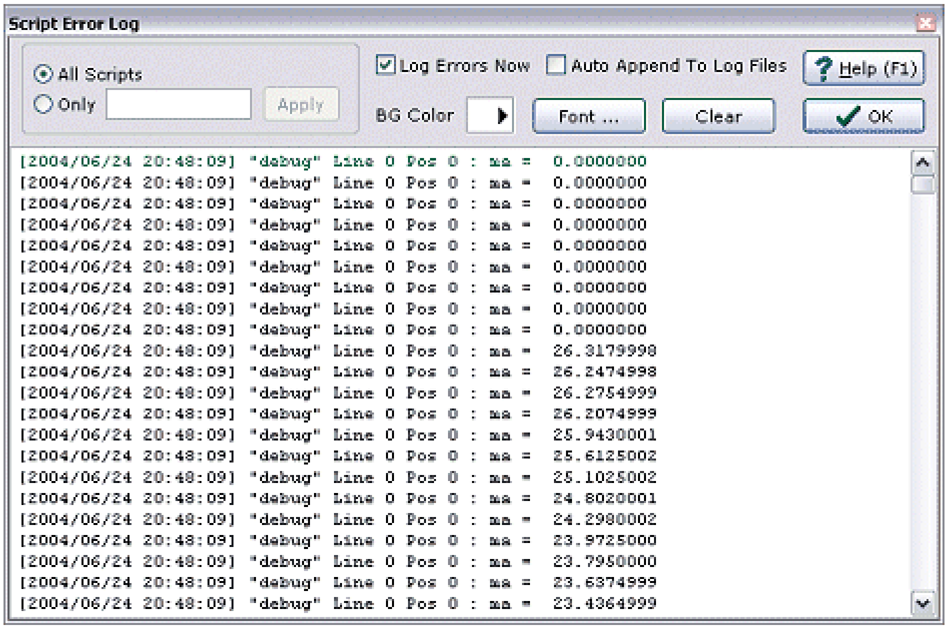
Debugging Scripts/IDL

Script Error Log Window can track the errors generated by the indicator scripts during their execution / calculation. It is a very useful tool if you are developing a script and is debugging your indicator creation.

To open script error log, choose **Program>Script Error Log** from the main window. To start tracking error,

1. Specify the indicator you want to track error.
2. Enable **Log Errors Now**.

3 When an indicator is run, the run time error is logged in the script error log.



● All Scripts

☑ Log Errors Now

□ Auto Append To Log Files

```
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  0.0000000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  26.3179998
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  26.2474998
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  26.2754999
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  26.2074999
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  25.9430001
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  25.6125002
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  25.1025002
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  24.8020001
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  24.2980002
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  23.9725000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  23.7950000
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  23.6374999
[2004/06/24 20:48:09] "debug" Line 0 Pos 0 : ma =  23.4364999
```

Options

Log Errors Now

When checked, the Script Error Log Window will start to record the runtime errors generated by running indicators. It has a limit of 1000 lines. If more than 1000 errors are recorded, the oldest ones will be discarded to make room for the new ones.

Auto Append to Log Files

When checked, the errors recorded by the Script Error Log Window will be saved to the external log files at the same time. The log files are located within the NeoTicker® directory under the sub-directory `ScriptErrorLog`. For each date, the errors are saved into multiple log files each named in the format `yyyymmdd_XX.txt`.where the first part of the file name is the date of the error log.

All Scripts or Specific Script

You can choose to monitor all scripts or just a specific script. If you choose to monitor a particular script, you must type its indicator name and click the **Apply** button.

Font Button

You can modify the font used in the Script Error Log Window.

BG Color Button

You can modify the background color used in the Script Error Log Window.

Clear Button

To clear all the errors recorded in the window, click this button.

Shortcuts

Double Click - the script that caused the error will be opened in a script editor with the cursor moved to the appropriate line. If the script is already opened, it will be activated instead.

Visual Basic IDL

For indicator written in Visual Basic and connected through IDL, run-time OLE errors are shown in script error log.

Evaluating Formula Parameters

In scripts/IDL, you can specify a parameter to be formula. For this parameter, user can enter a formula. This formula is given to your script as a string, and you need to implement a way in the script to evaluate the formula.

The idea is to use the fml indicator to help you evaluate the formula. Since script/IDL supports indicator-on-indicator, this is relatively easy to do with the `MakeIndicator` method.

The only thing you need to decide is how many links the formula needs to support. For example, if your indicator is a 2-link indicator, you probably want your user to be able to enter a formula parameter that can access to both linked data. In this case, you will need to use the fml2 indicator, which supports 2 data links. Similarity, if you want the formula parameter to support 3 links, use fml3. If you want the formula parameter to support 1 link, use fml.

Example: Long Signal Entry for 2 Links (Delphi Script)

Here is an example taken from Backtest EZ. The parameter in this case is the long entry signal (`param1`). The user can type in any formula parameter that compares the 2 data links of Backtest EZ and when the formula is evaluated to true, a long signal is fired.

Because the formula needs to access 2 link, we will use fml2 to evaluate the formula. The first thing is to define the fml2 indicator inside the script:

```
   if param1.str <> '' then
   begin
      longentry := itself.makeindicator ('le', 'fml2', ['1', '2'],
[param1.str]);
      has_longentry := true;
   end;
```

Here, we create an fml2 indicator that uses data link 1 and data link 2 of Backtest EZ.The formula parameter, `param1`, is accessed as a string, and is passed to fml2.

We also do 2 things here. If the formula parameter is empty, we consider there is no long signal. Thus we set a flag, `has_longentry,` to false. Also, the indicator series that has been created is assigned to the variable `longentry`. This variable is declared as a variant and is used to access the signal later:

```
   if has_longentry and
      longentry.valid [0] and
      (longentry.value [0] <> 0) then
   begin
       // Make a long order
   end;
```

Here, the condition in the if statement determines whether a long order is fired. The `longentry` variable created earlier is accessed like any other indicator series, and contains the evaluated result of the formula. So if the value is valid and is non-0, a long order is fired.

Handling Valid and Invalid Values

NeoTicker® makes the distinction between valid and invalid values. That is, a bar can be invalid in NeoTicker®. An invalid bar does not contain any value, yet it occupies a time slot. In daily charts, most common cause of invalid bar in data is holidays. In minute charts, it can be that a security is not traded at all in a minute.

By having valid and invalid values, NeoTicker® provides a framework to properly handle holes in data. The indicator writer has the flexibility to decide what to do when a data is not available. This is in contrast to classic trading software which simply ignores invalid time slots or fill the time slot in with some arbitrary default values, and provides no choice to the indicator writer.

Examples in this section is written using a VBScript/Visual Basic syntax. If you use other languages, you need to adjust the syntax accordingly.

Querying Validity of Input

You can query the validity of a data by the `Valid` and `ValidEx` properties:

```
Data1.Valid(0)
Data1.ValidEx(2, 0)
```

The first line returns the validity of the first data link's current bar. Without the plot information, the first line defaults to the first plot. The second line returns the validity of the first data link's second plot's current bar.

Similarly, you can query the validity of the indicator itself:

```
ItSelf.Valid(1)
ItSelf.ValidEx(2, 1)
```

The code above query the validity of the last bar of the indicator. Similarity the EX version queries the second plot.

Returning Validity for Indicator

You can return the validity of an indicator by setting the `Success` and `SuccessEx` properties:

```
ItSelf.Success = false
ItSelf.SuccessEx(2) = false
ItSelf.SuccessAll = false
```

The first line sets the indicator's validity to false. Without the plot information, the first line defaults to the first plot. The second line sets the indicator's second plot's validity to false. The third line set the validity of all plots to false.

MakeValidArray and MakeValidArrayEx

`MakeValidArray` and `MakeValidArrayEx` lets you construct an array from only the valid value in a data series. This comes in handy when you need a fixed size list of valid values. The array created by make array is 0-based.

The following is a VBScript that implements a simple moving average using `MakeValidArray`.

```
function movvalidvb()
   dim price
   period = Params(1).int
   Data1.MakeValidArray price, "C", period, false
   sumx = 0
   for i = 0 to period - 1
       sumx = sumx + price(i)
   next
   movvalidvb = sumx / period
end function
```

The array `price` is created to store only the valid data values from link 1 for moving average calculation.

`MakeValidArrayEx` is a generalized version of `MakeValidArray`. `MakeValidArray` lets you specify a plot after the array name. For example, you can replace the `MakeValidArray` call with the following and the calculation is identical:

```
   Data1.MakeValidArrayEx price, 1, "C", period, false
```

Important Differences Among Scripting Languages

- Delphi Script and VBScript are case insensitive while JavaScript is case sensitive. When you access the ItSelf, ParamX, DataX objects, you have to be careful with case in JavaScript.
- VBScript's methods are called without brackets. The following statements are equivalent:

  Delphi Script - `Report.AddLine (' ', 'Hello World');`
  VBScript - `Report.AddLine " ","Hello World"`
  JavaScript - `Report.AddLine (" ","Hello World");`

- Delphi Script uses square brackets to access data property and round brackets for function calls. VBScripts and JavaScript do not make this distinction. For example, the following statements are equivalent:

  Delphi Script - `sum := Data1.Value[0] + tq_min(Data2.Value[0],0);`
  VBScript - `sum = Data1.Value(0) + tq_min(Data2.Value(0),0)`
  JavaScript - `sum = Data1.value(0) + tq_min(Data2.Value(0),0);`

Following are examples of implementation of a simple moving average function in different scripting languages.

Delphi Script

```
function Average : double;
var i : integer;
begin
   result := 0;

   for i := 0 to param1.int - 1 do
   begin
      result := result + data1.close [i];
   end;

   result := result / param1.int;
end;
```

VBScript

```
function average()
   period = param1.int
   sumx = 0

   for i = 0 to period - 1
      sumx = sumx + data1.value(i)
   next

   average = sumx / period
end function
```

JavaScript

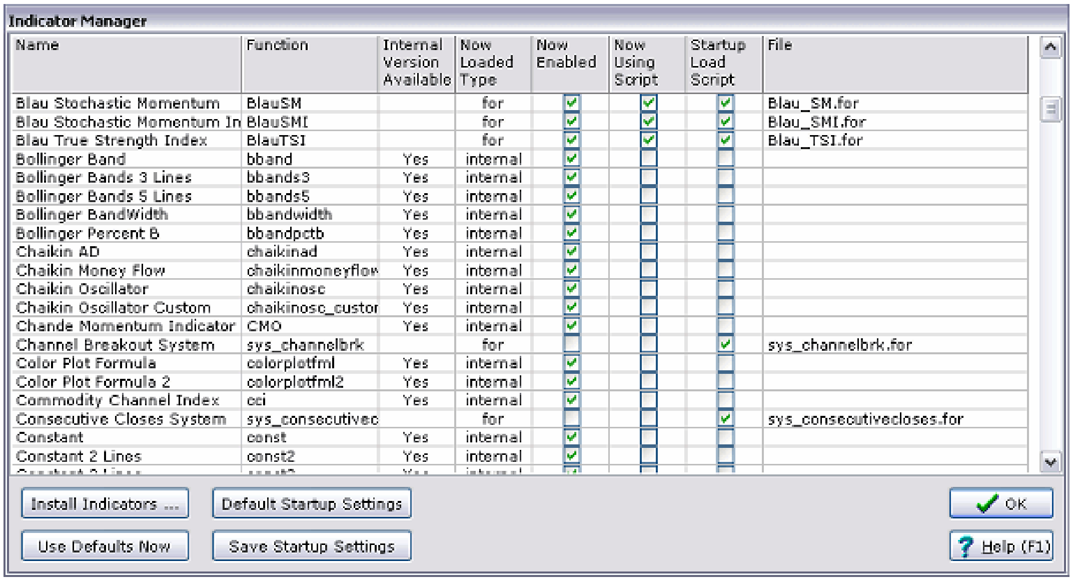
```
function average()
{
   var sumx = 0,
       period = 0;

   period = Param1.Int();
   sumx = 0;

   for (var i = 0; i < period; i++)
      sumx += Data1.Close(i);
```

```
        return sumx / period;
    }
```

Indicator Manager

Indicator Manager (formerly known as Script Manager) displays the status of indicators in NeoTicker®, and lets you control how NeoTicker® handles indicator loading at program start.

To open Indicator Manager, choose **Program>Indicator Manager**.



Name	Function	Internal Version Available	Now Loaded Type	Now Enabled	Now Using Script	Startup Load Script	File
Blau Stochastic Momentum	BlauSM		for	☑	☑	☑	Blau_SM.for
Blau Stochastic Momentum In	BlauSMI		for	☑	☑	☑	Blau_SMI.for
Blau True Strength Index	BlauTSI		for	☑	☑	☑	Blau_TSI.for
Bollinger Band	bband	Yes	internal	☑	□	□	
Bollinger Bands 3 Lines	bbands3	Yes	internal	☑	□	□	
Bollinger Bands 5 Lines	bbands5	Yes	internal	☑	□	□	
Bollinger BandWidth	bbandwidth	Yes	internal	☑	□	□	
Bollinger Percent B	bbandpctb	Yes	internal	☑	□	□	
Chaikin AD	chaikinad	Yes	internal	☑	□	□	
Chaikin Money Flow	chaikinmoneyflow	Yes	internal	☑	□	□	
Chaikin Oscillator	chaikinosc	Yes	internal	☑	□	□	
Chaikin Oscillator Custom	chaikinosc_custor	Yes	internal	☑	□	□	
Chande Momentum Indicator	CMO	Yes	internal	☑	□	□	
Channel Breakout System	sys_channelbrk		for	□	□	☑	sys_channelbrk.for
Color Plot Formula	colorplotfml	Yes	internal	☑	□	□	
Color Plot Formula 2	colorplotfml2	Yes	internal	☑	□	□	
Commodity Channel Index	cci	Yes	internal	☑	□	□	
Consecutive Closes System	sys_consecutivec		for	□	□	☑	sys_consecutivecloses.for
Constant	const	Yes	internal	☑	□	□	
Constant 2 Lines	const2	Yes	internal	☑	□	□	

Internal vs. External Indicators

There are two types of indicators: internal and external indicators. Internal indicators are part of the program, whereas external indicators are defined by scripts/formula/IDL. In general, you will want to use internal version unless you have made some modifications to the indicator yourself.

Many of the common technical indicators have both internal version and script-based version available. By default, the internal versions are used. Indicator Manager lets you to override this behavior for individual indicators.

In the figure above, you can see that the Bollinger Band indicator has an internal version and the internal version is currently loaded. You can switch to the script-based version by clicking on the check box under the **Now Using Script** column. This will make NeoTicker® uses the script-based version for this session of NeoTicker®. You can revert the indicator to use the internal version by clicking on the checkbox again.

If you always want to use the script-based version of the indicator, click on the the check box under the **Startup Load Script** column. Once checked, NeoTicker® will always load and use the script-based indicator.

For indicators that only have script version, for example, Backtest Basic, you can check off the check box under the **Startup Load Script** column such that the indicator will not be loaded on start up. By turning off the script-based indicator you do not use, you can speed up NeoTicker®'s startup and the indicator will not be available. You can later load the indicator by checking the check box under the **Now Enabled** column.

Once an indicator has been loaded, it cannot be unloaded within the same session.

Reference

Name

Long name of the indicator. This is the more descriptive name you see in Indicator Editor, e.g. Bollinger Bands 3 Lines. See also ***Indicator Quick Reference*** (on page 606).

Function

The function name or short name of the indicator. Function name is used when you reference an indicator in scripts and formulas. See also ***Indicator Quick Reference*** (on page 606).

Internal Version Available

Whether an internal version of the indicator is available. Internal version indicators are indicators that is an integral part of NeoTicker®.

Now Loaded Type

The type of the indicator that is currently loaded. It can be Internal, for (formula), pas (Delphi Script), vbs (VBScript), js (javascript) or idl.

Now Enabled

Whether the indicator is currently enabled.

Now Using Script

Whether the script is currently being in used.

Startup Load Script

Whether to load script on startup. Not loading script can improve start up speed, especially if you have been programming in NeoTicker® for a while and have lots of scripts that are not actively in use.

File

File name for non-internal indicator.

Install Indicator

Choose any script file on your computer, NeoTicker® will copy that file into the indicator directory and install it

Use Default Now

Revert the settings to default and use the default settings now

Default Startup Settings

Revert the start up settings to default

Save Startup Settings

Save the start up settings

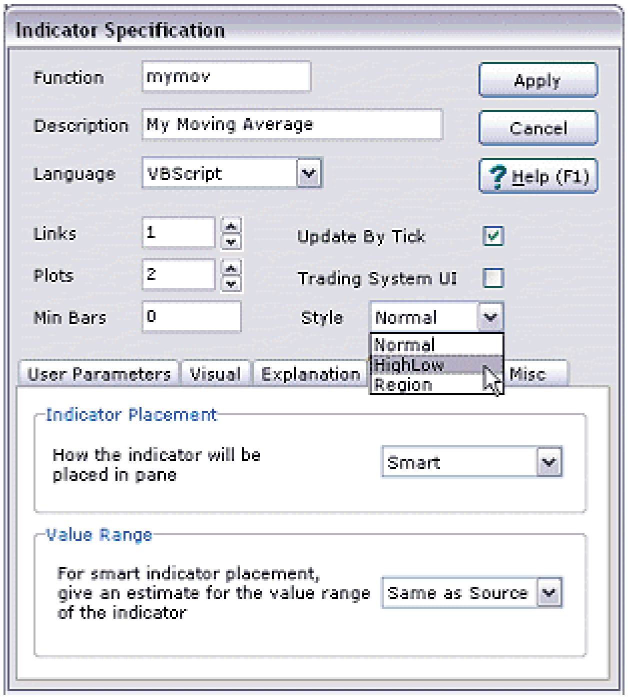
OK

Close Indicator Manager

Indicator Meta Plot Style

The Meta Plot Style has the ability to combine multiple plots into special drawing style.

It can be defined under **Style** in the Indicator Specification Window in the Script Editor.



Indicator Specification

Function: mymov

Description: My Moving Average

Language: VBScript

Links: 1

Plots: 2

Min Bars: 0

Update By Tick ☑

Trading System UI ☐

Style: Normal

Indicator Placement

How the indicator will be placed in pane: Smart

Value Range

For smart indicator placement, give an estimate for the value range of the indicator: Same as Source

There are many Meta Plot Styles available. They are made available when you change the number of plots in your indicator specification.

Normal

It is available for any number of plots.

It is the default style. All the plots are drawn individually and not connected together.

Region

It is available for indicators with 2 plots.

It connects plot 1 and plot 2 into a surface. The Region Indicator is using this style.

HighLow

It is available for indicators with 2 plots.

It connects plot 1 and plot 2 into a vertical line. Thus you can mark and highlight specific bars in the chart easily. The highlight formula indicator is using this style.

HighLowBar

It is available for indicators with 3 plots.

The first 2 plots are connected together into a vertical line like the HighLow style. Plot 3 is used as the coloring information. With this style, you can easily create highlight markers with different colors to indicate different market conditions.

OHLC

It is available for indicators with 4 plots.

The 4 plots are treated as Open, High, Low, and Close. And when you add indicators with this style, it will show up as a bar series.

Candle

It is available for indicators with 4 plots.

The 4 plots are treated as Open, High, Low, and Close. And when you add indicators with this style, it will show up as a candle series.

Special Usage of OHLC and Candle

When you define an indicator with OHLC or Candle style. These indicators can be used as if they are data series themselves. It is a very powerful way for constructing custom indexes and virtual data series based on user formulas.

The section Creating Virtual Data Using Formula Language has a complete example of using this special feature to get you started.

Indicator Updated by Timer

Normal indicators are updated whenever a change is happening to its links. You can create indicators that are updated by timer. When an indicator is updated by timer, its latest bar is calculated:

- Periodically at a certain time interval
- Whenever a change is happening to its links

You can only create updated by timer indicator at design time.

To create an updated by timer indicator:

1. In **Script Editor**, choose **Indicator>Setup**.
2. In the **Class** field, change the value to **Timer**.
3. In the **Timer Interval** field, enter a timer interval in millisecond.
4. Press the **Apply** button.

In the indicator calculation, you can use the `ItSelf.TimerCall` property to determine whether a particular indicator calculation is triggered by timer (`TimerCall` is true) or by regular indicator update (`TimerCall` is false).

Usage

Using an updated by timer indicator is no different from using a regular indicator. You can add the updated by timer indicator to a chart, create links, etc.

If the indicator is intensive in calculation, you should reduce the firing frequency of an updated by timer indicator.

Initialization and Finalization

Usually you do not need to worry about initializing and finalizing an indicator. NeoTicker® takes care of that for you.

You will need initialization and finalization if you need to:

- Allocate a lot of lists in gheap. Because these lists are in gheap, they are not automatically disposed when the indicator is removed. So they will increase NeoTicker®'s memory use overtime if you do not dispose the list when the indicator is removed.
- Allocate memory in IDL indicator.

Initialization

Initialization happens at the first bar of an indicator, i.e. when an indicator is created by the user, or when the indicator is being recalculated.

In the indicator, you can use the `ItSelf.FirstCall` property to detect whether the indicator should perform any initialization tasks. Initialization is part of normal calculation, so the indicator should also performs its normal calculation and returns values.

The following is example code in Delphi Script:

```
function myindicator : double;
begin
    if ItSelf.FirstCall then
    begin
        // Performs initialization, e.g. allocate list in gheap
        ...
    end;

    // Performs calculation and returns values
    ...
end;
```

Finalization

Finalization happens when an indicator is removed, or just before the indicator is being recalculated. NeoTicker® will notify the indicator by calling the indicator with `ItSelf.RemoveCall` property set to true.

You must enable finalization when designing the indicator if you require the finalization call. To enable:

1. In Script Editor, choose **Indicator>Setup**.
2. Enable **Notify On Removal**.
3. Press **Apply** button.

Finalization call is separate from regular calculation. During finalization, the indicator should not expect anything other than gheap and pheap to contain valid values. The indicator does not need to return values.

The following is example code in Delphi Script:

```
function myindicator : double;
begin
    if ItSelf.RemoveCall then
    begin
        // Performs finalization, e.g. dispose list in gheap
        ...

        // Exits procedure right away. No calculation is required
        exit;
    end;

    // Performs calculation and returns values
    ...
end;
```

Identifying Indicator Instance

If you need to identify an instance of an indicator for memory allocation purpose, you can use the `ItSelf.UniqueId` property.

`ItSelf.UniqueId` returns a unique integer id for each instance of an indicator.

The following is example code in Delphi Script:

```
function myindicator : double;
begin
    if ItSelf.RemoveCall then
    begin
        // Performs finalization, e.g. use ItSelf.UniqueId to
identify what
        // to dispose in gheap.
        ...

        // Exits procedure right away. No calculation is required
        exit;
    end;

    if ItSelf.FirstCall then
    begin
        // Performs initialization, e.g. allocate list in gheap.
The list
        // is identified by ItSelf.UniqueId.
        ...
    end;

    // Performs calculation and returns values
    ...
end;
```

Making Arrays

Many objects such as ItSelf, LinkSeries, DataN support methods to create arrays. These methods create an variant array of data from the object.

The methods are `MakeArray`, `MakeArrayEx`, `MakeValidArray`, and `MakeValidArrayEx`. Once array is made, you can access the array using your language's array construct.

MakeArray (ArrayName, Type, Period, DirectOrder, InvalidDefault)

`MakeArray` returns a variant array.

`ArrayName` is a string. It is the name of the array.

`Type` is a string. It determines the type of value MakeArray generates.

If the underlying object is a data series:

Type Value	Meaning
'b'	Validity. The array will contains 0's and 1's. A value of 0 indicates an invalid bar. A value of 1 indicates a valid bar.
'c'	Close value. This value is meaningful only for data series type source.
'o'	Open value. This value is meaningful only for data series type source.
'h'	High value. This value is meaningful only for data series type source.
'l'	Low value. This value is meaningful only for data series type source.
'v'	Volume value. This value is meaningful only for data series type source.
'd'	Date Time value. This value is meaningful only for data series type source.

If the underlying object is not a data series (e.g. indicators, another array, etc).

Type Value	Meaning
'v'	Values.
'b'	Validity. The array will contains 0's and 1's. A value of 0 indicates an invalid bar. A value of 1 indicates a valid bar.
'c'	Same as v.

'o'	Same as v.
'h'	Same as v.
'l'	Same as v.
'd'	Date Time value. This value is meaningful only for data series type source.

Tip: use `'c'` if you want `MakeArray` to work for both data series and indicator.

`Period` is an integer. It is the length of the array.

`DirectOrder` is a boolean. It is the direction of the data. If the `DirectOrder` is set to false, the array is reverse chronological order. If the `DirectOrder` is set to true, the array is in chronological order.

`InvalidDefault` is a number. It is the value to use if a bar is invalid.

For example, consider the following VBScript statement:

```
Data1.MakeArray prices, "C", 5, false, -1
```

This statement creates an array object of length 5 called prices. The array elements are the close prices of `Data1`. The third parameter determines the chronological order of the array elements. Because this parameter is set to false, the array elements are in reverse chronological order, i.e. the first element of the array is the latest close price. The above statement creates an array that is equivalent to:

```
Array(Data1.Close(0), Data1.Close(1), Data1.Close(2),
Data1.Close(3), Data1.Close(4))
```

The last parameter -1 is the default value for invalid data points.

MakeArrayEx(ArrayName, N, Type, Period, DirectOrder, InvalidDefault)

`MakeArrayEx` is the multi-plot version of `MakeArray`. The parameter n indicates the plot, where n = {1..ItSelf.PlotCount} from the current indicator. For example,

```
ItSelf.MakeArrayEx(myarray, 2, 'V', 5, false, 0);
```

makes an variant array of size 5 using the indicator values of plot 2.

Calling `MakeArrayEx` with n=1 is equivalent to calling `MakeArray`.

MakeValidArray (ArrayName, Type, Period, DirectOrder)

`MakeValidArray` is similar to `MakeArray` except it skips over the invalid data.

For example, consider the following VBScript statement:

```
Data1.MakeValidArray price, "C", 5, false
```

Notice that compared to `MakeArray`, the `MakeValidArray` statement does not have a default invalid value. When an invalid bar is encountered, `MakeValidArray` skips the invalid bar.

MakeValidArrayEx(ArrayName, N, Type, Period, DirectOrder)

`MakeValidArrayEx` is the multi-plot version of `MakeValidArray`. The parameter n indicates the plot, where n = {1..ItSelf.PlotCount} from the current indicator. For example,

```
ItSelf.MakeArrayEx(myarray, 2, 'V', 5, false, 0);
```

makes an variant array of size 5 using the indicator values of plot 2.

Retaining Values Between Calculations

In previous programming examples, each bar of an indicator is calculated independently from other bars. Sometimes you want to retain values between bar calculations. NeoTicker® provides three objects for this purpose: `Heap`, `PHeap`, `GHeap`.

In this tutorial, we will dissect the previous day close indicator to show you how to use the Heap object.

Previous Day Close Indicator

The previous day close indicator is a good example for `Heap` object usage. In this indicator, the day close price is stored in the `Heap` object and returned to the caller.

The following listing is a VBScript version of the previous day close indicator, prevDCloseVB:

```
function prevDCloseVB()

    PrevClose   = 0
    CurrDate    = 1
    CurrClose   = 2
    HasValue    = 3

    ' Allocate heap
    if Heap.Size = 0 then
        Heap.Allocate(4)
        Heap.Value(CurrDate)     = Data1.DateTime(0)
        Heap.Value(CurrClose)    = Data1.Close(0)
        Heap.Value(PrevClose)    = 0
        Heap.Value(HasValue)     = 0
    end if

    ' If the day changed, mark that at least one
    ' previous day close value is available and
    ' record the value
    if Data1.Day(0) <> tq_day(Heap.Value(CurrDate)) then
        Heap.Value(HasValue) = 1
        Heap.Value(PrevClose) = Heap.Value(CurrClose)
    end if

    ' If there is a previous day value, return it
    if Heap.Value(HasValue) = 0 then
        ItSelf.Success = false
    else
        prevDCloseVB = Heap.Value(PrevClose)
    end if

    ' Update the values
    Heap.Value(CurrClose) = Data1.Close(0)
    Heap.Value(CurrDate)  = Data1.DateTime(0)
end function
```

The first part of the script defines some constants:

```
PrevClose   = 0
CurrDate    = 1
CurrClose   = 2
HasValue    = 3
```

These constants are later used to index into the Heap object.

Heap object needs to be initialized before use:

```
if Heap.Size = 0 then
    Heap.Allocate(4)
    Heap.Value(CurrDate)    = Data1.DateTime(0)
    Heap.Value(CurrClose)   = Data1.Close(0)
    Heap.Value(PrevClose)   = 0
    Heap.Value(HasValue)    = 0
end if
```

Each indicator has its own heap. In this code segment, the heap size is checked. If the heap size is 0, the script initializes it to a size of 4. Once the heap is initialized, it will stay at that size every time this indicator is calculated. You can increase the heap size, but not decreasing it.

The next step checks the date:

```
if Data1.Day(0) <> tq_day(Heap.Value(CurrDate)) then
    Heap.Value(HasValue) = 1
    Heap.Value(PrevClose) = Heap.Value(CurrClose)
end if
```

If the date has changed, then the close value in the heap becomes the previous day close value. The if statement here uses the tq_day function. This is a function that converts a date time value into a day value.

```
if Heap.Value(HasValue) = 0 then
    ItSelf.Success = false
else
    prevDCloseVB = Heap.Value(PrevClose)
end if
```

If a previous day close value has been found, it is returned. Otherwise the indicator is invalid.

```
   Heap.Value(CurrClose) = Data1.Close(0)
   Heap.Value(CurrDate)  = Data1.DateTime(0)
```

Store the close value and the date time into the heap. The next time this indicator is calculated, these values will be available.

Difference Between Heap, PHeap and GHeap

The methods are the same across Heap, PHeap and GHeap. The difference lies in the way they handle the persistency of data.

When a tick comes into an indicator, Heap restores itself to the initial state of the bar the tick belongs to before the indicator script is executed. This ensures a consistent value when the indicator is calculated from ticks and when the indicator is calculated from historical minute bars.

PHeap does not restores itself to the initial state of the bar. Thus storage in PHeap is persistent from tick to tick. PHeap is useful when the indicator's correctness depends on this persistency. The drawback is it is not possible to ensure indicator value calculated from ticks will agree with indicator value calculated from historical minute bars.

For most usage, Heap is a better choice than PHeap because of the consistency Heap ensures. Use PHeap only if you need the tick to tick persistency it offers. Methods and properties of Heap and PHeap are identical.

GHeap is similar to PHeap, i.e. it does not restore to initial state of the bar. The difference is GHeap is shared by all indicators, for all charts, quote window, etc. The main purpose of GHeap is for data communication between indicators.

Single Plot vs. Multiple Plot Indicators

Each indicator can have one or more plot associated with it. Each plot is a separate series of data.

If an indicator method do not have a plot parameter, it is by default working on the first plot. If you are working on a single plot indicator, this is good because you do not have to manually specify the plot.

If you are working on a multiple plot indicator, you should look for the EX version of indicator methods. These method has a plot parameter that allows you to specify the plot you are working on.

Query the Number of Plots

To query the number of plots in the indicator or a data link, you can use the `PlotCount` property. For example,

```
ItSelf.PlotCount
Data1.PlotCount
```

Accessing Values and Validity

The properties `Value` and `Valid` return the value and validity of the indicator or the data link.

If you need to access the value and validity of a specific plot, use `ValueEx` and `ValidEx.` For example,

```
Data1.ValueEx(3, 0)
Data1.ValidEx(3, 0)
```

These methods return the current bar value and validity of the third plot of the first data link. In `ValueEx` and `ValidEx`, the first parameter is the plot number, which ranges from 1 to `PlotCount`. The second parameter is the bar number.

Returning the Indicator Validity

The indicator validity is assigned by the `ItSelf.SuccessEx` property. The argument specifies the plot. The single plot version `ItSelf.Success` assigns the validity of the first plot. For example,

```
ItSelf.SuccessEx(3)
```

assigns the validity of the third plot.

There is a `SuccessAll` property you can assign to. This property will assign the success status to all plots.

Returning the Indicator Value

Single plot indicator returns the indicator value using function returns, i.e. in VBScript, by assigning to the function name or in Delphi Script, assigning to the result variable. For example, in VBScript,

```
mymov = 10
```

returns the value 10 to a function named `mymov`.

Multiple plot indicators are a bit different. You need to assign the indicator values to the `Answer` property. For example,

```
ItSelf.Answer(1) = 10
ItSelf.Answer(2) = 20
ItSelf.Answer(3) = 30
```

will assign the value 10, 20, 30 to plot 1, 2, 3 respectively.

The property `Plot` is identical to `Answer`. The choice is a personal preference. For example,

```
ItSelf.Plot(1) = 10
ItSelf.Plot(2) = 20
ItSelf.Plot(3) = 30
```

perform exactly the same operation as assigning to `Answer` in the previous example.

PlotValue Indicator

`PlotValue` is an indicator that helps you map a specific plot to plot 1. It is useful when you need to send a specific plot to an indicator that expects a single plot.

`PlotValue` can be created visually in charts and scanners, or by the `ItSelf.MakeIndicator` method in scripts.

Making Array

`MakeArrayEx` and `MakeValidArrayEx` are the multiple plot version of `MakeArray` and `MakeValidArray`. The plot is specified after the array name. For example,

```
ItSelf.MakeValidArrayEx myarray, 2, "V", 5, false
```

makes an array from the second plot with 5 elements.

Time Frame Information

You can query about time frame information for objects such as ItSelf, DataN, LinkSeries.

The supported properties for these objects are `TimeFrameType` and `TimeFramePeriod`.

For example, if `ItSelf.TimeFrameType` equals ppMin, and `ItSelf.TimeFramePeriod` equals 5, then the indicator is currently working in a 5-minute bar environment.

For example, if `Data2.TimeFrameType` equals ppSec and `Data2.TimeFramePeriod` equals 30, then the second linked data has 30-second bars.

TimeFrameType

`TimeFrameType` returns the time frame of the current instance of the indicator. It is an integer property. It can be any one of the following,

Value	Meaning
ppTick	tick based (e.g. 1-tick driven bar, 50-tick driven bar)
ppSec	second based (e.g. 20-second bar, 45-second bar)
ppMin	minute based (e.g. 1-minute bar, 5-minute bar)
ppHour	natural hour based (e.g. 1-hour bar, 5-hour bar)
ppDaily	daily data based (e.g. 1-day bar, 3-day bar)
ppWeekly	weekly data based
ppMonthly	monthly data based
ppQuarterly	quarterly data based
ppYearly	yearly data based

TimeFramePeriod

`TimeFramePeriod` returns the bar size of the current instance of the indicator. It is an integer property.

Updated by Primary Link Only

For an indicator with multiple links, i.e. has data1, data2, and more, when one of the link updates, it will trigger the calculation on the indicator.

You can force the indicator to update only when the primary link (data1) updates. This is a setting useful for trading systems when system correctness depends on the number of times the system is updated.

To set update by primary link only in design time:

1 In **Script Editor**, choose **Indicator>Setup**.

2 Check **Primary Link Only**.

3 Press the **Apply** button.

This property can be overridden in run time in indicator editor, under **Other** tab, by changing the **Update Dependency** setting.

Using Other Indicators

An alternative to call other indicators is by using formula within a script/program. To find out more, refer to ***Quick Combination of Multiple Indicators*** (on page 1395).

You can call other indicators to perform calculations. This allows you to build an indicator using another indicator in scripts. The following methods are related to calling indicators.

The methods `ItSelf.MakeIndicator` and `ItSelf.MakeIndicatorEx` are for creating indicator instances.

The methods `ItSelf.UpdateIndicator` and `ItSelf.UpdateIndicatorEx` are for updating indicator instances.

The methods `ItSelf.Indicator` and `ItSelf.IndicatorEx` are for referencing indicator instances.

Creating an Indicator with MakeIndicator and MakeIndicatorEx

In order for the script to use another indicator, first you need to create an indicator instance. An indicator instance takes the indicator definition, and apply the necessary data link and parameters to it. For example, simple moving average is an indicator definition. By applying the parameter 10 and link it to Data1, you have an indicator instance of 10-period simple moving average on Data1.

You can use the `ItSelf.MakeIndicator` or `ItSelf.MakeIndicatorEx` method to create an indicator instance.

Consider the following code(VBScript):

```
ItSelf.MakeIndicator "SD", "stddev", Array("1"),
Array(Param1.str)
```

and

```
ItSelf.MakeIndicator "avg", "mov", Array("1"),
Array("Simple",Param1.str)
```

Consider the following code (Delphi Script):

```
ItSelf.MakeIndicator('SD', 'stddev', ['1'], [Param1.str]);
```

and

```
ItSelf.MakeIndicator('avg', 'mov', ['1'],
['Simple',Param1.str]);
```

These lines use the `MakeIndicator` method to create a standard deviation indicator and a moving average indicator for the script to use.

`MakeIndicator` method takes four parameters - name, function name, data link and parameters.

The first parameter is the name you give to the indicator instance for later referencing. In this example, `"SD"` is given to the standard deviation indication and `"avg"` is given to the moving average indicator.

The second parameter is the function name of the indicator to create. The function name for standard deviation and moving average are `"stddev"` and `"mov"` respectively. You can find a listing of the function names using the indicator quick reference, which can be opened by choosing **Help>Indicator Quick Reference** in the script editor.

The third parameter specifies the data link to the indicator. Data link is the input to the indicator. A link can be either a data series or an indicator. In the lines above, `Array("1")` specifies that the first data link of the hosting indicator is used as the input to the created indicator. For indicator that requires a second data link, specifies the second link in an array. For example, in a subtraction indicator, use `Array("1", "2")`.

By default, the close price or indicator value is used in the third parameter. You can specify the parameter such that other values are used. For example, `"1.h"` specifies the high value of the first data link is used. For a complete list of attributes, refer to the make indicator reference below.

The fourth parameter specifies the parameters that are passed to the indicator. All elements of the array must be presented in string form. Therefore, in the moving average indicator, `Param1` is read as a string even though it is actually an integer. For example, if you need to create a 10 period moving average, you will write a line like (VBScript):

```
ItSelf.MakeIndicator "avg", "mov", Array("1"), Array("Simple",
"10")
```

Delphi Script and Borland Delphi users: Use square brackets to create an array, e.g. `['Simple', Param1.str]`.

`MakeIndicatorEx` method takes five parameters - name, unique id, function name, data link and parameters.

The additional unique id parameter is an integer you choose for accessing the indicator. Unique id is an alternative to the "name" parameter. For indicators written in IDL, using unique id to access indicators can provide significant performance advantages.

The following are examples of calling `MakeIndicator` and `MakeIndicatorEx`. In the `MakeIndicatorEx` example, the unique id 35 is chosen by the programmer.

Delphi Script

```
Itself.MakeIndicator ('MA1', 'average', ['1'], ['50']);
Itself.MakeIndicatorEx ('MA1', 35, 'average', ['1'], ['50']);
```

VBScript

```
Itself.MakeIndicator "MA1", "average", Array ("1"), Array
("50")
Itself.MakeIndicatorEx "MA1", 35, "average", Array ("1"), Array
("50")
```

At this point you may want to look at examples of MakeIndicator in ***Example: BBandVB Script*** (on page 1486) and ***Example: TRIX*** (on page 1487) to see how the call works.

`MakeIndicator (or MakeIndicatorEx)` creates and updates an indicator instance within a script. When a `MakeIndicator` call is first encountered, it automatically creates the indicator instance. NeoTicker® keeps track of this instance of the indicator. So if you re-declare the same indicator with the same name, a new instance will not be created. Therefore, you can use `MakeIndicator` anywhere in the script just before you need the indicator instance without worrying about memory usage.

Once an indicator instance is created, you cannot change the underlying indicator nor its parameters.

When `MakeIndicator` is called, it returns an indicator instance object. You can simply store the object in a variant variable for later use. For example (in Delphi Script):

```
obj := Itself.MakeIndicator ('myavg', 'average', ['1'],
['100']);
result := obj.Value[0];
```

Updating an Indicator with UpdateIndicator and UpdateIndicatorEx

Just like any other indicators, indicator instances created inside a script needs updating (i.e. calculating the latest value). This is usually done automatically by the `MakeIndicator` (or `MakeIndicatorEx`) call and you do not need to worry about updating the indicators.

However, to squeeze out extra performance out of a script, you may want to manually updating an indicator, bypassing `MakeIndicator`.

Consider the following code for making and using the moving average indicator.

```
obj := Itself.MakeIndicator ('myavg', 'average', ['1'],
['100']);
result := obj.Value[0];
```

The above code works, but is not the most efficient way to use `MakeIndicator`. An indicator is run for every tick/bar, so most of the time, `MakeIndicator` does not create an indicator instance, but instead updates and returns an indicator instance that already exists. By calling `MakeIndicator` for every bar/tick, you are doing a lot of unnecessary work.

The method `UpdateIndicator` and `UpdateIndicatorEx` allows you to manually update an indicator without the unnecessary work to improve performance.

`UpdateIndicator` and `UpdateIndicatorEx` returns an indicator instance if it is already create, nil otherwise.

For example, the code above can be rewritten as:

```
obj := Itself.UpdateIndicator ('myavg');
if obj = nil then
   obj := Itself.MakeIndicator ('myavg', 'average', ['1'],
['100']);
result := obj.Value[0];
```

Or with `UpdateIndicatorEx`:

```
obj := Itself.UpdateIndicatorEx (35);
if obj = nil then
   obj := Itself.MakeIndicatorEx ('myavg', 35, 'average',
['1'], ['100']);
result := obj.Value[0];
```

Note that in the code above, `MakeIndicator (or MakeIndicatorEx)` is called only the first time the script is run. By avoiding excessive call to `MakeIndicator`, performance is improved.

See ***Example: UpdateIndicator*** (on page 1488) for another example.

Referencing an Indicator

In section above, we see that `MakeIndicator, MakeIndicatorEx, UpdateIndicator` and `UpdateIndicatorEx` methods return an indicator instance object you can use.

In addition, you can reference an indicator instance by its name using the `ItSelf.Indicator` method, or by its unique id using the `ItSelf.IndicatorEx` method. For example (in VBScript):

```
AStdDev = ItSelf.Indicator("SD")
AMean = ItSelf.IndicatorEx(35)
```

Or in Delphi Script:

```
AStdDev := ItSelf.Indicator('SD');
AMean := ItSelf.IndicatorEx(35);
```

These calls return indicator instance objects you can use the same way as a data link. Methods such as `Value` and `Valid` are all available.

Delphi Script: Declare `AStdDev` and `AMean` with no type information, i.e. variants.

Borland Delphi: Declare `AStdDev` and `AMean` as variants.

Referencing a Specific Plot in Input Source

When making indicators that uses another indicator as the link, you can specify which plot to use by using the `.plotN` or `.pN` suffix. Consider the following Delphi Script example,

```
makeindicator('myindicator', 'bbands3', ['1'], ['20', '1']);
makeindicator('myosc', 'slowk', ['myindicator.plot3'], ['5',
'3']);
```

The example applies a slowk indicator on bband3 (Bollinger Band, 3 plots). Because bband3 has 3 plots, the example uses the `myindicator.plot3` to apply the slowk on the third plot of bbands3.

MakeIndicator Reference

`MakeIndicator` is a method of the `ItSelf` object and has the following syntax:

```
ItSelf.MakeIndicator (Name, Indicator, [Source1 {, Source2,
...}], [Param1 {, Param2, ...}])
```

Item	Explanation
`{}`	Items in curly brackets are optional
`Name`	The name of the newly create indicator instance. It is not case sensitive. This name will be used to reference the indicator instance in other parts of the script
`Indicator`	The indicator function that you want to create an instance for. Either the actual function name or the description can be used here. For example, you can use the function name 'average' or the full description 'Simple Moving Average'. You can look up the indicator name under **Help>Indicator Quick Reference** in script editor

`Source1 ...` These parameters specify the source data for the indicator instance. They are strings that identify either the original data sources for the script itself ('1' for Data1 object, '2' for Data2 object), or the name of another valid series created within the same script (e.g. another indicator).

For `Data1`, `Data2` and `LinkSeries` the user can specify a specific field to obtain the value. If a field is not specified, the default value of the data object is used. For example,

`'1'`	default value of Data1
`'1.c'` or `'1.close'`	close of Data1
`'1.o'` or `'1.open'`	open of Data1
`'1.h'` or `'1.high'`	high
`'1.l'` or `'1.low'`	low
`'1.v'` or `'1.vol'`	volume
`'1.n'` or `'1.oi'`	open interest
`'1.t'` or `'1.tick'`	ticks
`'1.d'` or `'1.datetime'`	date time

For items referenced using the `DataSeries` object, add the '#' sign before the number. For example,

`'#1'`	default value of `DataSeries.Items[1]`
`'#2.h'`	the high value of `DataSeries.Items[2]`

For Series created from `CompressSeries` or `SetSeries,` use the name of the series with the '$' sign as prefix.

You can use an * before the name of an indicator series to designate the indicator is to be accessed like a data series, similar to Meta Plot Style.

To link to a specific plot of an indicator, use `.plotN` or `,pN`. For example, `myindicator.plot3`.

`Param1 ...` They are the parameters that you specify in the indicator setup window. Provide all parameters in string only. For example, if you need to provide a value of 50, use `'50'` or `"50"` instead

Example: BBandVB Script

In this example, we will dissect a single plot Bollinger band VBScript and see how these methods work.

The following VBScript implements the Bollinger band indicator. This script has one plot and takes two parameters (offset and period). The listing includes the header so you can simply cut and paste the code into a script editor to try it. (Make sure you enable the script header display by choosing **Visual>Show Script Header** before pasting the script). You can install the script and apply it on a data series. You will see that it does exactly the same calculation as the indicator bband.

```
# NEOTICKER DATA BEGIN
ScriptType=Indicator
Description=Bollinger Band VB
Name=bbandvb
Language=VBScript
Links=1
MinBars=0
Multiplot_num_plots=1
Multiplot_color_0=255
Multiplot_style_0=Line
Multiplot_width_0=1
Multiplot_enabled_0=1
UpdateByTick=1
Param_count=2
Param_name_0=period
Param_inuse_0=1
Param_type_0=integer.gt.1
Param_default_0=10
Param_name_1=offset
Param_inuse_1=1
Param_type_1=integer.gt.1
Param_default_1=1
Explanation_Lines=1
Explanation0=Example VB implementation of Bollinger band
# NEOTICKER DATA END

function bbandvb()
   ' Create Standard Deviation indicator
   ItSelf.MakeIndicator "sd", "stddev", Array("1"),
Array(Param1.str)
   AStdDev = ItSelf.Indicator("SD")

   ' Create Moving Average indicator
   ItSelf.MakeIndicator "avg", "mov", Array("1"), Array("Simple",
Param1.str)
```

```
   AMean = ItSelf.Indicator("avg")

   if Data1.Valid(0) then
       if AStdDev.Valid(0) and AMean.Valid(0) then
           ' Calculate Bollinger band value
           bbandvb = AMean.Value(0) + AStdDev.Value(0) *
Param2.Real
       else
           ItSelf.Success = false
       end if
   else
       ItSelf.Success = false
   end if
end function
```

Example: TRIX

The following is an example in Delphi Script of applying nested indicators using the `MakeIndicator` and `Indicator` methods.

```
function trix : double;
begin
  with itself do
  begin
    makeindicator ('ma1', 'xaverage', ['1'], [param1.str]);
    makeindicator ('ma2', 'xaverage', ['ma1'], [param2.str]);
    makeindicator ('ma3', 'xaverage', ['ma2'], [param3.str]);
    makeindicator ('answer', 'roc', ['ma3'], [param4.str]);

    result := indicator ('answer').value [0];
  end;
end;
```

The following is the VBScript version of the TRIX indicator.

```
function vb_trix()
  itself.makeindicator "ma1", "xaverage", Array ("1"), Array
(param1.str)
  itself.makeindicator "ma2", "xaverage", Array ("ma1"), Array
(param2.str)
  itself.makeindicator "ma3", "xaverage", Array ("ma2"), Array
(param3.str)
  itself.makeindicator "answer", "roc", Array ("ma3"), Array
(param4.str)

  vb_trix = itself.indicator ("answer").value (0)
end function
```

Example: MakeIndicator Meta Style Support

When applying indicator on another indicator inside a script using `MakeIndicator`, you can specify that the source indicator behaves like a data series. This is equivalent to meta plot style when using indicators interactively.

The source indicator need to have 4 plots or more. The first 4 plots will become OHLC values when it is being referenced by the indicator applied on it.

All you need to do is add an * before the indicator name when referencing the source indicator.

Example Usage (DelphiScript):

```
// my4plotindicator is an indicator that has 4 plots.  Create an
indicator call M4P.
ItSelf.MakeIndicator ('M4P', 'my4plotindicator', ['1'], []);

// Applying moving average on M4P, treating M4P as a metaplot
indicator. In this
// case, moving average will work on plot 3 (the close value),
rather than plot 1.
ItSelf.MakeIndicator ('MA1', 'average', ['*M4P'], ['50]);
```

Example: UpdateIndicator

We've discussed by eliminating excessive calls to `MakeIndicator` using `UpdateIndicator`, you can improve the performance of a script.

This example is a complete Delphi Script that illustrates this coding technique. The indicator created is an RSI indicator.

```
function myrsi : double;
var
    obj;
begin
    obj := Itself.UpdateIndicator ('myrsi');
    if obj = nil then
        // Avoid calling MakeIndicator excessively to improve
performance
        obj := Itself.MakeIndicator ('myrsi', 'RSIndexMod', ['1'],
['14']);
    result := obj.Value[0]
end;
```

Visual Break Programming

Typically, a user can set an indicator to break on invalid data points.

If you are writing an indicator, you have more flexibility to create visual breaks. In an indicator script/IDL, simply set the `ItSelf.VisualBreak` property to true if you want to create a break in the indicator.

To access previous visual break values, use the `ItSelf.PrevVisualBreak` and `ItSelf.PrevVisualBreakEx` properties.

See the properties and methods section of ***ItSelf Object*** (on page 1525) for a description of these properties.

When apply indicator with visual break, you will need to set the indicator to use **Visual** setting. See ***Broken Lines*** (on page 1123).

Trading System Programming Topics

Overview

Consider using Backtest EZ before trying to write trading in scripts. Backtest EZ takes care of many trading system work.

What are Trading Systems

Trading systems are mechanical models that make trading decision. In NeoTicker®, trading systems are implemented as specialized type of indicators.

As specialized indicators, trading systems has all the tools of indicators at its disposal.

It's a good idea you get yourself familiar with indicator programming. You can follow the tutorials ***Creating an Indicator, Example 1*** (see "Tutorial: Creating an Indicator, Example 1" on page 1369) and ***Creating an Indicator, Example 2*** (see "Tutorial: Creating an Indicator, Example 2" on page 1383).

What Distinguishes Trading Systems from Regular Indicators

Two things:

- Trading systems make use of `Trade` object. Trade object provides properties and method for trading related activities such as placing order, setting up commission, etc.
- The option **Trading System UI** when you design the indicator. While you can still implement a trading system without turn on this option, having this option on makes the user's life a lot easier because he/she can easily set commissions, fill type etc. from the UI, and performance reporting is available.

Example Trading System Scripts

The script files:

- `sys_mov_avg_crossover.pas`
- `sys_hist_mov_avg_crossover.pas`
- `sys_xavg_crossover_vb.vbs`

are example trading system scripts. They can be found in the `indicator` and `source\indicator source` folders in the NeoTicker® installation.

Trade Object Reference

Go to ***Trade Object*** (on page 1577).

Steps to Writing a Trading System

Creating the Trading System

Creating a trading system is just like creating an indicator. You will use the script editor to open and edit the trading system. In this script editor, you set it things like your preferred language, etc.

Under most cases, you will want to turn on the **Trading System UI** option.

Before the Trading Logic

You will want to set up the trading system properly before the actual trading logic. Things you want to set are whether you want to close the position automatically at end of trading day, commission, etc.

Many of initial settings can be set during run-time by the user. You only need to set things to override user settings.

Logic behind the Trading System

A trading system works by placing orders to Trade object.

As a system writer, your job is to find out the right time to enter a position, and exit the position. Generally speaking, you will create signals. Then you will check these signal if it is the right time to enter/exit positions.

Commonly used signals include moving average cross over and various kinds of divergence. NeoTicker® provides a rich set of tools to help you construct signals.

Because trading system are indicators and indicators expect a return value, a common convention is to return the equity curve of the trading system as the indicator value.

Entering Orders

Once signals are detected, your trading system will enter orders through the `Trade` object.

`Trade` object provides a rich set of order placement methods. Not all brokers implement all of the order placement methods provided. When you design a trading system, you should keep that in mind. Otherwise you may end up with a trading system that is impossible to carry out due to broker limitation.

Installing the Trading System

Once you complete the logic of the system. You will need to install the trading system so NeoTicker® is aware of its existence.

Installing a trading system is just like installing an indicator, you can refer to a the following section in the tutorial: ***Installing Script*** (on page 1379).

Testing the Trading System

To test a trading system:

1. Create a chart
2. Add appropriate data series to the chart
3. Add indicator. Choose the indicator that implements the trading system.
4. You can adjust the parameters of the trading system and re-apply for different testing.
5. If you modify the trading system script itself, you need to re-install the trading system and refresh the chart.

Performance Analysis

A trading system will give you many information on how it performs: it will mark the orders and trades on the chart and it will have details statistics collected inside the `Trade` object.

One way to extract the statistics from `Trade` object is to use the system performance viewer. You can launch system performance viewer directly from a chart. Alternatively, you can use the reporting methods to for a text-only presentation.

Typically, you will find various weakness in a trading system. By analyzing its performance, you will find ways to improve it by modifying the signals and improving the money management side of the trading system.

Once the improvements are made, you will test the trading system and analyze its performance. The cycle continues until you are satisfy with the trading system.

Real-time Deployment

This section is not applicable to NeoTicker® EOD.

Once you decide a trading system is good enough, you will deploy it for real-time use. There are two ways you can do this:

- Use system monitor to observe the orders being made and follow the orders manually.
- Add an automatic order entry system. Trading system scripts can drive ActiveX objects, so an order entry system is simply a relay from the trading system to your broker's order entry system.

Make sure update by tick is off if you plan to deploy the system in real-time. Failure to do so will result in **serious monetary consequences**. To know why this is the case, read ***Real-time Deployment*** (on page 1511).

Trading Sytstem UI

To turn on trading system ui:

1. When you design the trading system in the script editor, choose **Indicator>Setup**.
2. Enable the option **Trading System UI**.
3. Re-install the indicator.

Having this option enabled will make the following services available in run-time to the user of the trading system:

- When the user edit the indicator that implements the trading system, the **System** and **Report** tab will be available. Under these tabs, the user can provide settings for commission structure, initial capitals, reporting options.
- When the user select the trading system, the **Trading System** item will be available in the pop up menu. This item lets the user performs quick adjustment to the trading system as well as providing reporting facilities.

Setting up Pane Settings for Trading System

When you set up an indicator as a trading system, you may want to pay some attention to the pane setting.

By convention, a trading system returns the equity curve. Equity curve are usually is a very large positive number and do not mix display with the data series in the same pane.

You can instruct the trading system to always be created in a new pane by:

1. Open the trading system in script editor.
2. Choose **Indicator>Setup**.
3. Under the **Pane Options** tab, set **Indicator Placement** to **New Pane**, **Value Range** to **Other**.

Returning the Equity Curve

Trading system convention is to return the equity curve in plot 1 of the indicator. This serves some purposes:

- In time chart, the equity curve is immediately visible.
- If you scan the trading system, you can easily rank the trading system by equity curve.

To set equity curve, the trading system will include a statement like:

```
Delphi Script - ItSelf.Plot[1] := Trade.CurrentEquity;
VBScript - ItSelf.Plot(1) = Trade.CurrentEquity
```

Initial Money Related Settings in Trading System

Initial settings include the following values as `Trade` object properties.

- Initial capital (`InitialCapital`)
- Interest rate (`InterestRate`)
- Margin (`MarginType, MarginValue`)
- Commission (`CommissionType, CommissionBaseAmount, CommissionPerUnit`)

Descriptions can be found at ***System Setting Properties.*** (see "System Setting Properties" on page 1604)

Normally, you will want to leave these values alone as they can be set in run-time by user under the **System** tab in indicator edit. This is preferable as the user can try the trading system under different conditions.

If you set these properties, they will override the settings made by the user.

Portfolio Trading System Settings

Margin requirement can be set individually for each data series that is being traded. For more information, refer to ***Portfolio System Setting Properties*** (on page 1601).

Price Multiples and Minimum Tick Size

Price multiples determine how much money each point in an instrument worth. For stocks, price multiple is 1 (each point worth $1). For E-mini SP500, price multiple is 50 (each point worth $50).

Minimum tick size specifies the minimum change in the symbol. For example stocks, it is $0.01. For E-mini SP 500, it is 0.25 point.

Descriptions can be found at ***System Setting Properties.*** (see "System Setting Properties" on page 1604)

Normally, you will want to leave these values alone as they can be set in run-time by user under the **System** tab in indicator edit. This is preferable as the user can try the trading system under different conditions.

If you set these properties, they will override the settings made by the user.

Portfolio Trading System Settings

These values can be set individually so you can trade different instruments in the same portfolio trading system. For example, it is possible to create a spread trading system that works on E-mini SP 500 and some large cap stocks.

For more information, refer to ***Portfolio System Setting Properties*** (on page 1601).

Closing a Position at End of Day

For intraday trading systems, you can set all positions to be closed at the end of the trading day. The property is `Trade.ClosePositionEOD`.

Single Entry Per Direction

If you want your trading system to only long/short one position (i.e. you can't long 100 shares, then long another 100 shares), you can force this condition by setting the boolean property `Trade.SingleEntryPerDirection` property. When this property is true, and you attempt to enter more than one position for a direction, the order will be ignored.

This property is useful if your system is consist of a many long/short signals, but you want to control the commitment of your trading system for a direction.

See ***Order Defaults and Completion Related Topics, Single Entry Per Direction*** (see "Single Entry Per Direction" on page 749) for more information on how orders are blocked.

Trading Time Range for Historical Testing

You can control the trading time range by setting the `UseTimeRange`, `TimeRangeStart` and `TimeRangeEnd` properties of Trade object.

Controlling time range let you reproduce historical testing result without being affected by new data.

Order Placement

All orders placed at the current bar will be active from the next bar onward. Orders can stay active for as long as they are supposed to. You can cancel active orders.

Time In Force

Orders that are not market order can be one of the following time in force parameter:

`otfFillorKill`	If order is not filled by the next bar, just cancel it
`otfDay`	Order will stay for the day only
`otfGoodtilCancel`	Order is placed to stay in the system until it is filled or cancelled

For all orders, you can specify the size of the trade. For example, if you want to instruct the system to buy 100 shares at the market then the call would look like the following,

```
VBScript      - Trade.BuyAtMarket 100, "My Buy Signal"
Delphi Script - Trade.BuyAtMarket (100, 'My Buy Signal');
```

The comment "My Buy Signal" is the comment of the trade. For all orders, you can attach a comment when the order is placed. The comment can be programmatically changed for each order and can store information aside from just simple naming of the signal. For example, if your order is a cost averaging entry and you want to identify that this is the Nth entry in the same direction, then the order may look like this:

```
VBScript     - Trade.BuyAtMarket 100, "Buy # " + tq_inttostr
(N)
DelphiScript - Trade.BuyAtMarket (100, 'Buy # ' + tq_inttostr
(N));
```

Order ID

All order methods return an order id. You can save the order id to later reference the order.

Basic Order Methods vs. Smart Order Methods vs. Special Order Methods

Basic order methods are simple buy sells. Smart order methods provide management of long/short positions for you. Additional special order methods are provided to handle orders at a high level.

Smart order methods are easier to write, but may need adjustment for real-time deployment because your broker may not be smart enough to handle these orders.

For more information, see ***Order Placement - Basic Buy Sell Order Methods*** (on page 1580), ***Order Placement - Smart Order Methods*** (on page 1586) and ***Order Placement - Special Order Handling Methods*** (on page 1588)

Querying Open Order

Methods are provided to query open orders.

See ***Order Properties and Methods*** (on page 1589).

Multiple Exit Orders

An important design issue must be kept in mind when placing multiple exit orders. The ones placed first always get the highest priority and if they got filled first, and eventually changed the current position size, the rest of the exit orders will not be executed. For example, if you have a long position of 1000 shares at the moment, and then a `LongExitAtMarket` is placed for 500 shares and a `LongExitLimit` is placed for 1000 shares at the same time. The first order, `LongExitAtMarket` will be executed first and reducing the current open position to long 500 shares only. Thus even if the condition is met for the `LongExitLimit` in the same bar, it will have a fill of 500 shares only.

Placing Orders to Trade Simulator and Real Life Broker

You can programmatically place an order to Trade Simulator, or a Real Life Broker. The methods are described in ***Order Placement - Order Interface Methods*** (on page 1585).

For a comprehensive discussion on order interface, refer to ***Order Interface Manager*** (see "Order Interface" on page 735).

Portfolio Trading System Order Placement

For order placement methods in a portfolio trading system, refer to ***Portfolio Order Placement Methods*** (on page 1598) and ***Portfolio Special Order Handling Methods*** (on page 1600).

Order Filling Mechanism

This section deals with how orders are filled in trading system testing. It is not applicable if your trading system is wired to a live broker and sending orders there for execution.

In system testing, we need some kind of order filling mechanism. During testing, suppose your trading system sends out an order, NeoTicker® needs to decide if the order can be filled, and if filled, what is the fill price. Unlike in real life, historical data only provides a summary (e.g. 1-minute bars) and it is not possible to determine an exact fill price for testing purpose.

In this section, we discuss how orders are filled in a system testing context. You will find that the performance of trading system in historical testing often depends on how optimistic you set these parameters.

Property

Order filling is controlled by the `Trade.FillType` property. Worst fill is the default.

See ***System Setting Properties*** (on page 1604) for more information about the property itself.

Orders Activated on Next Bar

After orders are placed at a specific bar, it is activated at the beginning of the next bar. Thus the first bar that an order can get filled is the bar right after the order is placed.

Transactions are Available on Filled Bar

If the order is filled on a particular bar, a transaction is generated on this bar. You can access this transaction information in your script within the same bar. You can then have rules of money management to generate orders to protect the position based on the transaction just happened.

Minimum Tick Size

If minimum tick size is set through the properties `MinTickSize` and/or `MinTickSizes`, then the orders are filled at prices rounded towards the nearest tick. For example, `MinTickSize` at 0.05 will enforce orders to fill at 1.00, 1.05, 1.10, etc.

No “On Close” Orders

There is no filling mechanism that tries to fill with the close of the current bar. It is against the principle of not allowing the act of peeking into the future. Remember you have access to all the data of the current bar, which includes the close of the bar. Close of a bar in principle is not known to the trading system until data for the next bar has arrived.

By having on close orders available, a trading system can check the close for profitability first, then place the on close order to close out a position with profit. That makes sense in the days when there was no intraday data available. Nowadays, you can always use intraday data to properly design the order placement rules.

The way to make an order with close price is by using the next close orders (`BuyNextClose`, `SellNextClose`, etc). These ordering methods allow you to place orders that will be filled at next bar's close. Next bar close orders do not peek the close price of current bar, yet provide a mechanism for filling an order based on close price.

Rules of Order Filling

The following are the rules how the price for a fill is determined.

Limited Worst Case

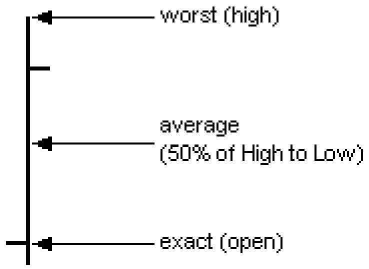
Limited worst case is similar to worst case, but provide a limit on the worse scenario.

In limited worst case, the fill price is either:

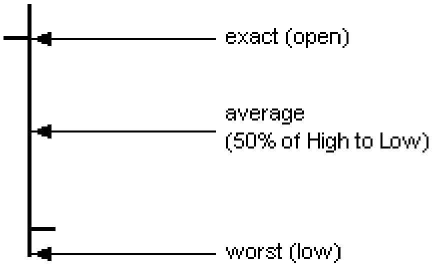
- the worst price, or
- exact plus/minus the slippage value (Order object property)

The better fill is used.

Buy at Market

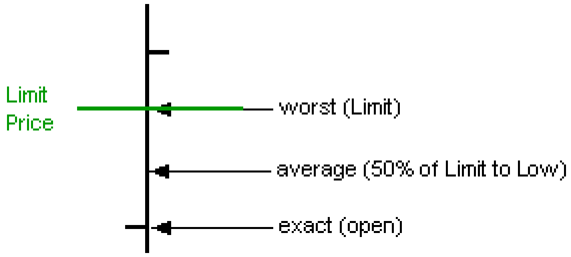


Sell at Market

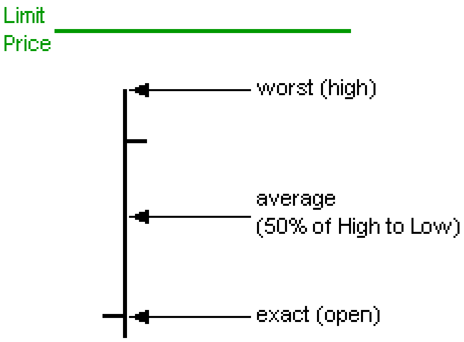


Buy Limit

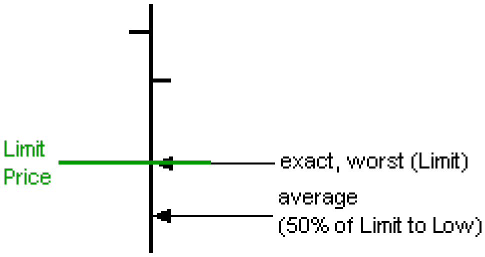
Case 1



Case 2

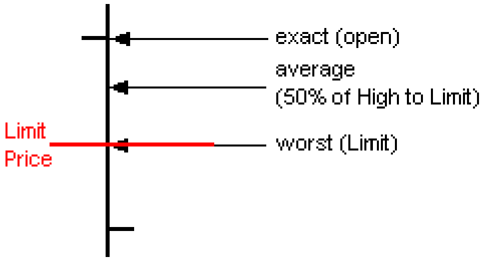


Case 3

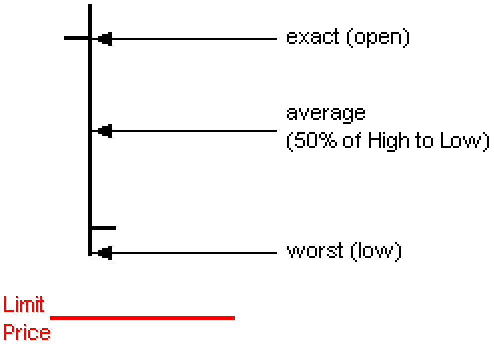


Sell Limit

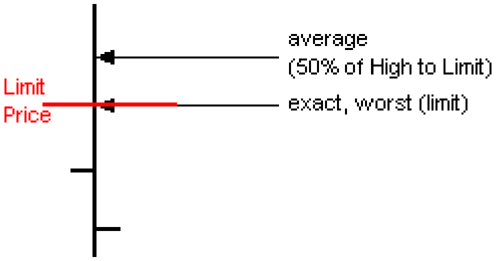
Case 1



Case 2

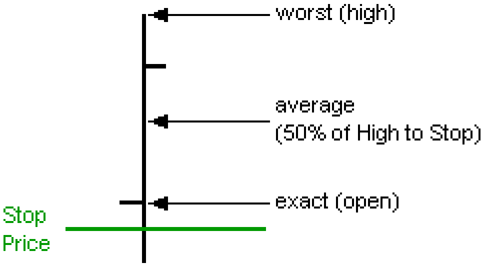


Case 3

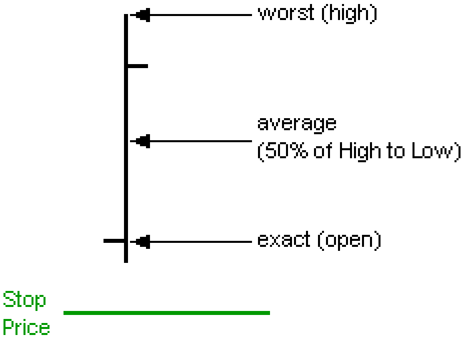


Buy Stop

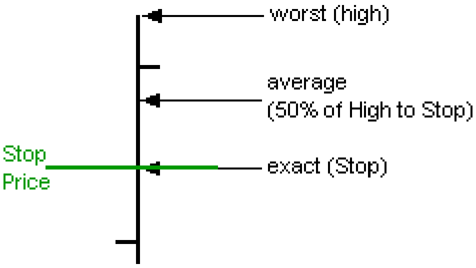
Case 1



Case 2

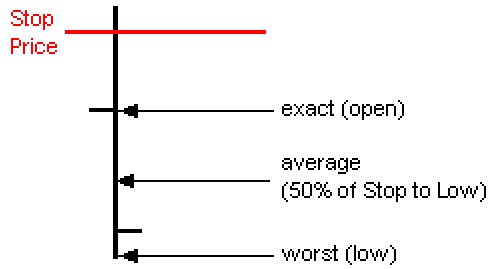


Case 3

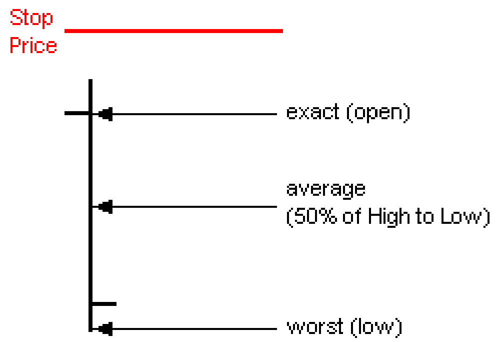


Sell Stop

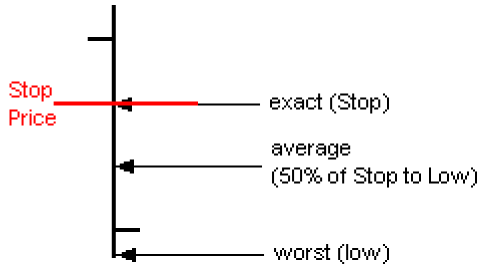
Case 1



Case 2



Case 3



Position Management

For open positions, refer to ***Open Position Management Properties and Methods*** (on page 1593).

For closed positions, refer to ***Closed Positions Properties and Methods*** (on page 1578).

For position management information for portfolio trading systems, refer to ***Portfolio Open Position Management*** (see "Portfolio Open Position Management Properties and Methods" on page 1598) and ***Portfolio Closed Position Properties and Methods*** (see "Portfolio Closed Positions Properties and Methods" on page 1595).

Transaction Management

Each filled order is a transaction in `Trade` object.

You can analyze a transaction by looking its properties. Transaction object is described in ***Transaction Properties and Methods*** (on page 1611).

Retrieving Transaction

Transaction can be iterated to match specific criteria.

Alternatively, to find the transaction from a closed position:

1. From the closed position, locate the transaction id from properties such as `EntryTransId`.
2. Use the property `Transaction.TransactionById` to locate the transaction record.

Sample Usage

One of the ways to utilize the transaction information is to calculate customized statistics of the transactions. You can do this in the last bar calculation as follows,

```
' in VBScript
… other code …
c = Trade.TransactionCount - 1
for i = 0 to c
   t = trade.transactions (i)
   if t.symbol = "IBM" then
      … your code here …
   end if
next
```

Monitoring Current Money Status

At a specific point in time, your trading system may want to query for money related issues (such as current available cash). These queries can help the trading system make trading and money management decisions.

These properties are part of Trade object. For more information, refer to ***System Status Properties*** (on page 1608).

Performance Analysis

One way to extract the statistics from `Trade` object is to use the system performance viewer. You can launch system performance viewer directly from a chart. See ***System Performance Viewer*** (on page 951) for more information.

Alternatively, you can use `Trade` object's reporting methods to for a text-only presentation, or have the numeric result send to Excel directly. Refer to ***Reporting Methods*** (on page 1603) and ***Reporting Methods for Microsoft Excel*** (on page 1604) for more information.

Portfolio Trading System

System performance viewer handles portfolio trading system automatically. There is nothing you need to set to make it work.

For portfolio reporting in a text-only format, refer to ***Portfolio Reporting Methods*** (on page 1600).

Portfolio Trading Systems

A portfolio trading system works on all data series available in the chart in addition to the primary linked series, i.e. the portfolio trading system can place orders for anything that is accessible by the `ItSelf.DataSeries` object.

The trading system still need a primary link for a portfolio trading system, the primary link acts as time frame reference to the trading system.

A tutorial is available that shows how to construct a portfolio trading system step-by-step. See ***Tutorial: Writing a Portfolio Trading System*** (on page 1413).

What Differentiates a Portfolio Trading System from a Single Symbol Trading System

The trading system script itself is the same, but a portfolio system will use portfolio ordering methods to make trades for specific data series.

The single symbol methods still works. Thus it is easy to extend a single symbol system to a portfolio system. You simple call the portfolio methods for data series other than the primary link.

Settings

Settings such as margin requirement and multiples for a portfolio trading system takes the single symbol version parameters by default. You have the ability to override these settings for specific instruments. For example, you can set different margin requirements for different symbols in the portfolio trading system.

For setting information, refer to ***Portfolio System Setting Properties*** (on page 1601).

Order Placement

For order placement information, refer to ***Portfolio Order Placement Methods*** (on page 1598).

Position Management

For position management information, refer to ***Portfolio Open Position Management*** (see "Portfolio Open Position Management Properties and Methods" on page 1598) and ***Portfolio Closed Position Properties and Methods*** (see "Portfolio Closed Positions Properties and Methods" on page 1595).

Reporting

System performance viewer handles portfolio trading system automatically. There is nothing you need to set to make it work.

For portfolio reporting in a text-only format, refer to ***Portfolio Reporting Methods*** (on page 1600).

Real-time Deployment

Monitoring using System Monitor

If you are going to monitor the trading system in real-time, you can use the System Monitor window (see ***System Monitor Operation Guide*** (on page 939)).

System Monitor is suitable if you manually enter your order.

You will need to set `Trade.Monitor` property to true.

If you leave this property to the default false state, the user can change the property in run-time.

Order Interface

Make sure you read ***Trading System Life Deployment Guide*** (see "Trading System Live Deployment Guide" on page 785) first.

You can send the orders to Order Interface by setting the `Trade.OrderPlacementEnabled` property to true. You will need to configure Order Interface to send the order to a real-life broker. See ***Order Interface*** (on page 735).

Once setup, orders are sent to your broker for execution.

We strongly recommend you test your trading system using Trade Simulator before actually sending the orders to a broker.

Do Not Update By Tick

Do not have **Update by Tick** turned on if you intend to execute the trades, unless your trading system is properly written to handle orders placed in a tick-by-tick basis. See ***Updated-by-Tick Trading Systems*** (on page 1512) for more information.

Updated-by-Tick Trading Systems

Do Not Update By Tick

Do not have **Update by Tick** turned on if you intend to execute the trades, unless your trading system is properly written to handle orders placed in a tick-by-tick basis.

While NeoTicker® can handle update by tick properly, your broker cannot.

The situation is worst if you have automatic order placement hooked to NeoTicker®. Having **Update by Tick** turned on can result in multiple orders being placed and and can have **serious monetary consequences**.

Even if you carry out the orders manually, when NeoTicker® decides to retract a trade, it is often not possible for you to retract it. Thus you will be carrying out the orders unfaithfully, defeating the purpose of having a trading system in the first place.

Consider the scenario: your trading system is based on price break out from a moving average line. If trading system is updated by tick, and suppose the price hovers around the moving average. If update by tick is on, NeoTicker® will issue an order when a tick comes in that is above the break out, and retracts the orders when a tick comes in that is below the break out. The final outcome will depend on the final tick for the bar.

However, when an order is issued to a broker, it will be filled when conditions are met. It is not possible for you to retract a filled order.

Properly Handle Updated-By-Tick

Last section explains why you should not updated-by-tick in most cases.

However, it is possible to update a trading system by tick. It has the advantage of placing the order before a bar is finished. This allows you to place order early if a trading decision can be made before a bar is finished. It is especially true if your trading system works on a high time frame (e.g. 5-minute).

For example, if your trading system works on 5-minute bars and will place a long order when the current high is higher than day's high. This decision can be made before the current 5-minute bar is finished. In this case, tick-by-tick update is suitable.

Don't cheat yourself. If trading decision cannot be confirmed until the bar is finished, you should not use updated-by-tick. Otherwise there is no way to consistently perform historical testing.

To properly handle update-by-tick in a trading system:

1 The trading system should have updated-by-tick enabled. This can be done by the user when adding the trading system to a chart. You can assist the user by setting updated-by-tick to true by default when you set up the indicator.

2 Update by tick restore condition should be off. This can be done by setting `Trade.UpdateByTickRestoreCondition` to false in the trading system code, or done by the user when adding the trading system to chart. This step is critical to ensure orders will not be removed automatically and it allows you to manage the orders yourself.

3 When the trading system has an entry signal, the trading system should loop through all orders that are placed in the same bar. You can use the properties described in ***Order Properties and Methods*** (on page 1589) to retrieve orders, then check the orders that have the same BarsNum as the trading system. These are the orders that have been placed in the same bar.

   Addition logic is required to determine the nature of the order. This part is trading system dependent. For example, you can use order comment to help labelling orders.

4 When checking an order, if you believe the same order has been already placed in a previous tick, you should handle the signal accordingly. Typical response is to ignore the entry signal and do not place a new order.

5 When checking an order, if you believe the same order has not been placed in a previous tick, you should handle the signal accordingly. Typical response is to place a new order, just like a non-updated-by-tick system would do.

Repeat Testing and Long Historical Testing

If you plan to run your trading system repeatedly to analyze its performance, or if you plan to system test your trading system over a very long historical period, instead of writing the trading system in script language, you should consider writing the trading system in a programming language such as Visual Basic or Borland Delphi and use IDL to interface to NeoTicker®.

Compiled languages can easily be many times faster than scripting languages and are particularity suitable such tasks.

Custom Statistics Programming Topics

You can defined custom statistics calculations on trading system performance. Custom statistics can be used to calculate statistics such as average profit loss per trade. Custom statistics is suitable to calculate statistics on historical data, and if properly written, provide real-time update on trading system statistics.

Definition

A custom statistics is an indicator that satisfies several requirements. So you can create custom statistics using scripts or IDL, and everything available to help you write scripts can be used to help you write custom statistics.

Since custom statistics is an indicator, you should have some basic knowledge about indicator programming. You can go to ***Programming Tutorials*** (on page 1367) and ***Indicator Programming Topics*** (on page 1443) to learn more about indicator programming.

A custom statistics indicator has the following requirements:

- 2 parameters set to datetime type. The first parameter specifies a from date time. The second parameter specifies a to date time. The custom statistics indicator is responsible to calculate the statistics between the specify from and to date time.

  You need to detect whether the from date time and to date time are 0. If they are 0, the custom statistics should return statistics covering all time when data is available.
- 3 output plot lines to return the statistics. First line returns combined long-side short-side statistics. Second line returns long-side statistics. Third line returns short-side statistics.

Installation

To install a custom statistics:

1. Install the custom statistics as a regular indicator.
2. In main window, choose **Manager>System Custom Statistics** to open System Custom Statistics Manager.
3. In Custom Statistics Manager, make sure **Enable Custom Statistics Calculation** is checked.
4. Choose statistics from **Available Statistics** by selecting a statistics and press the **<** button. The statistics will show up in the **In Use** list.
5. Once in-use, the custom statistics is calculated when the performance data of a trading system is collected.

Usage

For historical testing, you can view custom statistics in system performance viewer, under **Summary>Custom Statistics** page.

For real-time update, add the custom statistics as an indicator to a time chart or quote window. You should set the from/to parameter in this case to 0.

Warning: Many classes of statistics requires heavy computation, for example, standard deviation. You should avoid viewing these classes of statistics in a real-time.

Example

A custom statistics indicator is included in the installation (Stat averagePL) for calculating average profit/loss. This statistics is already installed. To put it in use, follow the installation instruction above, then run a trading system. In the performance viewer, you can view the statistics.

You can find the source code under the indicator folder in your installation (`Stat_averagePL.pas`).

Objects Reference

NeoTicker®'s programming model is designed to work with multiple development environments.

The technology that allows these different environments to work with NeoTicker® is NeoTicker®'s objects. Objects encapsulates NeoTicker®'s functionality and allow you to access features and properties through them. Objects are universal to all supported programming environments. Features in objects are available to all scripting languages as well as through the IDL interface to external programs.

Objects encapsulate the following tasks:

- Access to data
- Run time memory management
- Assistance to data analysis
- Automation
- Connectivity
- Trading system design

Notation: About Square Brackets

In this reference, we make a distinction between square brackets `[]` and round brackets `()`. Square brackets are used by languages such as Delphi Script, Delphi and Visual C++ to make distinction between array and function.

For VBScript, JavaScript and Visual Basic, array and function all use round brackets. You should read the square brackets as round brackets if you use these languages.

Notation: About String

In this reference, we use single quote for strings (e.g. `'This is a string'`). For VBScript, JavaScript, Visual Basic and Visual C++, this is equivalent to double quoting (e.g. `"This is a string"`).

Param and Params Object

Overview

`Param1` to `Param8` and `Params` objects are for handling user specified parameters.

Accessing 1-8 Parameters

To access the user parameters of the indicator, the objects `Param1` to `Param8` are provided. These are the user specified parameters in the indicator setup window.

Accessing more than 8 Parameters

To access a parameter based on numeric indexing or label indexing, the object `Params` can be used.

For example, to access the 9th parameter you can use the following call:

```
VBScript      - Params(9).int
Delphi Script - Params.items[9].int
JaveScript    - Params.items(9).Int()
```

`Params(1)` is the same as `Param1`

Accessing Parameters by Its Label

You can access a parameter by its labelling string. For example, if an integer parameter has the label **Period**, then you can access the parameter by:

```
VBScript      - Params("Period").int
Delphi Script - Params.items['Period'].int
JavaScript    - Params("Period").Int()
```

Interpreting Parameter Values

When designing an indicator, the indicator writer specifies how the parameters are shown to the user of the indicator (see the Parameter Type section in Indicator Specification (on page 1627)).

Depending on the specification, you access the parameters differently in scripts/IDL.

For string type parameters, you need to use the Str property to access value, e.g. Param1.Str.

For integer type parameters, you need to use the Int property to access value, e.g. Param1.Int.

For floating point type parameters, you need to use the Real property to access value, e.g. Param1.Real.

For date, time, and date time type parameters, you need to use the DateTime property to access value, e.g. Param1.DateTime. DateTime property is a floating point value. The integral part is number of days since 1899/12/30. The fraction part is the fraction of a 24-hour day. NTLib provides many functions that can help you decode/encode date time values.

For formula type parameters, you need to use the Str property to access value. The formula is given to you as a string and you need to launch a formula engine to evaluate the formula. For more information see ***Evaluating Formula Parameters*** (on page 1458).

For color type parameters, you need to use Color or Int property to access value, e.g. Param1.Color. The value is a standard 32-bit unsigned integer representing the ABGR values of the color. For example, the following Delphi Script shows how to access color parameter:

```
// Param1's type is set to color
if Param1.color = clRed then
    a := 1
else
    a := 0;
// a is 1 if user set param1 to red, 0 otherwise
```

Multiple Items per Parameter

Sometimes you would like to allow the user of your indicator be able to enter a comma/space separated list within one single parameter. The param objects have a number of methods to handle multiple items: `ValueCount, IntValue, RealValue, StrValue, DateTimeValue`.

The format of the input parameter is not restrictive and you can have multiple spaces between items and they will be skipped as one single separator. Only comma is used as the hard separator which for each occurrence identifies that there is a new item right after. If you need to include space in your string item, use double quote (") to enclose the item.

For example, you have set parameter 1 to be of type string and the user entered the following into the parameter entry,

```
Hello, "this is" NeoTicker 3
```

Within your indicator, you will be able to retrieve each word and expect they will return values like the following,

`param1.ValueCount` will returns 4

`param1.StrValue [0]` will return the string "Hello"

`param1.StrValue [1]` will return the string "this is"

`param1.StrValue [2]` will return the string "NeoTicker"

`param1.StrValue [3]` will return the string "3"

`param1.IntValue [4]` will return the integer 3

When combined with the use of Heap object `List` property. You can easily obtain and save list of values for use in your indicators.

Example: Exponential Moving Average

The following VBScript example illustrates the use of parameters. If you want to cut and paste the example into script editor, make sure the script header is visible under the **Visual** menu.

Consider the line:

```
Factor = 1 / Params(1).Int
```

This line calculates the smoothing factor of the exponential smoothing. Here we use the `Params(1)` which get the first parameter of the user input. Since we are expecting an integer input therefore the property of `Params(1)` is `Int`.

```
# NEOTICKER DATA BEGIN
ScriptType=Indicator
Description=VB Exponentail Moving Average
Name=vbxaverage
Language=VBScript
Links=1
```

```
MinBars=1
Multiplot_num_plots=1
Multiplot_color_0=255
Multiplot_style_0=Line
Multiplot_width_0=1
Multiplot_enabled_0=1
UpdateByTick=0
Param_count=1
Param_name_0=period
Param_inuse_0=1
Param_type_0=integer.gt.1
Explanation_Lines=1
Explanation0=Eponential Moving Average of N Period written in
VBScript.

# NEOTICKER DATA END

function vbxaverage()
dim Factor

   if not Data1.Valid (0) then
      ItSelf.Success = false
      exit function
   end if

   Factor = 1 / Params(1).Int

   if Heap.Size = 0 then
      Heap.Allocate (1)
      Heap.Value (0) = Data1.Value (0)
      vbxaverage = Data1.Value (0)
      exit function
   end if

   vbxaverage = Heap.Value (0) *  (1 - Factor) + _
                Data1.Value (0) * Factor
   Heap.Value (0) = vbxaverage
end function
```

Properties and Methods for Param Object

Int

`Int` returns the parameter as an integer.

For example, to obtain parameter 1's value as integer:

```
   VBScript      - Param1.int
   JavaScript    - Param1.Int()
   Delphi Script - Param1.int
```

Color

`Color` returns the parameter as a color. `Color` property is identical to `Int` property. Using Color makes your script/IDL easier to read.

Real

`Real` returns the parameter as a real number.

Str

`Str` returns the parameter as a string.

DateTime

`DateTime` returns the parameter as an internal date time value.

ValueCount

`ValueCount` returns the number of items identified within the parameter.

IntValue[Index]

`IntValue [Index]`returns the Index-th item in integer within the parameter. Index range from 0 to `ValueCount` - 1.

RealValue[Index]

`RealValue [Index]`returns the Index-th item in real number within the parameter. Index range from 0 to `ValueCount` - 1.

StrValue[Index]

`StrValue [Index]`returns the Index-th item in string within the parameter. Index range from 0 to `ValueCount` - 1.

DateTimeValue[Index]

`DateTimeValue [Index]`returns the Index-th item in internal date time format within the parameter. Index range from 0 to `ValueCount` - 1.

Properties and Methods for Params Object

Count

`Count` returns the number of parameters available.

Exists(Aid : variant; var Aindex : integer) : boolean

Returns true if `Aid` can be resolved to a parameter. `Aid` is either a name or an integer. If function returns true, the index of the parameter is assigned to `Aindex` so that the parameter can be accessed directly by calling `Params.items[Aindex]`.

Items

`Items` takes a variant parameter and returns a `Param` object. `Items` is the default property of Params. For usage examples, see ***Param and Params Object*** (on page 1519).

ItSelf Object

The ItSelf object encapsulates properties and methods that are focus around the indicator.

For example, ItSelf object is responsible for assigning output values, creating indicator instances that are used within the hosting indicator, setting off alerts, etc.

Properties and Methods

Answer[n]

`ItSelf.Answer` is for returning multiple values in an indicator. ItSelf.Answer takes a single parameter n, where n = {1..`ItSelf.PlotCount`}. Assigning to `ItSelf.Answer[1]` is equivalent to returning the value through the indicator function. For example,

```
ItSelf.Answer[1] := 10;
ItSelf.Answer[2] := 20;
ItSelf.Answer[3] := 30;
```

will return the value, 10, 20, 30 to plot 1, 2, 3 respectively.

Alert (Priority, Message, FGColor, BGColor, Blink)

`Alert` generates an alert event that the Alert Log can receive and display.

`Priority` is a string that you can freely assign any message to it. The standard message is one of Highest, High, Normal, Low, Lowest.

`Message` is a string with detail information about the alert. You can freely assign any string to it. For example, details of a buy / sell signal.

`FGColor` and `BGColor` are color information in integer. They are used for specifying the foreground color and the background color to be used in the Alert Log respectively.

`Blink` is a boolean for controlling whether the alert log entry will blink.

Notice that alert does not keep track of its triggering, it only create an event for the Alert Log. If you want more complex control of the alert say triggering only once per condition, then you have to control that programmatically. If you would like to play a sound file when triggering the alert, use the built-in function of tq_playsound.

CompressSeries (SeriesName, Source, TimeFrame, BarSize)

`CompressSeries` constructs higher time frame data and indicators within the indicator script. It returns a data series object that you can use just like the `Data1` object. See ***Creating Higher Time Frame Series*** (on page 1452).

CompressSeriesEx (SeriesName, UniqueID, Source, TimeFrame, BarSize)

`CompressSeriesEx` constructs higher time frame data and indicators within the indicator script. It returns a data series object that you can use just like the `Data1` object. `CompressSeriesEx` is identical to `CompressSeries` except `CompressSeriesEx` accepts a programmer provided `UniqueID` to update the compress series for performance enhancement.

See ***Creating Higher Time Frame Series*** (on page 1452).

CurrentBar[N]

`CurrentBar[N]` returns a sequential index starting from 1. When the script is first called to do calculation, usually that is the first bar right after the indicator's **Min Bars** setting, the current bar is set to 1.

CurrentDateTime : double

Returns a double value representing the current date time (in internal format) in the locale of the hosting window. If the hosting window (e.g. a Time Chart) is using the NeoTicker default locale, then this property would return the same value as `NTLib.NTNow`.

CustomBarFutureDateTime(TimePeriod, TimePeriodSize, BarsIntoFuture, CurrentTime)

`CustomBarFutureDateTime` returns a date time for a future bar for the given parameter. It is useful for projecting time for a custom data series. For example,

```
    CustomBarFutureDateTime(ppMin, 5, 10, mytime)
```

returns the date time 10 bars into future for `mytime` for a 5-min series.

DeveloperChecksum

Returns an integer value that is a checksum of a third party developer product. This property is intended for protecting an IDL indicator when a third party developer distributing the IDL indicator to his/her customers. For more information, see ***Protecting Indicators*** (on page 1665).

FirstCall

Returns true when currently the first bar of the indicator is being calculated.

See ***Initialization and Finalization*** (on page 1468).

FuncName

`FuncName` returns the function name of the indicator in use. it is a string property.

FutureDateTime (BarsIntoTheFuture)

FutureDateTime returns the date time for a future bar useful for projecting drawing objects into the future. The data time returned is based on the time frame that the indicator inherited from, which is the time frame of the primary linked series.

Host

`Host` returns the hosting function window name. It is a string property

HostType

`HostType` returns the function window type of the hosting function window. It is an integer property. It can be one of the following,

Value	Meaning
`htUnknown`	the window type is unknown
`htTimeChart`	a time chart
`htQuote`	a quote window
`htPatternScanner`	a pattern scanner window

Indicator (Name)

`Indicator` returns an indicator instance object constructed by `MakeIndicator`. ***See Using Other Indica***tors (on page 1479).

`Indicator` supports all the properties of a standard linked data series. See ***Data and LinkSeries Object*** (on page 1537).

IndicatorEx(UniqueID)

`IndicatorEx` returns an indicator instance object constructed by `MakeIndicatorEx`. See ***Using Other Indicators*** (on page 1479).

`IndicatorEx` supports all the properties of a standard linked data series. See ***Data and LinkSeries Object*** (on page 1537).

LastBarOnRecalcDoNotUpdate : boolean (read only property)

Returns the indicator setting assigned by the user.

ManagedSeries(ASeriesName)

Returns the managed series of the name `ASeriesName`. `ASeriesName` is a string.

`ManagedSeries` returns a Data object. See ***Data and LinkSeries Object*** (on page 1537) for usage of this type of object.

MinBars

`MinBars` returns the value of the minimum bar property of the indicator in use. MinBar is an integer.

MakeArray, MakeArrayEx, MakeValidArray, MakeValidArrayEx

Methods for constructing arrays. See ***Making Arrays*** (on page 1470).

MakeIndicator (SeriesName, IndicatorName, Array of Links, Array of Parameters)

`MakeIndicator` constructs indicator instance objects within the indicator script. It returns the indicator instance object. See ***Using Other Indicators*** (on page 1479).

MakeIndicatorEx (SeriesName, UniqueID, IndicatorName, Array of Links, Array of Parameters)

`MakeIndicatorEx` constructs indicator instance objects within the indicator script. It returns the indicator instance object. See ***Using Other Indicators*** (on page 1479).

`MakeIndicatorEx` is identical to `MakeIndicator`, except it takes an extra integer parameter UniqueID which can be used to reference the indicator instance object later.

Pane

If the window that hosts the indicator is a time chart, `Pane` returns pane that hosts the indicator. Otherwise, `Pane` returns 0.

This is a read-only property.

PrevVisualBreak[BarsAgo], PrevVisualBreakEx[Plot, BarsAgo]

Access the visual break settings of bars ago.

Plot[n]

`Plot` is an alias of `Answer` and does exactly the same thing. Using `Plot` or `Answer` is a personal choice.

PlotColor[n]

`PlotColor` returns the color of plot `n` set by the user. `PlotColor` is read-only.

PlotCount

`PlotCount` returns the number of plots of the current indicator. `PlotCount` is read-only.

PlotVisible[n]

`PlotVisible` returns whether the plot `n` is visible. `PlotVisible` is read-only.

RemoveCall

Returns true if the indicator is about to be removed.

This property will be true only if you have set the indicator's **Notify On Removal** option to true in indicator setup when you design the indicator.

If this flag is true, the indicator is not in a normal calculation mode. You do not need to return any bar calculation value. Instead, the indicator should perform clean up tasks (e.g. deallocate memory in IDL indicators).

See ***Initialization and Finalization*** (on page 1468).

Series (Name)

`Series` returns an embedded series constructed by `CompressSeries` or `SetSeries`. See ***Compressed Series*** (see "Creating Higher Time Frame Series" on page 1452).

SetManagedSeries(ASeriesName, TimeFrame, BarSize, Value, TickVol, Volume, PF, Valid, NewBar)

`SetManagedSeries` creates/updates a managed series. You can visualize a managed series simply as a series of bars. You need to create and update the bars manually. Managed series do not have time frame in a meaningful way, i.e. it is not aligned to the hosting indicator. The time frame information is only used when you apply an indicator on a managed series.

`ASeriesName` is a string. The name of the series to be created or updated.

`TimeFrame` is a TimeFrameType, e.g. ppMin, ppTick (see ***Time Frame Information*** (on page 1477)). `TimeFrame` is used only when the managed series is created.

`BarSize` is an integer. `Barsize` is used only when the managed series is created.

`Value` is a double. It is used to update a bar's OHLC value.

`TickVol` is an integer. It is used to update a bar's tick volume value. This value is added to the current bar's tick volume value.

Volume is an integer. It is used to update a bar's volume value. This value is added to the current bar's volume value.

PF is a boolean. It is used to mark the PF field of the bar. PF field is used in Point+Figure charts. In other types of charts, you can use the PF field to store any type of boolean value.

Valid is a boolean. It is used to mark the validity of the bar.

NewBar is a boolean. If true, a new bar will be generated for the managed series. The parameters are then used to update the new bar. If false, the parameters are used to update the current bar.

SetManagedSeries returns a Data object. See ***Data and LinkSeries Object*** (on page 1537) for usage of this type of object.

Note: Managed series is a generalized version of regular series created with SetSeries. In a regular series, each bar is a single data point only and you do not have the option to set time frame.

SetSeries(ASeriesName, ACurrentBarValue, ACurrentBarValid)

SetSeries sets value and validity to current bar of the series. If the series has not been referenced before, it will be created on demand. ASeriesName is a string. ACurrentBarValue is a numeric value. ACurrentBarValid is boolean. You can use series to store a series of values and thee result can be used in places like MakeIndicator.

Conceptually, SetSeries is equivalent to series assignment in formula.

Success

Success is a boolean property. Default is true. If you set Success to false, the current bar of the indicator will become an invalid bar.

Success only marks the first plot. For multi-plot indicators, use SuccessAll or SuccessEx.

SuccessAll

If you set SuccessAll to true, all the plots will have a success value of true (valid bar). If you set SuccessAll to false, all the plots will have a success value of false (invalid bar). When read, SuccessAll returns true if all plots have their success values set to true.

SuccessEx[n]

SuccessEx is for setting the successful state of multiple plot. SuccessEx takes a single parameter n, where n = {1..ItSelf.PlotCount}. Assigning to SuccessEx[1] is equivalent to assigning the successful state to Success. For example,

```
ItSelf.SuccessEx[1] := true;
ItSelf.SuccessEx[2] := true;
```

```
        ItSelf.SuccessEx[3] := false;
```

will set plot 1 and 2 to success, plot 3 to failure.

TimeDriven

Returns true if the hosting environment is time driven. False if the hosting environment is bar driven. This property is mainly for handling drawing tools in time chart where time/bar driven charts have distinct look.

TimeFrameType, TimeFramePeriod

Properties to query about time frame, see ***Time Frame Information*** (on page 1477).

TimerCall

If an indicator is updated by timer (see ***Indicator Updated by Timer*** (on page 1467)), this property is true when the indicator calculation is triggered by timer. False if the indicator calculation is triggered by regular indicator update.

For non-updated by timer indicators, this property is always false.

TradingTimeStart

`TradingTimeStart` returns the starting time of trading for the day. (It returns 0.375 for 9 AM of the day. It returns values ranges from 0 to less than 1.)

NTLib provides a set of date time conversion functions, see ***Date Time Functions*** (on page 1621).

TradingTimeEnd

`TradingTimeEnd` returns the ending time of trading for each day.

NTLib provides a set of date time conversion functions, see ***Date Time Functions*** (on page 1621).

TradingTime24Hours

`TradingTime24Hours` returns true if the host window is marked with 24 hours for the trading hours.

UniqueID

Returns an unique integer ID for the indicator. Each instance of an indicator will have an unique id within a NeoTicker® session.

See ***Initialization and Finalization*** (on page 1468).

UpdateByTick

`UpdateByTick` returns the Boolean value of the update by tick setting of the indicator in use.

UpdateCompressSeriesEx(UniqueID)

Returns a compress series object. If the compress series has not been created yet, a nil pointer is returned.

`UpdateCompressEx` optimizes compress series performance by reducing the number of calls to `CompressSeriesEx. See Creating Higher Time Frame Series (on page 1452).`

UpdateIndicator(IndicatorName)

Returns an indicator instance object. If the indicator instance has not been created yet, a nil pointer is returned.

Used in the context of calling other indicators, `UpdateIndicator` optimizes indicator performance by reducing the number of calls to `MakeIndicator`. See ***Using Other Indicators*** (on page 1479).

UpdateIndicatorEx(UniqueID)

Returns an indicator instance object. If the indicator instance has not been created yet, a nil pointer is returned.

Used in the context of calling other indicators, `UpdateIndicatorEx` optimizes indicator performance by reducing the number of calls to `MakeIndicatorEx`. See ***Using Other Indicators*** (on page 1479).

UpdateOnPrimaryLinkOnly : boolean (read only property)

Returns the indicator setting that the user assigned.

Value[N]

`Value` returns the indicator value of N bars ago. N must be an integer >= 0

For example, to access the previous value of the indicator:

```
VBScript      - ItSelf.Value(1)
JavaScript    - ItSelf.Value(1)
Delphi Script - ItSelf.Value[1]
```

If `Value[N]` is an invalid bar, the property locates the previous value that is valid and returns that number instead. Most users will find this behavior acceptable. For indicators that depend on the validity of data, you can check Valid[N] before using the data point.

ValueEx[n, b]

ValueEx returns the value of plot n of b bars ago, where n = {1..ItSelf.PlotCount} and b is number of bars ago. Reading `ValueEx[1, b]` is equivalent to reading `ItSelf.Value[1]`. For example,

```
ItSelf.ValueEx[2, 1]
```

returns the value of plot 2 of the last bar. `ItSelf.ValueEx` is read-only.

Valid[N]

`Valid` returns true or false depending on whether the indicator value at N bars ago is valid or not

ValidEx[n, b]

`ValidEx` returns the validity of plot n of b bars ago, where n = {1..ItSelf.PlotCount} and b is number of bars ago. Reading `IValidEx[1, b]` is equivalent to reading `Valid[1]`. For example,

```
ItSelf.ValidEx[2, 1]
```

returns the validity of plot 2 of the last bar. `ValidEx` is read-only

Visible

`Visble` returns the status of the visible property of the indicator in use. `Visible` is a Boolean value.

Visible is useful for indicators that generate not just values but also drawing tools. You can choose not to create drawing tools if indicator is not visible to reduce system resource usage and improve calculation speed.

VisualBreak

`VisualBreak` is a boolean property you assign to. This property controls the visual breaks of first visible plot.

Visual break is applicable to time charts only. The indicator plot must be set to either line or square style, with the break property set to **Visual**. See ***Broken Lines*** (on page 1123) for information on how to set up in a chart.

Setting `VisualBreak` to true (1) means there is a break and the indicator will stop plotting at the point. Setting `VIsualBreak` to false (0) means there is no break and the indicator will continue plotting.

VisualBreakEx[n]

`VisualBreakEx` is a boolean property you assign to. This property controls the visual breaks of the specified plot. N ranges from 1 to plot count.

Visual break is applicable to time charts only. The indicator plot must be set to either line or square style, with the break property set to **Visual**. See ***Broken Lines*** (on page 1123) for information on how to set up in a chart.

Setting `VisualBreak` to true (1) means there is a break and the indicator will stop plotting at the point. Setting `VIsualBreak` to false (0) means there is no break and the indicator will continue plotting.

Data and LinkSeries Object

For First 2 Linked Data

To access the first two linked data, you can use the objects `Data1` and `Data2`.

`Data1` and `Data2` are objects referring to the first and second linked data sources for the indicator (Link 1 and Link 2).

For example,

```
   Data1.Value(0)
```

returns the current value of the first link (VBScript syntax).

For Arbitrary Linked Data

Use `LinkSeries[n]` to access the linked data, where n = 1 .. `LinkSeries.Count`. LinkSeries[1] is equivalent to Data1.

For example,

```
   LinkSeries(3).Value(0)
```

returns the current value of the third link (VBScript syntax).

Properties and Methods

BarsNum[N]

`BarsNum` returns the bar index since the first loaded bar. N is the number of bars ago.

Close[N]

Close value of N bars ago.

Date[N]

`Date` returns date in internal real number format, thus is the same as Windows date time value, i.e. 1 day = 1.0

DateTime [N]

DateTime returns the combined date time value in double of the price bar specified N periods ago

Day[N]

Day of the month. Ranges from 1 to 31.

DayOfWeek[N]

Day of Week value. Ranges from 0 to 6. 0 = Sunday, 1 = Monday, etc.

High[N]

High value.

Hour[N]

`Hour` returns the hour of the bar.

IsData : boolean

Returns true or false value based on the nature of the data source

IsIndicator : boolean

Returns true or false value based on the nature of the data source

IsLastBar

`IsLastBar` returns true or false value based on the position of the current bar.

LastBarsnum

Returns an integer of the barsnum of the last bar in series.

`LastBarsnum` is for accessing the bar position of the last bar in historical calculation. It is mainly intended for indicators that draw. Note that information about last bar price is deliberately left out to prevent future peeking. For example, `Data1.LastBarsnum` returns an integer of the barsnum of the last bar in the data series.

LastDateTime

Returns a double of date time of the last bar in series. The value returned is in Windows date time format.

`LastDateTime` is for accessing the date time of the last bar in historical calculation. It is mainly intended for indicators that draw. Note that information about last bar price is deliberately left out to prevent future peeking. For example, `Data1.LastDateTime` returns a double of the date time of the last bar in the data series.

Low[N]

Low value.

MakeArray, MakeArrayEx, MakeValidArray, MakeValidArrayEx

Methods for making arrays. See ***Making Arrays*** (on page 1470).

Minute[N]

`Minute` returns the minute of the bar.

Month[N]

Month. Ranges from 1 to 12.

Open[N]

Open value.

OpenInt[N]

Open Interest value.

PF[Index]

Returns 1 when a Point and Figure bar is up. Returns -1 when a Point and Figure bar is down. Same call also works with 3LB, Kagi and Superposition bars.

Tick[N]

Tick count value.

Time[N]

`Time` returns the time of the bar in internal time format.

TimeFrameType, TimeFramePeriod

Properties to query about time frame, see ***Time Frame Information*** (on page 1477).

Second[N]

`Second` returns the second of the bar.

Symbol

`Symbol` returns the symbol of the data series.

Value[N]

`Value` of the linked item. If the link is a data series, `Value` maps to one of open, high, low, close, volume or OpenInt. The mapping is defined when the user specify the link in the Indicator editor. If the link is an indicator, Value maps to the linked indicator's value.

For example, to access the current value of Link 1:

```
VBScript      - Data1.Value(0)
JavaScript    - Data1.Value(0)
Delphi Script - Data1.Value[0]
```

ValueEx[N, b]

`ValueEx` is useful only if the linked series is a multi-plot indicator. It returns the value of plot N of b bars ago, where n = { 1.. Datax.PlotCount} and b is number of bars ago. Reading `ValueEx[1, b]` is equivalent to reading `Value[b]`. For example:

```
Data1.ValueEx[2,1]
```

returns the value of the Data1, plot 2 of the last bar.

And

```
Data2.ValueEx[3,1]
```

returns the value of the Data2, plot 3 of the last bar.

Valid[N]

Validity of the bar.

ValidEx[N]

`ValueEx` is useful only if the linked series is a multi-plot indicThe validity of plot N of b bars ago, where n = { 1..Datax.PlotCount} and b is number of bars ago. Reading `ValidEx[1,b]` is equivalent to reading `Valid[b]`. For example:

```
Data1.ValidEx[2,1]
```

returns the validity of data series Data1, plot 2 of the last bar.

And

```
Data2.ValidEx[3,1]
```

returns the validity of data series Data2, plot 3 of the last bar.

Volume[N]

Volume value.

Year[N]

`Year` returns a 4-digit year (YYYY).

Properties and Methods Specific to LinkSeries

Count

`Count` returns the number of link series.

Items[N]

The Nth link series. The link series that you can access range from 1 to `Count`.

`Items` is the default property of `LinkSeries`. For VBScript and JavaScript users, you can directly index to `LinkSeries` without explicitly specifying the property Items.

Example on accessing the 2nd data series in the time chart,

```
Delphi Script - LinkSeries.items[2].Value[0]
VBScript      - LinkSeries(2).Value(0)
```

The properties of an individual data series is exactly the same as Data1 and Data2.

Properties and Methods for Real-Time Bid/Ask/Last Trade Data

You can use properties listed here to access latest bid/ask/last trade information, regardless of the underlying series' time frame.

To prevent accidental future peeking, bid/ask/trade data are restricted. You can only access them in real-time or when the user is doing a tick replay (see ***Replay by Tick*** (on page 1123).). They are not available when recalculating the indicator using historical data. You can use the corresponding 'has' function to determine the availability of data.

An equivalent set of fields are available in formula. See ***Accessing Data Links*** (see "Accessing Data Links and Fields" on page 278).

AskPrice

Returns the latest ask price. Use `HasAsk` to determine if this property is available.

AskSize

Returns the latest ask size. Use `HasAsk` to determine if this property is available.

BidPrice

Returns the latest bid price. Use `HasBid` to determine if this property is available.

BidSize

Returns the latest bid size. Use `HasBid` to determine if this property is available.

HasAsk

Returns true if latest ask information is available.

HasBid

Returns true if latest bid information is available.

HasLastTrade

Returns true if last trade information is available.

LastTradeDateTime

Returns last trade date/time information in internal format. Use `HasLastTrade` to determine if this property is available.

LastTradePrice

Returns last trade price. Use `HasLastTrade` to determine if this property is available.

LastTradeSize

Returns last trade size. Use `HasLastTrade` to determine if this property is available.

SideAsk

This property complements `UpAsk`. Query this property to see if the latest ask is actually a side ask. Use `HasAsk` to determine this property is available.

SideBid

This property complements `UpBid`. Query this property to see if the latest bid is actually a side bid. Use `HasBid` to determine this property is available.

UpAsk

Returns true when the latest ask is an up ask. Use `HasAsk` to determine this property is available.

For side ask, it retains the previous up ask value. You can use `SideAsk` to determine if the ask is actually a side ask.

UpBid

Returns true when the latest bid is an up bid. Use `HasBid` to determine this property is available.

For side bid, it retains the previous up bid value. You can use `SideBid` to determine if the bid is actually a side bid.

Properties and Methods for Tick Statistics (Data Only)

Properties listed here are tick level statistics that are collected on a tick-by-tick basis. In another words, unless the chart has been running in real-time for a long time, you will need to perform a tick replay to get an accurate value. For more information on tick replay, see ***Replay by Tick*** (on page 1123).

All the properties with names started with the prefix "Day" are reset on every trading day.

An equivalent set of fields are available in formula. See ***Accessing Data Links*** (see "Accessing Data Links and Fields" on page 278).

Property	Type	Meaning
DayCancelledBidVolume	integer	This field accumulates the cancelled bid size at the highes
DayCancelledAskVolume	integer	This field accumulates the cancelled ask size at the lowest
DayVWAP	double	Volume Weighted Average Price.
DayTWAP	double	Tick Weighted Average Price.
DayTotalTicks	integer	Total number of ticks.
DayTotalUpTicks	integer	Total number of up ticks.
DayTotalDownTicks	integer	Total number of down ticks.
DayTotalUpSideTicks	integer	Total number of side ticks (up side ticks only).
DayTotalDownSideTicks	integer	Total number of side ticks (down side ticks only).
DayTotalBidTrades	integer	Total number of trades at bid price.
DayTotalAskTrades	integer	Total number of trades at ask price.
DayTotalVolume	integer	Total volume.
DayTotalUpVolume	integer	Total volume from up ticks.
DayTotalDownVolume	integer	Total volume from down ticks.
DayTotalUpSideVolume	integer	Total volume from up side ticks.
DayTotalDownSideVolume	integer	Total volume from down side ticks.
DayTotalBidTradeVolume	integer	Total volume on trades at bid price.

`DayTotalAskTradeVolume`	integer	Total volume on trades at ask price.
`DayTick16`	integer	Statistics on the last 16 ticks. Each up tick counts as 1. Ea down tick counts as -1. Side tick assumes the value from previous tick.
`DayBATick16`	integer	Statistics on the last 16 ticks. Each trade at ask counts as 1 trade at bid counts as -1. Side tick assumes the value from previous tick.

DataSeries Object

To direct access all data series within a function window, like the time chart, `DataSeries` object can be used. It does not require linkage as in the case of `Data1`, `Data2` and `LinkSeries` objects.

Properties and Methods

Count

`Count` returns the number of data series exists within the current context.

GetIndexBySymbol(Symbol)

Returns the integer index of the first data series that has the symbol that matches.

Items[N]

The Nth data series. The data series that you can access range from 1 to `DataSeries.Count`

`Items` is the default property of `DataSeries`. For VBScript and JavaScript users, you can directly index to `DataSeries` without explicitly specifying the property Items.

Example on accessing the 2nd data series in the time chart,

```
Delphi Script - DataSeries.items[2].Value[0]
VBScript      - DataSeries(2).Value(0)
```

The properties of an individual data series is exactly the same as Data1 and Data2. ***See Data and LinkSeries Ob***ject (on page 1537).

Heap, PHeap and GHeap Objects

Global variables are unreliable to store values between calculations, especially when you have multiple charts and indicators. Heap objects provides a way to save real values and keeps those values intact from one price bar to next between indicator calculations.

NeoTicker® provides three types of heap objects: `Heap`, `PHeap` and `GHeap`.

When a tick comes into an indicator, `Heap` restores itself to the initial state of the bar the tick belongs to before the indicator script is executed. This ensures a consistent value when the indicator is calculated from ticks and when the indicator is calculated from historical minute bars.

`PHeap` does not restores itself to the initial state of the bar. Thus storage in `PHeap` is persistent from tick to tick. `PHeap` is useful when the indicator's correctness depends on this persistency. The drawback is it is not possible to ensure indicator value calculated from ticks will agree with indicator value calculated from historical minute bars.

For most usage, `Heap` is a better choice than `PHeap` because of the consistency `Heap` ensures. Use `PHeap` only if you need the tick to tick persistency it offers. Methods and properties of `Heap` and `PHeap` are identical.

`GHeap` is persistent across all indicators, charts, etc. `GHeap` is useful when you need to pass values, strings and other information back and forth among multiple indicators and function windows. You have to use this heap with caution because it is a shared resource and every indicator that is in use can read and write into the same area of the heap. Like `PHeap`, `GHeap` does not restore itself to the initial state of any indicators.

All three types of heap objects share the same class properties and methods. Once you have an understanding using the basic Heap, you will learn to use the other ones quickly.

Accessing Items by Index

If you want to access heap items by index, you need to allocate a storage, e.g.

```
Heap.Allocate(10)
```

will allocate 10 slots for accessing. Heap index starts from 0, so the above call will make slot 0 to slot 9 available for use. Later, a value can be written to and read from a slot, e.g. the following Delphi Script call:

```
Heap.Value[5] := 10;
```

will assign the value 10 to the slot index 5.

Addition features are available to help you sort, query, search heap objects.

A limit of 100,000 slots in the heap is imposed.

Accessing Items by Label

Heap objects items can be accessed by labels. The storage is separated from heap slots that are accessed by index. So you can mix the two type of usages together.

Single slot items do not need explicit storage declaration. When you reference the item by its label, storage is automatically allocated for you.

List/table items occupy multiple slots. You need to allocate storage by calling `AllocateRealList`, `AllocateIntList`, `AllocateRealTable` or `AllocateIntTable` e.g.

```
Heap.AllocateRealList('myarray', 1000)
```

allocates a 1000 slot list that can be accessed by the name `myarray`. The index starts at 0.

The name you choose for the label has a 255 character limit in length.

If a list/table is allocated in heap or pheap, when the indicator is deleted, the list/table is automatically dispose. You do not need to worry about memory management issues.

However, if a list/table is allocated in gheap, the list/table is not automatically dispose with indicator deletion. In this case, you will need to use the `DisposeRealList, DisposeIntList, DisposeRealTable or DisposeIntTable` call in the indicator's finalization to free memory. See ***Initialization and Finalization*** (on page 1468).

Properties and Methods for Index Referencing

Allocate(N)

`Allocate` requests for N number of slots for storing real values. N will become the current usable size of the heap. Data saved in the Heap will not be affected when you enlarge the size of the heap. You cannot reduce the size of the heap during runtime. So if you request a number smaller than an allocated size, heap object will ignore the request.

Average(Start, End)

`Average` returns the average of all values within index range of `Start` to `End.`

Dec(N)

`Dec` decreases `Value[N]` by 1, and returns the decreased value.

Fill(From, To, AValue)

This method fills `Value[From]`, `Value[From+1]`, ... , `Value[To]` with `AValue`.

Inc(N)

`Inc` increases `Value[N]` by 1, and returns the increased value.

IncMod(N, M)

IncMod increases `Value[N]` by 1, modulus the value by `M`, and returns the result value, i.e.

```
Value[N] := (Value[N] + 1) mod M.
```

Max(Start, End)

`Max` returns the maximum value within index range of `Start` to `End.`

MaxPos(Start, End, From)

`MaxPos` returns the slot number of the maximum value within index range of `Start` to `End`, using `From` position to initiate the search

Min(Start, End)

`Min` returns the minimum value within index range of `Start` to `End.`

MinPos(Start, End, From)

`MinPos` returns the slot number of the minimum value within index range of `Start` to `End`, using `From` position to initiate the search.

Size

The number of slots available currently.

Sort(Start, End)

This method sorts the values in the heap in ascending order, from index `Start` to index `End.`

Sum(Start, End)

`Sum` returns the sum of all values within index range of `Start` to `End.`

Value[N]

You can assign and retrieve values directly through this property. `N` can ranges from 0 to `Size` -1

An example Delphi Script using the heap object follows. This script returns the high of the previous trading day.

```
function prevDHigh : double;
begin

   //If the current bar is the first bar, then store
   //the high value as the previous high in the heap
   if data1.barsnum[0] = 1 then
   begin
      heap.allocate (1);
      heap.value[0] := data1.high[0];
   end;

   //If the day has changed, then returns the previous high
   //and scrap the old high.  Otherwise, checks if the
   //bar makes a new high for the day
   if data1.day[0] <> data1.day[1] then
   begin
      result := heap.value [0];
      heap.value [0] := data1.high[0];
   end
   else
   begin
      if heap.value [0] < data1.high[0] then
         heap.value [0] := data1.high[0];
      result := itself.value[1];
   end;
end;
```

Properties and Methods for Label Referencing (Single Slot)

Real['name']

`Real` refers to a real number, indexed by the string `name`. It is a read and write property.

Int['name']

`Int` refers to an integer variable, indexed by `name`. It is a read and write property.

Str['mystr']

`Str` refers to a string variable, indexed by `mystr`. It can store up to 255 characters. It is a read and write property.

Properties and Methods for Label Referencing (List)

AllocateIntList('myarray', N)

ReallocateIntList('myarray', N)

`ReallocateIntList` allocates an array called `myarray` with `N` integer number elements. The array is 0-based. `ReallocateIntList` will resize list if necessary.

`AllocateIntList` is identical to `ReallocateIntList` except it lacks the ability to resize. It is for backward compatibility only.

AllocateRealList('myarray', N)

ReallocateRealList('myarray', N)

`ReallocateRealList` allocates an array called `myarray` with N real number elements. The array is 0-based. `ReallocateRealList` will resize list if necessary.

`AllocateRealList` is identical to `ReallocateRealList` except it lacks the ability to resize. It is for backward compatibility only.

DisposeIntList('myarray')

Free the memory allocated by the array called `myarray`.

DisposeRealList('myarray')

Free the memory allocated by the array called `myarray`.

IntList['myarray', N]

`IntList` accesses the Nth element in a integer array named `myarray`. `N` ranges from 0 to list size - 1. It is a read and write property.

RealList['myarray', N]

`RealList` accesses the Nth element in a real number array named `myarray`. `N` ranges from 0 to list size - 1. It is a read and write property.

ListCReplace('listname', N, V)

`ListCReplace` inserts the value V to the Nth element and returns the next index item for next replacement call. This is a circular replace, if the max number of the list is hit, index 0 will be returned.

ListClear('listname')

`ListClear` clear the list with zeroes or null string.

ListCopy('source', 'target')

`ListCopy` copies the content of the list source to the list target. Both list must be allocated and have the same number of items.

ListCount('listname')

ListCount returns the number of items in the list.

ListMax('listname')

ListMax returns the maximum value in the list.

ListMaxBar('listname')

ListMaxBar returns the slot number of the maximum value in the list.

ListMin('listname')

ListMin returns the minimum value in the list.

ListMinBar('listname')

ListMinBar returns the slot number of the minimum value in the list.

ListPop('listname')

ListPop returns element 0 from the array, removes item 0, and moves the rest of the list up. ListPop does not change the size of the list. The last slot will be set to 0.

ListPush('listname', N)

ListPush pushes the value N to element 0 and move the rest of the list to the next higher slot. ListPush does not change the size of the list. The last slot will be dropped off from the list.

Listsum('listname')

Listsum returns the sum of all items in the list.

RealListLinearRegression('listname', slope, constant)

RealListLinearRegression applies linear regression on the list. Slope and constant resulted from the linear regression is stored in the real number variable slope and constant.

RealListNormalize ('listname', MinValue, MaxValue)

Normalize (linearly rescale) all the data stored in the list listname to range from MinValue to MaxValue.

RealListParallelSort('listname1', 'listname2')

RealListParallelSort sorts listname1 in increasing order. After the sort, listname2 will be arranged such that the order will match the sorting order of listname1.

For example, if listname1 is 0.5, 0.7, 0.3 and listname2 is 1, 2, 3. Then after calling RealListParallelSort, listname1 will become 0.3, 0.5, 0.7 and listname2 will become 3, 1, 2. Note that the items in listname1 and listname2 match before and after the sort.

Properties and Methods for Label Referencing (Table)

AllocateIntTable('mytable', Row, Col)

ReallocateIntTable('mytable', Row, Col)

`ReallocateIntTable` allocates a 2-dimensional array called `mytable` with `Row` x `Col` integer elements. The table is 0-based. It resizes table if necessary.

`AllocateIntTable` is identical to `ReallocateIntTable` except it lacks the ability to resize. It is for backward compatibility only.

AllocateRealTable('mytable', Row, Col)

ReallocateRealTable('mytable', Row, Col)

`ReallocateRealTable` allocates a 2-dimensional array called `mytable` with `Row` x `Col` real number elements. The table is 0-based. It resizes table if necessary.

`AllocateRealTable` is identical to `ReallocateRealTable` except it lacks the ability to resize. It is for backward compatibility only.

DisposeIntTable('mytable')

Free the memory allocated by the table called `mytable`.

DisposeRealTable('mytable')

Free the memory allocated by the table called `mytable`.

IntTable['mytable', Row, Col]

IntTable access the a table element. `Row` ranges from 0 to number of rows - 1. `Col` ranges from 0 to number of columns - 1. It is a read and write property.

RealTable['mytable', Row, Col]

RealTable access the a table element. `Row` ranges from 0 to number of rows - 1. `Col` ranges from 0 to number of columns - 1. It is a read and write property.

Properties and Methods for Price Profiling

Heap object can be used to perform price profiling calculation, similar to that of the Volume Profile indicator (see ***Power Indicators Guide, Volume Profile*** (see "Volume Profile" on page 1355)). Performing price profiling in heap makes the profile values available for further calculation. This makes it possible to design trading signals and systems that are based on price profiling.

The typical workflow of an indicator that uses price profile involves:

1 Allocating price profile in heap using the `AllocatePriceProfile` call.

2 Add new prices to profile by calling `PriceProfileAddPrice` or `PriceProfileAddPriceRange`.

3 Use `PriceProfileSlotCount` and `PriceProfileSlotInfo` to get individual pictures of the price slots in the profile.

4 Use one of the statistics calls like `PriceProfileTotalVolume` to analyze the overall profile.

5 (Optional) use `DisposePriceProfile` to remove profiles that are no longer needed.

For example, you can duplicate the calculation of Volume Profile indicator on a minute data series by using:

1 `AllocatePriceProfile` to create a profile.

2 For each minute bar, use `ProfileProfileAddPriceRange` to profile the bar.

3 For each slot, use `PriceProfileSlotInfo` to get the volume information of the slot.

What are Slots

A slot aggregate the volume at a specific price range called slot size (e.g. $0.10). Each profile has multiple slots of the same slot size. Slot size is specified when you allocate the profile. For example, if you profile a stock priced around $10 using $0.10 slot size, the profile may end up with slots at $9.75, $9.85, $9.95, $10.05, $10.15, etc.

Note that the number of slots are increased dynamically as you add more prices to the profile. `PriceProfileSlotCount` tells you how many slots are currently in the profile.

When to Dispose a Profile

Profiles are automatically disposed when an indicator is removed from NeoTicker®. So you do not have to dispose a profile manually unless:

- The profile is used only temporarily in the calculation, but is no longer needed. You may want to dispose it manually to save memory, or,
- The profile is used in GHeap and you want to account for its memory usage.

Profiling Tick Data

If you want to profile tick data, you should:

- Use `PHeap` instead of `Heap`, so that profile data is not reset as each tick arrives.
- Make sure your indicator is updated by tick.
- Use `PriceProfileAddPrice` to add price, volume data to the profile.

The following are the price profile properties and methods.

AllocatePriceProfile(const name : String; slotsize, fromprice, toprice : double)

Allocate a price profile of `name` with the specified `slotsize`. `Fromprice` and `toprice` are a suggested price range the profile will be working on. They are also used to align the slot prices. Avoid specifying too wide of a range to conserve memory.

ClearPriceProfile(const name : String; fromprice, toprice)

Reset the `name` profile with a new starting range specified by `fromprice` and `toprice`.

DisposePriceProfile(const name : String)

Dispose the `name` profile by de-allocating the memory.

PriceProfileAddPrice(const name : String; price, volume)

Add a `price, volume` pair to the profile `name`. Use this call if you are processing tick level data where exact price/volume of each trade is known. This call is also useful for indicator profile analysis.

PriceProfileAddPriceRange(const name : String; fromprice, toprice, volume)

Add a price range of `fromprice, toprice and volume` pair to the profile `name`. Use this call if you are processing bar level data where you only have a price range (i.e. high/low price) to work on, but not the price of individual trades.

Volume is divided evenly among the slots in the specified price range, e.g. for a volume of 1000 and 10 slot, each slot will have volume increased by 100.

PriceProfileAddPriceRangeFixedVolume(const name : String; fromprice, toprice, volume)

Similar to `PriceProfileAddPriceRange`, but volume is not divided. Each slot receive the specified volume, e.g. for a volume of 1000 and 10 slot, each slot will have volume increased by 1000.

PriceProfileRemovePrice(const name : String; price, volume)

Remove a `price, volume` pair from the profile `name`. Use this call if you are processing tick level data where exact price/volume of each trade is known.

PriceProfileRemovePriceRange(const name : String; fromprice, toprice, volume)

Remove a price range of `fromprice, toprice and volume` pair to the profile `name`. Use this call if you are processing bar level data where you only have a price range (i.e. high/low price) to work on, but not the price of individual trades.

Volume is divided evenly among the slots in the specified price range, e.g. for a volume of 1000 and 10 slot, each slot will have volume reduced by 100.

PriceProfileRemovePriceRangeFixedVolume(const name : String; fromprice, toprice, volume)

Similar to `PriceProfileRemovePriceRange`, but volume is not divided. Each slot receive the specified volume, e.g. for a volume of 1000 and 10 slot, each slot will have volume decreased by 1000.

PriceProfileMax(const name : String) : double

The maximum price of the profile `name`.

PriceProfileMin(const name : String) : double

The minimum price of the profile `name`.

PriceProfileVWAP(const name : String) : double

Volume weighted average price of the profile `name`.

PriceProfileTotalVolume(const name : String) : double

Total volume of the profile `name`.

PriceProfileMedian(const name : String) : double

The median price of the profile `name`.

PriceProfileMode(const name : String) : double

The mode price of the profile `name`.

PriceProfileStdDev(const name : String) : double

The standard deviation of price of the profile `name`.

PriceProfileSlotCount(const name : String) : integer

Number of slots currently in the profile `name`.

PriceProfileSlotInfo(const name : String; slot : integer; var slotmidprice : double; var slotvolume : double) : boolean

Information for `slot` in profile `name`. `Slot` ranges from 0 to `PriceProfileSlotCount` - 1. Mid price and volume of the slot is returned.

This call returns true if the profile name and slot number are valid.

Report Object

If you need special reporting from within a chart, you can use the `Report` object to output text messages to a report window.

When you use methods in `Report` object, you need to provide the name of the report window the messages are sent to. If you do not provide a name, i.e. leave the report window name blank, `Report` object will report to the first report window in your active group.

For example, the following statements print out the message 'Hello World' to the first report window of the active group:

```
VBScript       - Report.AddLine "","Hello World"
JavaScript     - Report.AddLine("","Hello World");
Delphi Script  - Report.AddLine('','Hello World');
```

Properties and Methods

AddLine (WindowName, Mesg)

`AddLine` sends a text message `Mesg` to the report function window named `WindowName`. If `WindowName` is empty, the first report window in the current active group will be used. If the specified report window is not available, (e.g. not opened) the message will not be sent.

AddValue (WindowName, Variable, Label)

AddValue sends `Variable` (can be integer, real, string etc.) to the report window `WindowName` and formats the variable based on its type. `Label` is an optional label for the output. Basically it is a script debugging method.

Clear (WindowName)

`Clear` erases all content from the report window

LineCount (Window)

`LineCount` returns the number of lines used in the specified report window.

OpenNew(Window)

Open a report window. If the report window is already open, this call does nothing.

Ready (WindowName) : boolean

Returns true if the report window is in the current session and can be written to.

Save (WindowName)

`Save` saves the content of the report window `WindowName` to a file. If a file name is not previously specified in the target report window, no file will be saved.

SaveTo (Window, Filename)

`SaveTo` saves the specified report window to the target file. If the file exists, the operation will do nothing.

Excel Object

`Excel` object controls Microsoft Excel. It is a faster and easier way than automating Excel yourself through Excel's ActiveX interface.

Methods and Properteis

CloseWorkbook(ABookname : string; ASaveChanges : boolean)

Closes a workbook in Excel. `ABookname` is the bookname. If `ASaveChanges` is true, the book is saved before close.

OpenWorkbook(AFileName : string)

Opens a workbook in Excel. The workbook is in the file `AFileName`.

SetArray(WorkBookName, WorkSheetName, Col, Row, [s1, ..., , sn], AutoCreate)

This method sends an array of series (s1..sn) to the Excel workbook and worksheet specified. The array is exported to the cells starting at `Col` and `Row`, with each series occupying a column. Each series of s1..sn is created by the array making methods (see ***Making Arrays*** (on page 1470)).

If `AutoCreate` is true, `SetArray` will create a workbook and worksheet if they are not present in Excel.

SetValue(WorkBookName, WorkSheetName, Col, Row, Value, AutoCreate)

This method sends a value to the Excel workbook and worksheet specified. The value is exported to the cell at `Col` and `Row`.

If `AutoCreate` is true, `SetValue` will create a workbook and worksheet if they are not present in Excel.

ChartDrawingObjects and DrawingObjects Objects

You can create indicator driven drawing tools using the object `ChartDrawingObjects` and `DrawingObjects`. You can create, modify, and delete the drawing objects during runtime to create any visual effects you are interested in doing. You can even combine indicator plotting with drawing tools to improve the visualization impact.

`ChartDrawingObjects` and `DrawingObjects` have access to all internal drawing tools and user defined custom drawing tools. `ChartDrawingObjects` and `DrawingObjects` are essentially the same type of object and share the same methods and properties.

ChartDrawingObjects vs. DrawingObjects

The drawing tools created in `DrawingObjects` are not user adjustable and they are available to the indicator that creates them only. When you delete an indicator from a time chart, all these drawing tools will be removed at the same time.

If you choose to use the same indicator in pattern scanner or the quote window, you can design your indicator to be able to distinguish which window it is being hosted by, using `ItSelf.Host` and `ItSelf.HostType`, then choose to creating the drawing objects only if your script is running within a chart. So you will not create drawing tools unnecessarily in quote window and pattern scanner.

For example, if you want the script to automatically use trend lines to mark a chart, use `DrawingObjects`. The script will prevent users from accidentally moving the trend lines. Once the script is deleted, the trend lines will be removed.

The drawing tools created in `ChartDrawingObjects` are user adjustable and they are visible to all indicators, i.e. they behave like drawing tools a user drawn on a chart.

For example, if you want the script to help user to create a trend line, and ultimately the user will drag and move the trend line, use `ChartDrawingObjects.` Once the trend line is created, user has full control of the trend line. Even if the user delete the indicator, the trend line will remain on the chart.

Creating and Managing Drawing Tool

To create a drawing tool, use the Add method. For example,

```
Id = DrawingObjects.Add(cotHorizontalLine)
```

will create a horizontal line drawing tool. The identifier of the drawing tool is returned. You should save the id somewhere for later reference.

To delete a drawing tool, use the `Delete` method. For example,

```
Delete(Id)
```

will delete a drawing tool.

Drawing tools are automatically removed when the indicator is deleted.

Drawing Tool Types

When you create a drawing tool, the types are as follows:

```
cotHorizontalLine
cotVerticalLine
cotTrendline
cotText
cotRectangle
cotEllipse
cotAndrew
cotArrowLine
cotTrendChannel
cotSupportResistance
cotFan
cotGannFan
cotFibLevel
cotFibArc
cotFibTime
cotMarker
cotUpArrow
cotDownArrow
cot3PointFibLevel
cot3PointFibTime
cot3PointProjection
cotCustom1
cotCustom2
cotCustom3
cotCustom4
```

The values `cotCustom1` to `cotCustom4` are for user defined custom drawing tools.

Colors

Many properties of `DrawingObjects` expect a color value. For more information about color, see ***Color Constants*** (on page 1455).

Tagging

You can put a tag into a drawing tool. Later, you can easily find the drawing tool and work on it by the tag.

For example, to assign a tag of 9999 to a newly created drawing tool:

```
Id = DrawingObjects.Add(cotHorizontalLine)
DrawingObjects.Tag(Id) = 9999
```

Later in the code, if you want to find the drawing tool associated with tag 9999, and change the color of the drawing tool to red:

```
Id = DrawingObjects.GetIndexByTag(9999)
DrawingObjects.color(Id) = clRed
```

Coordinate System

When you specify the coordinate for a point in a drawing tool, there are three coordinate systems you can use: value, percent and exact.

If you use a value coordinate system (`cocValue`), then the X, Y value of the point is time and price respectively.

If you use a percent coordinate system (`cocPercent`), then the X, Y value are percentage (0%-100%, represented by 0 to 1) of the time chart. Thus a point at 0, 0 will be positioned at the upper left corner of the pane and a point at 1, 1 will be positioned at the lower right corner of the pane.

Exact coordinate system (`cocExact`) is valid only for X-axis for bar driven charts. It is an exact mapping to bar number, i.e. the first bar will have a coordinate of 0 and so on. Exact coordinate system lets you easily map to a coordinate in the future in a bar driven chart.

To set coordinate system, use XStyle and YStyle properties. For example,

```
XStyle[0] := cocValue;
YStyle[0] := cocPercent;
```

The parameter for `XStyle` and `YStyle` is the drawing tool is object id. In the example above, it is drawing tool id 0.

Looping Through the Drawing Tools

If you need to loop through all the drawing objects you have created, then you can use the `Count` property to get the number of drawing objects already created. You can then refer to these drawing objects using the index range from 0 to `Count` - 1. For example,

```
For i = 0 to DrawingObjects.Count - 1
    ' do whatever you need to do on each drawing objects here
Next
```

Update by Tick Issue (DrawingObjects only)

If your indicators is updated by tick in real-time, the drawing tools created by `DrawingObjects` with a tick will be removed automatically when the next tick arrives, unless a new bar is to be created.

This is the default behavior and is desirable because you do not have to manage the drawing tools on a per tick basic.

If you want to manage a drawing tool yourself on a per tick basis, set `AutoRemoveOnUpdateByTick` to true for the drawing tool.

IDL Issue (ChartDrawigObjects only)

`ChartDrawingObjects` is introduced in Version 4 of NeoTicker®. If you use an older IDL example as a starting point for your project, `ChartDrawingObjects` is not immediately ready yet.

To use `ChartDrawingObjects`, you need to change the `Reserved2` parameter in the IDL call to `ChartDrawingObjects`, and the type from `IDispatch` to `IDrawingObjects`. For example, in Delphi, change:

```
reserved2 : IDispatch
```

to

```
ChartDrawingObjects : IDrawingObjects
```

and recompile your project.

Methods and Properties

Drawing object properties are often referenced by the ID of the drawing object (e.g. ArrowStyle[ID]). You can get the ID when the drawing object is first created, e.g.

```
ID := DrawingObjects.Add(cotMarker);
```

Add (ObjectType)

Add a drawing tool. This method returns the integer Id of the drawing tool. For example,

```
Id := DrawingObjects.Add(cotMarker);
```

AndrewPitchforkConnectAB [ID] : boolean

Read/write property. Set the Andrew Pitchfork tool `Connect AB` option.

AndrewPitchforkConnectAC [ID] : boolean

Read/write property. Set the Andrew Pitchfork tool `Connect AC` option.

AndrewPitchforkMultiplesExtendTowardsA [ID] : boolean

Read/write property. Set the Andrew Pitchfork tool `Multiples Extend Towards A` option.

AndrewPitchforkSchiff [ID] : boolean

Read/write property. Set the Andrew Pitchfork tool `Schiff Style` option.

ArrowStyle[ID]

Set the arrow style. Valid values are `astNarrow` and `astWide`.

This property is applicable only to types that support arrows: `cotMarker, cotUpArrow, cotDownArrow.`

AutoRemoveOnUpdateByTick [ID]

This property is default to true. The indicator will automatically remove the drawing too; created by a previous real time update within the same bar on true setting. If you expect to fully manage the drawing tool yourself, you have to set this option to false after you created the drawing tool.

Border [ID]

You can set whether the drawing tool will display its border. This property is applicable only to `cotRectangle`, `cotOval`, and `cotText` types. It is a Boolean property.

Color [ID]

You can set the color.

Count

Return the number of drawing tools currently in this indicator.

CrossPane [ID]

You can read and set whether the drawing tool will draw across panes. This property is valid only for vertical lines and Fibonacci time.

Delete (ID)

Delete removes a drawing tool. For example,

```
DrawingObjects.Delete (Id)
```

DeleteAll

Remove all drawing tools in this indicator.

DrawingStyle [ID]

You can read and set the line drawing style of the drawing tool. Valid values are dsSolid, dsDot, dsDash.

ExtendLeft [ID], ExtendRight [ID]

You can set the extend to left or extend to right properties of the drawing tools that have these properties. They are Boolean properties which take either true or false.

Fill [ID]

You can set whether the drawing tool will fill its drawing area. This property is applicable only to `cotRectangle`, `cotOval`, and `cotText` types. It is a Boolean property.

FillColor [ID]

You can set the fill color of the drawing tool. This property is applicable only to `cotRectangle, cotOval`, and `cotText` types. It is an integer property.

Font [ID]

You can set and modify the font in use within the object. To change the font, set the `Font` property with the name of the font to be used. This property is applicable only to `cotText` type.

FontSize [ID]

You can set and modify the font size within the object. This property is applicable only to `cotText` type.

FontColor [ID]

You can set and modify the font color within the object. This property is applicable only to `cotText` type.

GetIndexByTag (Tag)

`GetIndexByTag` returns the index of the last item created in `DrawingObjects` having its property `Tag` equals to `ATag`. This returned index value can be used in methods like `SetPoint`, etc. to modify the properties of this existing drawing tools.

GetPoint(ID, Point, XValue, YValue) : boolean

Returns the coordinate of the control point of a drawing tool.

`ID` is the identifier of the drawing tool.

`Point` is index of the control point. It is either 0, 1, or 2. For example, trend line has two control points - square (0) and diamond(1). Some drawing tools such as channels have a third control point (2).

`XValue, YValue` are returned coordinates. The values are in time/price coordinate.

If invalid parameters are passed to `GetPoint`, the function returns false.

HeadOnly[ID]

Whether an arrow shows the arrow head only, or arrow head with body.

This property is applicable only to types that support arrows: `cotMarker, cotUpArrow, cotDownArrow.`

IntTagValues[ID, AIndex] : integer

`IntTagValues` is a read/write property. It can be used to store multiple integer values. `IntTagValues` persists over multiple evaluations.

`ID` is the index of the drawing tool, ranges from 0 .. `DrawingObjects.Count` - 1.

`AIndex` is a slot number. There are 20 pre-defined slots per drawing tool, so `AIndex` ranges from 0 .. 19.

LabelsVisible[ID] : boolean

Checks and sets label visibility for the drawing tool.

MarkerStyle[ID]

`MarkerStyle` returns the marker style of the drawing tool. This property is applicable only to `cotMarker` type.

`msSquare`	Square marker
`msCircle`	Circle marker
`msUp`	Up triangle
`msDown`	Down triangle
`msLeft`	Left triangle
`msRight`	Right triangle

ObjectType [ID]

`ObjectType` returns the type of the drawing tool indexed by `Id`.

OffsetLineVisible[ID]

`OffsetLineVisible` returns the visibility of the offset line in boolean. You can turn on/off offset line visibility by setting this property. `OffsetLineVisible` is applicable only to types that support offset lines: `cotMarker, cotUpArrow, cotDownArrow, cotRectange, cotEllipse, cotText`.

PaneCount

If the window that hosts the drawing tool is a time chart, `PaneCount` returns the number of panes in the chart. Otherwise, `PaneCount` returns 0.

Pane[ID]

Gets/sets the pane of the drawing tool. The range of acceptable values is from 1 to PaneCount, e.g.

```
ChartDrawingObjects.Pane[ID] := 1
```

will set the drawing object ID to pane 1.

You cannot set the pane of an indicator drawing tool (`DrawingObjects`), only of a chart drawing tool (`ChartDrawingObjects`).

PenWidth [ID]

You can set and modify the pen width within the object. It is an integer property.

PosStyle [ID]

To set the anchoring style of the `cotText` type. The available styles are as follows:

`tpsRectangle`	Text will be arranged to display within the defined boundary.

`tpsLeftTop,`
`tpsRightTop,`
`tpsLeftBottom,`
`tpsRightBottom`
	Text is anchored to the specific corner and the boundary will expand or contract to accommodate the text assigned

SaveTags[ID] : boolean

`SaveTags` is a read/write property. It can be used to signal NeoTicker to save the TagValues and IntTagValues for specific drawing objects. `SaveTags` persists over multiple evaluations.

`ID` is the index of the drawing tool, ranges from 0 .. `DrawingObjects.Count` - 1.

SetColors (ID, [color1, color2, ...])

Set color for each individual multiple of the `cotTrendChannel` or `cotFan` tool. `Color1`, `color2`, etc are integer values that makes up an array.

SetLineWidths (ID, [Width1, width2, ...])

Set line width for each individual multiple of the `cotTrendChannel` or `cotFan` tool. `Width1`, `width2`, etc are integer values that makes up an array. Line width of zero or negative value are translated to using default line width.

SetFibLineWidths (ID, [Width1, width2, ...])

Set line width for each individual multiple of any of the Fibonacci type objects (`cotFibLevel, cotFibTime, cotFibArc`). `Width1`, `width2`, etc are integer values that makes up an array. Line width of zero or negative value are translated to using default line width.

SetHMultiples, SetVMultiples, SetFMultiples, SetCMultiples (ID, [multiple1, multiple2, ...])

Set the horizontal line, vertical line, fan line, arc multiples respectively. These methods are applicable to `cotSupportResistance` or one of the `cotCustom` type. `Multiple1`, `multiple2`, etc are double values that makes up an array.

SetHColors, SetVColors, SetFColors, SetCColors (ID, [color1, color2, ...])

Set color for each individual multiple for horizontal line, vertical line, fan line and arc respectively. These methods are applicable to `cotSupportResistance` or one of the `cotCustom` type. `Color1`, `color2`, etc are integer values that makes up an array.

SetHLineWidths, SetVLineWidths, SetFLineWidths, SetCLineWidths (ID, [Width1, width2, ...])

Set line width for each individual multiple for horizontal line, vertical line, fan line and arc respectively. These methods are applicable to `cotSupportResistance` or one of the `cotCustom` type. `Width1`, `width2`, etc are integer values that makes up an array. Line width of zero or negative value are translated to using default line width.

SetFibMultiples (ID, [multiple1, multiple2, ...])

Set multiples to the ones in the list on any of the Fibonacci type objects (`cotFibLevel, cotFibTime, cotFibArc`).

SetFibColors (ID, [color1, color2, ...])

Set colors for each individual multiple added with `SetFibMultiples.`

SetMultiples (ID, List)

To set or change the multiples of the `cotTrendChannel` or `cotFan` tool, you can use this method. The `List` is an array of double values that will be used as if all the numbers are enabled to display.

```
DelphiScript - DrawingObjects.SetMultiples (0, [0.5, 1, 1.5, -
2])
VBScript     - DrawingObjects.SetMultiples (0, Array (0.5, 1,
1.5, -2))
```

SetPoint (ID, PointID, X, Y)

SetPoint positions the drawing tool. After setting the coordinate style, you can set its handles to the value you want directly. `PointID` refers to the ordered handles of the drawing object. It ranges from 0 to 2. For example, Horizontal Line has only one handle, thus you can set its handle using the `PointID` 0. If you are modifying the position of a trendline, then you can set the position of `PointID` 0 and 1.

Tag [ID]

It is a user customizable field for storing an integer for each drawing tool you have created. You can use this field as an identification value or for any other purpose you deem useful.

TagValues[AIndex, AItem]

`TagValues` can be used to store multiple values. `TagValue` persists over multiple evaluations.

`AIndex` is the index of the drawing tool, ranges from 0 .. `DrawingObjects.Count` - 1.

`AItem` is a slot id. There are 20 pre-defined slots per drawing tool, so `AItem` ranges from 0 .. 19.

Example usage (Delphi Script, assumed there are 2 indicator-created drawing tools):

```
DrawingObjects.tag[0, 3] := 5;
DrawingObjects.tag[0, 4] := 6;
DrawingObjects.tag[1, 4] := 7;
a := DrawingObjects.tag[0, 3];
b := DrawingObjects.tag[0, 4];
c := DrawingObjects.tag[1, 4];
// a is 5, b is 6, c is 7 at this point
```

Text [ID]

You can set and modify the text displayed within the object. This method is applicable only to the `cotText` type.

Visible [ID]

Set this property to true to make the drawing tool visible. It is a boolean property.

XOffset[ID], YOffset[ID]

Set the offset properties of the drawing tool. This method is applicable only to types that support offset line: `cotMarker, cotUpArrow, cotDownArrow, cotRectange, cotEllipse, cotText`.

XStyle [ID], YStyle [ID]

You can read or set the drawing style of an object separately on each part of its coordinates.
The drawing style you can choose from is one of the following:

`cocValue` Enforce the coordinate to display in the actual coordinate based on the pane axis.

`cocPercent` Enforce the coordinate to display the drawing tool using screen position based on percentage of the screen. For XStyle, 0 represents the left side of the pane and 1 represents the right side of the pane. For YStyle, 0 represents the bottom of the pane and 1 represents the top of the pane. This option allows a drawing object to stay on the chart as a "sticky" item.

`cocExact` This setting is defined only when the chart is in bar driven mode and for X-axis only. X value is the bar number.

Use this setting if you want the drawing tool to draw into the future with a concise coordinate in bar driven mode (in bar driven mode, future date time position is undefined with `cocValue` and `cocPercent`).

Trade Object

`Trade` Object is designed to assist the design and tracking of trading systems.. `Trade` object provides trading functionality such as order entry and equity tracking.

You can think of `Trade` object as a virtual brokerage account. You can create trading system rules to place orders, cancel orders, requesting transaction records, as if you are working with a real life one. This way, you can relate the trading system you have created much more closely towards actual usage in real life.

`Trade` object is designed with the following principles in mind.

Real Life Order Placement Emulation

The order placement mechanism is designed to work like orders you place through your brokers. Orders are placed, canceled, and managed by the user. Orders have their own lifetime within the trading system and will stay in the system as long as they should be.

For example, you have a buy stop day order. This order will stay in the system for the full day that it is placed and will be canceled after the day is over. During the time period in which the order is active, it will be filled if its condition is met.

True Portfolio Analysis

`Trade` object can track multiple positions at the same time. Orders can be placed on specific data series to allow the ability of true portfolio analysis.

Complete History

`Trade` object holds the complete list of orders being placed and all the trades it has filled. You can easily access all these records as if you are looking at an account statement. You can easily create your own custom analysis of your system using this extensive information.

Equity Driven

`Trade` object emulates a trading account. It tracks all the transactions and provides feedback to the trading system what the current cash and open positions are any time. You can easily incorporate your equity management rules into your trading system.

Safe Guard Against Peek Into Future

The overall design of NeoTicker®'s indicator calculation architecture forbids the users from looking ahead into the future. This design is inherited by `Trade` object. Thus your trading system designed in NeoTicker® would not be mistakenly created with rules driven by future data information.

Broker Properties and Methods

GetBrokerPos(symbol, reportedsize, reportedprice, estsize, estprice)

Returns the position of symbol at your brokerage. This function requires you to have either a live brokerage or Trade Simulator Server configured in NeoTicker® to carry out your trades. See ***Order Related Topics*** (on page 693) for more information.

`Reportedsize, reportedprice, estsize` and `estprice` are return values.

`Reportedsize` is the position size reported by your broker. All brokers provide this information.

`Reportedprice` is the position price reported by your broker. Not all brokers provide this information. This value is 0 if your broker does not provide this information

`Estsize` is the position size estimated by NeoTicker®. Provided that the symbol has not been traded manually outside of NeoTicker®, this value should match that of `reportedsize`.

`Estprice` is the position price estimated by NeoTicker®. If your broker does not provide position price information, you can substitute it by `estprice` when:

- `Reportedsize` equals `estsize`.
- `Reportedprice` is 0.

Closed Positions Properties and Methods

Trade object supports the following properties for direct access to detail information of each closed position. A closed position is defined as establishing a net long or net short position and then over time have the position closed out with trades that offset the net long or net short. Within one closed position, it can only take on one single trade direction (long or short), but can have multiple entries and exits that change the size of the position before finally taking the position to out of market (flat position).

ClosedPositionCount

Returns the number of closed positions recorded so far by the Trade object. The number of closed positions is always less than the number of transactions.

ClosedPositions [Index]

Returns a specific position object located at `Index` positions ago. `Index` ranges from 0 to `ClosedPositionCount` -1. For example, `ClosedPositions [0]` refers to the last closed position while `ClosedPositions [1]` refers to the position that was completed right before the last one.

`ClosedPositions` returns an instance of the Position object. For example, to check if the last closed position is long position, you can use the property long:

```
if Trade.ClosedPositions[0].Long then
```

ClosedPositions returns a position object. For properties and methods of position object, refer to ***Position Object Properties and Methods*** (on page 1596).

Order Placement - Basic Buy Sell Order Methods

All of the buy/sell methods here return the `OrderId` that you can save in your trading system for future reference.

The Buy/Sell order combination allows you to enter a trade and exit a trade easily. There are special consideration to deal with when your system is getting more complex. For example, when your system is long at the moment and you are interested to reverse from a long position to a short one, you will need to first exit the long position and then place an order to sell short the quantity you want. Or, you can place a sell order than has the size equals to the sum of the current long position size and the quantity you are short selling. Either case you will change the current position of the system from long to short.

BuyAtMarket(Size, Comment)

Buys at market in the next bar.

BuyLimit(Price, Size, TimeInForce, Comment)

Buys at limit price.

BuyNextClose(Size, Comment)

Buys at the end of next bar, at the closing price of the next bar. Trading system forbids buying at the current close because it is possible to construct look ahead system when buying at current close price.

BuyStop(Price, Size, TimeInForce, Comment)

Buys at market when price is hit.

SellAtMarket(Size, Comment)

Sells at market in the next bar.

SellLimit(Price, Size, TimeInForce, Comment)

Sells at limit price.

SellNextClose(Size, Comment)

Sells at the end of next bar, at the closing price of the next bar. Trading system forbids selling at the current close because it is possible to construct look ahead system when selling at current close price.

SellStop(Price, Size, TimeInForce, Comment)

Sells at market when price is hit.

CancelOrder(OrderId)

Cancels an existing open order. Refer to ***Order properties and Methods*** (on page 1589).

Order Placement - Market Day Order Methods

Market day order methods are specifically designed for trading systems that places market orders:

- At a low time frame (1-minute and below), or
- That have a high chance of not being filled

With a market day order, the order has a time in force of otfDay, and the order will stay until end of trading day if not filled.

So why and when should one use market day order? In an ideal world, you don't. For a market order, as long as there are trades, the orders will be filled. This is true in both system testing, and for live deployment of systems that work on a higher time frame (1 minute or higher).

This assumption starts to break down at low time frame. We will consider two scenarios.

Scenario 1 - Low Time Frame

Suppose you have a trading system that works on 5-second bars. Your trading system places a market order. The order needs to send to your broker for execution. Then the order status is returned from your broker to the trading system. This round trip of sending order receiving status can take significant time. Problem will happen when this round trip takes more than 5 seconds and if your trading system relies on your brokerage for fill reporting.

In 5 seconds, your trading system will need to create a new bar. However, because of the delay caused by the round trip, it does know the order is filled. With the standard time in force otfFillorKill logic, your trading system will cancel the order. But in fact, we know most of the time, your broker will fill such market order. It is just the order status cannot arrive in time. The result is an order that is cancelled in your trading system, but in your brokerage account, the order is filled.

Scenario 2 - Broker Unable To Fill Order

This scenario is the opposite of scenario 1. If your trading system does not rely on broker for fill reporting, the trading system will assume all market orders are filled as long as there are trades. However, if for some reason, your broker cancels the order sent by the trading system, the trading system will think the order is filled, while the order is cancelled in your brokerage account.

Consequences

Both scenarios lead to a mismatch in position and in the worst case, can cause the trading system logic to malfunction.

Scenario 1 is almost certain to happen with low time frame trading systems.

Scenario 2 is rare because we know that most of the time, market orders are filled, but it can happen with extremely thinly traded instruments, if your Internet connection is unreliable or if your trading account near its margin limit.

Day Order and Fill Reporting Settings

If you use market day order, you will need to set a time to carry out day order processing. This is the time needed for trading system to cancel orders at your brokerage account. So you need to assign a reasonable amount of time to do that.

Day order and fill reporting settings are set under trading system's brokerage settings. They are located in Edit Indicator dialog's **Brokerage 1** tab. See ***Time Chart, Sending Orders to Your Broker*** (see "Sending Orders to Your Broker" on page 1203).

Methods

Market day orders have identical syntax to their non-day order counter parts. For reference, refer to ***Order Placement - Basic Buy Sell Order Methods*** (on page 1580), ***Order Placement - Smart Order Methods*** (on page 1586) and ***Portfolio Order Placement Methods*** (on page 1598).

Below is the list of single instrument methods:

- `BuyAtMarketDay`
- `SellAtMarketDay`
- `LongAtMarketDay`
- `LongExitAtMarketDay`
- `ShortAtMarketDay`
- `ShortExitAtMarketDay`
- `ExitCurrentPositionDay`

Below is the list of portfolio methods:

- `BuyAtMarketDayEx`
- `SellAtMarketDayEx`
- `LongAtMarketDayEx`
- `LongExitAtMarketDayEx`
- `ShortAtMarketDayEx`
- `ShortExitAtMarketDayEx`
- `ExitCurrentPositionDayEx`
- `ExitAllCurrentPositionsDay`

Order Placement - Next Open Methods

The following single instrument methods place order with a price based on next open. An offset value is added to the next open for the ordering price. An equivalent set of methods are available for portfolio system.

For example, `BuyNextOpenOffsetLimit` places an order using next open price plus an offset as the limit price.

Here is the filling mechanism when testing against historical data:

1. In the current bar, a next open order is placed. The order is not filled.
2. At the next bar, the OHLC values are available. As long as the next bar prices is better than or equal to the limit price (open price + offset), the order is considered to be filled.

When you place a next open order in real-time, here is the filling mechanism:

1. In the current bar, a next open order is placed. The order is not filled.
2. When the current bar is completed when the first tick of next bar arrives, the order price becomes available. At this stage, if the trading system is updated by tick, the order will be filled when the arriving ticks matches the price. If the trading system is non-updated by tick, the order will be filled similarly to historical testing.
3. If the trading system is connected to a real-life broker or Trade Simulator through Order Interface, when the first tick of the next bar arrives, the order now has the order price and is sent to broker or Trade Simulator for filling.

Warning: although technically orders based on next open do not reveal future information, it is possible to use such orders combined with stop orders to create systems that are close to impossible to execute in real-time. You need to take extra caution when creating and deploying systems that are based on next open.

BuyNextOpenOffsetLimit(Offset, Size, TimeInForce, Comment)

Similar to BuyLimit, but use next open price + offset as the price.

SellNextOpenOffsetLimit(Offset, Size, TimeInForce, Comment)

Similar to SellLimit, but use next open price + offset as the price.

LongNextOpenOffsetLimit(Offset, Size, TimeInForce, Comment)

Similar to LongLimit, but use next open price + offset as the price.

ShortNextOpenOffsetLimit(Offset, Size, TimeInForce, Comment)

Similar to ShortLimit, but use next open price + offset as the price.

LongExitNextOpenOffsetLimit(Offset, Size, TimeInForce, Comment)

Similar to LongExitLimit, but use next open price + offset as the price.

ShortExitNextOpenOffsetLimit(Offset, Size, TimeInForce, Comment)

Similar to ShortExitLimit, but use next open price + offset as the price.

BuyNextOpenOffsetStop(Offset, Size, TimeInForce, Comment)

Similar to BuyStop, but use next open price + offset as the price.

SellNextOpenOffsetStop(Offset, Size, TimeInForce, Comment)

Similar to SellStop, but use next open price + offset as the price.

LongNextOpenOffsetStop(Offset, Size, TimeInForce, Comment)

Similar to LongStop, but use next open price + offset as the price.

ShortNextOpenOffsetStop(Offset, Size, TimeInForce, Comment)

Similar to ShortStop, but use next open price + offset as the price.

LongExitNextOpenOffsetStop(Offset, Size, TimeInForce, Comment)

Similar to LongExitStop, but use next open price + offset as the price.

ShortExitNextOpenOffsetStop(Offset, Size, TimeInForce, Comment)

Similar to ShortExitStop, but use next open price + offset as the price.

Order Placement - Order Interface Methods

OrderPlacementEnabled

When true, orders are sent to Order Interface Manager for further processing. Order Interface Manager will decide whether the order will be sent to Trade Simulator for realistic tick-by-tick simulation of orders, or to a real-life broker to make real orders.

OrderPlacementFillReport

Set to either frOrderDataFeed or frOrderServer.

If set to frOrderDataFeed, data from the data feed will be used to resolve fill price using Trade object's fill mechanism. This setting is used for backtesting, and for real-life brokers who do not return fill price.

If set to frOrderServer, the fill price will be returned from order interface. Therefore, OrderPlacmentEnabled must be set to true before this option will take effect. Note that not all real-life brokers return fill price information.

Order Placement - Smart Order Methods

To simplify system design, you can choose to use the following order placement methods that will take care of handling long and short position smartly.

The advantage of these meta-style order methods is that you can focus on system logic and avoid the tedious coding of taking care of position sizes.

The disadvantage is that if your system is going to be modified into automatic order placement with some execution platform, then you may be limited by the target order execution platform and have to change your code back to the basic buy/sell orders. See ***Position Discrepancy*** (see "Issue: Position Mismatch" on page 791) for an explanation.

LongAtMarket(Size, Comment)

Establishes a long position at market in the next bar. If there is currently a short position, it will be closed for you at the same time.

LongLimit(Price, Size, TimeInForce, Comment)

Establishes a long position at limit price. If there is currently a short position, it will be closed for you at the same time.

LongNextClose(Size, Comment)

Establishes a long position in the next bar, at the closing price of the next bar. Trading system forbids buying at the current close because it is possible to construct look ahead system when buying at current close price.

LongStop(Price, Size, TimeInForce, Comment)

Establishes a long position at market when price is hit. If there is currently a short position, it will be closed for you at the same time.

LongExitAtMarket(Size, Comment)

Exits a long position at market in the next bar. If the size is larger than the current position size, it will be capped at the current position size. If there is no long position, the order will not be placed at all.

LongExitLimit(Price, Size, TimeInForce, Comment)

Exits a long position at limit price. If there is no long position, the order will not be placed at all.

LongExitNextClose(Size, Comment)

Exits a long position in the next bar, at the closing price of the next bar. Trading system forbids selling at the current close because it is possible to construct look ahead system when selling at current close price.

LongExitStop(Price, Size, TimeInForce, Comment)

Exit a long position at market when price is hit. If there is no long position, the order will not be placed at all.

ShortAtMarket(Size, Comment)

Establishes a short position at market in the next bar. If there is currently a long position, it will be closed for you at the same time.

ShortLimit(Price, Size, TimeInForce, Comment)

Establishes a short position at limit price. If there is currently a long position, it will be closed for you at the same time.

ShortNextClose(Size, Comment)

Establishes a short position at the end of next bar, at the closing price of the next bar. Trading system forbids selling at the current close because it is possible to construct look ahead system when selling at current close price.

ShortStop(Price, Size, TimeInForce, Comment)

Establishes a short position at market when price is hit. If there is currently a long position, it will be closed for you at the same time.

ShortExitAtMarket(Size, Comment)

Exits a short position at market in the next bar. If the size is larger than the current position size, it will be capped at the current position size. If there is no short position, the order will not be placed at all.

ShortExitLimit(Price, Size, TimeInForce, Comment)

Exits a short position at limit price. If there is no short position, the order will not be placed at all.

ShortNextClose(Size, Comment)

Exits a short position in the next bar, at the closing price of the next bar. Trading system forbids buying at the current close because it is possible to construct look ahead system when buying at current close price.

ShortExitStop(Price, Size, TimeInForce, Comment)

Exits a short position at market when price is hit. If there is no short position, the order will not be placed at all.

Order Placement - Special Order Handling Methods

To allow simplification of order handling within a script, some complex order placing operations are created.

ExitCurrentPosition (Comment)

`ExitCurrentPosition` places a market order that exactly offsets the current open position size such that by the end of the next bar, the system will be flat.

CancelOrdersByComment (Comment)

`CancelOrderByComment` will cancel all open orders with the comment specified. Very useful for systems will multiple triggering in opening and closing positions.

CancelAllOrdersByType(OrderType)

Cancel all orders by the type specified, e.g. `otBuyLimit`, `otSellShort`. For a list of order types, see ***Order Object*** (on page 1590).

CancelAllOpenOrders, CancelAllBuyOrders, CancelAllSellOrders, CancelDayOrders, CancelAllExitOrders

These cancel orders methods will cancel orders of the type you specified all in one single call. There is no need of canceling the open orders one by one.

UpdateByTickRestoreCondition

By default, this is true. See ***Trading System Settings*** (on page 1193), under Update By Tick Restore Condition for more information.

Order Placement - Time In Force

Orders that are not market has a time in force parameter.

For example, the `BuyLimit` method has a time in force parameter. The call

```
Trade.BuyLimit(25, 100, otfDay, '')
```

will place a buy limit order at $25 for 100 shares. The order will stay open for the trading day.

Below is the time in force table:

`otfFillorKill`	If order is not filled by the next bar, it will be cancelled
`otfDay`	Order will stay for the day only
`otfGoodtilCancel`	Order is placed to stay in the system until it is filled or cancelled

Order Properties and Methods

Trade object supports the following properties for direct access to detail information of all orders.

OpenOrderCount

Returns the number of open orders at this point of calculation.

OpenOrders [Index]

Returns a specific order object located at the Index position. Index ranges from 0 to `OpenOrderCount` – 1. Orders are in chronological order. They are listed according to the time they are placed.

OrderCount

Returns the number of orders at this point of calculation.

Orders[Index]

Returns a specific order object located at the Index position. Index ranges from 0 to `OrderCount` – 1. Orders are in chronological order. They are listed according to the time they are placed

OrderById [AOrderId]

Returns the order object with `OrderId` equals to `AOrderId.`

OrderByComment ["Comment"]

Returns the last order object with the specified comment.

Order Object

An order object encapsulates the detail information of an order. You can access this detail information through the following properties.

Order objects are supporting object of `Trade` object. They are returned by order methods of `Trade` object.

BarsNum

Read-only. This property returns the absolute bars number when the order is placed.

OrderId

Returns an integer representing the unique identification number of the order. For example, Trade.CancelOrder method needs the order id as a parameter.

OrderTime

Returns the time of order placement.

OrderType

Returns the type of the order placed, it can be one of the following:

```
otBuyAtMarket
otBuyLimit
otBuyNextClose
otBuyStop
otBuyClose - not available, special internal use only
otBuyExact - not available, special internal use only

otSellAtMarket
otSellLimit
otSellNextClose
otSellStop
otSellClose - not available, special internal use only
otSellExact - not available, special internal use only

otLongAtMarket
otLongLimit
otLongNextClose
otLongStop

otLongExitAtMarket
otLongExitLimit
otLongExitNextClose
otLongExitStop

otShortAtMarket
otShortLimit
otShortNextClose
otShortStop

otShortExitAtMarket
otShortExitLimit
```

```
otShortExitNextClose
otShortExitStop

otBuyNextOpenOffsetStop
otBuyNextOpenOffsetLimit
otSellNextOpenOffsetStop
otSellNextOpenOffsetLimit

otLongNextOpenOffsetStop
otLongNextOpenOffsetLimit
otShortNextOpenOffsetStop
otShortNextOpenOffsetLimit

otLongExitNextOpenOffsetStop
otLongExitNextOpenOffsetLimit
otShortExitNextOpenOffsetStop
otShortExitNextOpenOffsetLimit
```

TimeFrame

Returns the time control type of the order. It is the same time frame information you have provided using one of the order placement methods.

Price

Returns the price of the order. If the order is market order, the price equals zero.

Size

Returns the size of the order.

Comment

Returns the comment of the order.

Symbol

Returns the symbol that the order is placed for.

LinkId

Returns the link id of the order. Link Id equals 0 if the order is placed on the primary linked series. Link Id value N greater than 0 points to the underlying series is `DataSeries (N).`

Status

Returns the status of the order. It can be one of the following

```
osOpen
osPending
osFilled
osCancel
```

FilledSize

Returns the filled size of the order. This is used when the trading system is connected to a life broker and is for checking partial fills.

FilledAvgPrice

Returns the filled size of the order. This is used when the trading system is connected to a life broker and the fill is comprised of multiple trades.

Open Position Management Properties and Methods

Open position of the trading system refers to the combined position from the current trade(s) that has not been offset by closing trade(s) that flat out the position.

Using the provided properties and methods, you can easily check on the current open position status and utilizes the information in many ways, like applying trailing stop orders, target profit taking orders, etc.

OpenPositionLong

Returns the boolean (either true or false) state on whether the linked series has a long position.

OpenPositionShort

Returns the boolean (either true or false) state on whether the linked series has a short position.

OpenPositionFlat

Returns the boolean (either true or false) state on whether the linked series has no position.

OpenPositionPartialExit

Returns the boolean (either true or false) state on whether the linked series has a position and partial exit trade(s) is taken.

OpenPositionSize

Returns the current position size. Positive values indicate the position is a long and negative values indicate the position is a short. 0 indicates the system is flat.

OpenPositionAbsSize

Shortcut for the size of the current open position without directional information.

OpenPositionPL

Returns a double read-only value. It is the current profit or loss on the open position you are holding at the moment.

OpenPositionCost

Returns the average cost per unit for the current open position. This function is useful for systems that utilize multiple entries and exits.

OpenPositionBestPriceLevel

Returns the best price level reached within the lifetime of the current open position.

OpenPositionWorstPriceLevel

Returns the worst price level reached within the lifetime of the current open position.

OpenPositionEntryPrice

Returns the first entry price of the current open position.

OpenPositionEntryBar

Returns the first entry bar id of the current open position.

OpenPositionEntryDateTime

Returns the first entry date time of the current open position.

OpenPositionEntrySize

Returns the first entry size of the current open position.

OpenPositionEntryTransId

Returns the transaction id of first entry of the current open position.

OpenPositionAverageEntryPrice

Returns the volume weighted average entry price of the current open position.

OpenPositionLastExitPrice

Returns the last exit price of the current open position.

OpenPositionLastExitBar

Returns the last exit bar id of the current open position.

OpenPositionLastExitSize

Returns the last exit size of the current open position

OpenPositionLastExitTransId

Returns the transaction id of last exit of the current open position

OpenPositionAverageExitPrice

Returns the average exit price of the current open position

The Last Exit information is available if and only if partial exit happened during the lifetime of the open position. As the final exit took place, the open position will be closed and the open position information is reset. You can always access to the closed positions with the properties `ClosedPositionCount` and `ClosedPositions [N]`.

All the methods listed above have corresponding `Ex` version for tracking specific position belonging to a specific data series.

For example, `OpenPositionBestPriceLevelEx [2]` will return the best price level of the current open position of `DataSeries [2]`.

Portfolio Closed Positions Properties and Methods

Similar to its counterpart for single instrument system, portfolio closed positions can be accessed with the following Ex extended properties.

ClosedPositionCountEx [Issue]

Returns the number of closed positions for the specific `Issue` recorded so far by `Trade` object.

ClosedPositionsEx [Issue, Index]

Returns a specific position object located at `Index` positions ago. `Index` ranges from 0 to `ClosedPositionCountEx [Issue]` -1. For example, `ClosedPositionsEx [1, 0]` refers to the last closed position of `DataSeries.Items [1]` while `ClosedPositionsEx [1, 1]` refers to the position of the same data item that was completed right before the last one.

Position object does not make distinction between single or portfolio version. For position object properties and methods refer to ***Position Object Properties and Methods*** (on page 1596).

Position Object Properties and Methods

A position object encapsulate the detail information of a closed position. You can access these detail information through the following properties.

Position objects are supporting object of `Trade` object. They are returned by closed position methods of `Trade` object.

Long

Returns true if this is a net long position.

Short

Returns true if this is a net short position.

PartialExit

Returns true if this position exit more than once before it is completed .

MaxSize

Returns the maximum number of shares or contracts that this position has ever hold at any point in time over the duration of the position.

NumEntries

Number of times the position has increased its size.

NumExits

Number of times the position has decreased its size.

EntryPrice

Returns the first entry price (in Double format) of the position.

EntrySize

Returns the first entry size (in Integer format) of the position.

EntryBar

Returns the first entry's barsnum (in Integer format) of the underlying data series.

EntryDateTime

Returns the first entry date time (in Double format / Internal date time format) of the position in internal format.

EntryTransId

Returns the first entry transaction id (in Integer format) of the position. To retrieve the specific trade for its information you can use the property `Trade.TransactionById` with this id.

LastExitPrice

Returns the last exit price (in Double format) of the position.

LastExitSize

Returns the last exit size (in Integer format) of the position.

LastExitBar

Returns the last exit barsnum (in Integer format) of the underlying data series.

LastExitDateTime

Returns the last exit date time (in Double format / Internal date time format) of the position in internal format.

LastExitTransId

Returns the last exit transaction id (in Integer format) of the position. To retreive the specific trade for its information you can use the property `Trade.TransactionById` with this id.

AverageEntryPrice

Returns the volume weighted average price of all the entries (in Double format) of the position.

AverageExitPrice

Returns the volume weighted average price of all the exits (in Double format) of the position.

PositionCommission

Returns the total commission paid (in Double format) throughout the duration of the position.

PositionProfit

Returns the total profit before commission (in Double format) throughout the duration of the position.

BestPriceLevel

Returns the best price level of the position.

WorstPriceLevel

Returns the worst price level of the position.

Portfolio Open Position Management Properties and Methods

Portfolio open position management is the extension of the normal open position management properties and methods.

All properties and methods listed here use the `Issue` parameter to identify which data series within the context to return the information requested.

For basic information about these properties, please refer to ***Open Position Management Properties and Methods*** (on page 1593).

OpenPositionLongEx [Issue]

OpenPositionShortEx [Issue]

OpenPositionFlatEx [Issue]

OpenPositionPartialExitEx [Issue]

OpenPositionSizeEx [Issue]

OpenPositionAbsSizeEx [Issue]

OpenPositionPLEx [Issue]

OpenPositionCostEx [Issue]

OpenPositionBestPriceLevelEx [Issue]

OpenPositionWorstPriceLevelEx [Issue]

OpenPositionEntryPriceEx [Issue]

OpenPositionEntryBarEx [Issue]

OpenPositionEntryDateTimeEx [Issue]

OpenPositionEntrySizeEx [Issue]

OpenPositionEntryTransIdEx [Issue]

OpenPositionAverageEntryPriceEx [Issue]

OpenPositionLastExitPriceEx [Issue]

OpenPositionLastExitBarEx [Issue]

OpenPositionLastExitSizeEx [Issue]

OpenPositionLastExitTransIdEx [Issue]

OpenPositionAverageExitPriceEx [Issue]

Portfolio Order Placement Methods

The portfolio order placement methods function exactly the same as the regular order placement methods except that an extra parameter `Issue` is taken to identify which `DataSeries` member the order is for. `Issue` is an integer range from 1 to `DataSeries.Count`. If the `Issue` does not exists, the order will not be placed.

For order placement methods that are based on next open, make sure you understand the implication of next open orders.

BuyAtMarketEx (Issue, Size, Comment)

SellAtMarketEx (Issue, Size, Comment)

BuyNextCloseEx(Issue, Size, Comment)

SellNextCloseEx(Issue, Size, Comment)

BuyLimitEx (Issue, Price, Size, TimeInForce, Comment)

SellLimitEx (Issue, Price, Size, TimeInForce, Comment)

BuyStopEx (Issue, Price, Size, TimeInForce, Comment)

SellStopEx (Issue, Price, Size, TimeInForce, Comment)

LongAtMarketEx (Issue, Size, Comment)

LongLimitEx (Issue, Price, Size, TimeInForce, Comment)

LongNextCloseEx(Issue, Size, Comment)

LongStopEx (Issue, Price, Size, TimeInForce, Comment)

LongExitAtMarketEx (Issue, Size, Comment)

LongExitLimitEx (Issue, Price, Size, TimeInForce, Comment)

LongExitNextCloseEx (Issue, Size, Comment)

LongExitStopEx (Issue, Price, Size, TimeInForce, Comment)

ShortAtMarketEx (Issue, Size, Comment)

ShortLimitEx (Issue, Price, Size, TimeInForce, Comment)

BuyNextOpenOffsetLimitEx(IssueId, Offset, Size, TimeInForce, Comment)

SellNextOpenOffsetLimitEx(IssueId, Offset, Size, Type, Comment)

LongNextOpenOffsetLimitEx(IssueId, Offset, Size, Type, Comment)

ShortNextOpenOffsetLimitEx(IssueId, Offset, Size, Type, Comment)

LongExitNextOpenOffsetLimitEx(IssueId, Offset, Size, Type, Comment)

ShortExitNextOpenOffsetLimitEx(IssueId, Offset, Size, Type, Comment)

BuyNextOpenOffsetStopEx(IssueId, Offset, Size, Type, Comment)

SellNextOpenOffsetStopEx(IssueId, Offset, Size, Type, Comment)

LongNextOpenOffsetStopEx(IssueId, Offset, Size, Type, Comment)

ShortNextOpenOffsetStopEx(IssueId, Offset, Size, Type, Comment)

LongExitNextOpenOffsetStopEx(IssueId, Offset, Size, Type, Comment)

ShortExitNextOpenOffsetStopEx(IssueId, Offset, Size, Type, Comment)

ShortNextCloseEx (Issue, Size, Comment)

ShortStopEx (Issue, Price, Size, Type, Comment)

ShortExitAtMarketEx (Issue, Size, Comment)

ShortExitLimitEx (Issue, Price, Size, Type, Comment)

ShortExitNextCloseEx (Issue, Size)

ShortExitStopEx (Issue, Price, Size, Type, Comment)

Portfolio Reporting Methods

Portfolio-wide reporting methods are for tracking the complex situation of multiple positions at the same time.

ReportOpenPositions (ReportWindow)

`ReportOpenPositions` returns a summary report of all the open positions you have at the moment. The report is sent to `ReportWindow`.

Portfolio Special Order Handling Methods

The portfolio special orders are designed to let you apply the same special orders on specific Issue. The methods function exactly the same as the regular order placement methods except that an extra parameter `Issue` is taken to identify which `DataSeries` member the method.

ExitCurrentPositionEx (Issue, Comment)

CancelOrdersByCommentEx (Issue, Comment)

CancelAllOpenOrdersEx (Issue)

CancelAllBuyOrdersEx (Issue)

CancelAllSellOrdersEx (Issue)

CancelAllExitOrdersEx (Issue)

CancelDayOrdersEx (Issue)

CancelAllOrdersByType(Issue, OrderType)

ExitAllCurrentPositions (Comment)

To exit all the positions you have in a portfolio, you can use the `ExitAllCurrentPosition` method.

Portfolio System Setting Properties

When using `Trade` object to track orders placed for data other than the primary linked series, the trade object will recognize the `DataSeries` object and uses the data available from the `DataSeries` object to execute buy sell signals and compute their related statistics.

DefaultOrderSizes [Index]

Returns the default order size. Default order size is set either in Symbol Info Manager or when applying the trading system.

MarginCount

`MarginCount` is an integer property. You can set this property to match the number of data series items available in your current indicator instance. (e.g. `Trade.MarginCount = DataSeries.count`) Once you have set this item, you can then assign values to the `MarginTypes` and `MarginValues` indexed properties for each indexed items. This property can only be set before the first order is placed.

MarginTypes [Index]

`MarginTypes` is an indexed property having the same set of possible values as the property `MarginType`. Its index range from 1 to `MarginCount`. You can set the margin type for `DataSeries [N]` by assigning the appropriate value to `MarginTypes [N]`. You have to set the `MarginTypes` values before the first order is placed.

MarginValues [Index]

`MarginValues` is an indexed double property. Its index range from 1 to `MarginCount`. You can set the margin value for `DataSeries [N]` by assigning the appropriate values to `MarginValues [N]` matching the setting of `MarginTypes [N]`. You have to set the `MarginValues` values before the first order is placed.

MinTickSizes [Index]

`MinTickSizes` is an indexed double property. Its index range from 1 to `PriceMultipleCount`. You can set the minimum tick size for `DataSeries [N]` by assigning the appropriate value to `MinTickSizes [N]`. You have to set the `MinTickSizes` values before the first order is placed.

OverrideMargin

`OverrideMargin` is a boolean property. It is default to false. If you set `OverrideMargin`, then `Trade` object will seek for the `MarginTypes` and `MarginValues` indexed properties for the appropriate margin information for the specific `DataSeries` mapped with the same index. If a specific set of margin information cannot be located or not available, the standard value from `MarginType` and `MarginValue` properties will be used instead.

OverrideMultiple

`OverrideMultiple` is a boolean property. It is default to false. If you set `OverrideMultiple`, then `Trade` object will seek for the `PriceMultiples` indexed property for the appropriate price multiple information for the specific `DataSeries` mapped with the same index. If a specific price multiple is not found or not available, the standard value from `PriceMultiple` property will be used instead.

PriceMultipleCount

`PriceMultipleCount` is an integer property. You can set this property to match the number of data series items available in your current indicator instance. (e.g. `Trade.PriceMultipleCount = DataSeries.count`) Once you have set this item, you can then assign values to the `PriceMultiples` indexed property for each indexed values. This property can only be set before the first order is placed. This property also affects the `MinTickSizes` property.

PriceMultiples [Index]

`PriceMultiples` is an indexed double property. Its index range from 1 to `PriceMultipleCount`. You can set the price multiple for `DataSeries [N]` by assigning the appropriate multiple to `PriceMultiples [N]`. You have to set the `PriceMultiples` values before the first order is placed.

ScaleOrderSizes [Index]

Returns the scale order size. Scale order size is set either in Symbol Info Manager or when applying the trading system.

Reporting Methods

Use Reporting methods when you want reporting in a text only form. System Performance Viewer is a better tool if you want to analyze the system interactively.

The reporting methods allow you to verify what has happened during the historical period. You can choose different combinations of reports that you want to look at and add them into a script.

Reporting methods are time consuming and can significantly slow down your computer if you try to output reports from a system that is updating in real-time. Worst yet, if you update such a system in real-time tick-by-tick, you could stall NeoTicker® totally. Learn to limit the number of times that reports are generated through using the heap object and `Data1.IsLastBar` property.

ReportAllOrders(WindowName)

`ReportAllOrders` sends to a report window the complete list of all orders placed during the testing period of the system. This will report all open, canceled, and filled orders. `WindowName` is the name of the report window. If you leave `WindowName` as an empty string, the first report window in active group is used.

ReportLastNOrders(WindowName, NumOfOrders)

`ReportLastNOrders` reports the last `NumOfOrders` orders.

ReportAllTrades(WindowName)

`ReportAllTrades` reports all trades (filled orders).

ReportAllEquityChanges(WindowName)

`ReportAllEquityChanges` generates the equity curve.

ReportCompressedEquityChanges(WindowName, CompressLevel)

`ReportCompressedEquityChanges` generates a compressed equity curve. `CompressLevel` controls the time frame to report the changes. The three time frames that you can specify are `clDaily`, `clWeekly` and `clMonthly`.

ReportTradesSummary(WindowName)

`ReportTradesSummary` generates a transaction driven summary of the trading system.

ReportSystemSummary(WindowName)

`ReportSystemSummary` generates a symbol driven summary of the trading system.

Reporting Methods for Microsoft Excel

`Trade` object has the ability to output its reports directly to Microsoft Excel. The methods that send reports to Excel perform exactly the same task as the ones that output to report windows. Thus, you can just pick the ones you need to use inside the scripts and then apply further analysis from within Excel.

All methods that send data to Microsoft Excel are ended with the suffix `2Excel`.

ReportAllOrders2Excel

ReportAllTrades2Excel

ReportAllEquityChanges2Excel

ReportCompressedEquityChanges2Excel (CompressLevel)

ReportTradesSummary2Excel

ReportSystemSummary2Excel

The methods are equivalent to normal reporting methods. Refer to ***Reporting Methods*** (on page 1603) for more information.

The following reporting methods work exactly like the ones above except that you can specify the target workbook (`ABook`) and worksheet (`ASheet`), plus the ability to start from your choice of upper left corner cell position with `ACol` and `ARow`. If `AutoCreate` is set to true, it will create the necessary item for putting the report on.

ReportAllOrders2ExcelEx (ABook, ASheet, ACol, ARow, AutoCreate)

ReportAllTrades2ExcelEx (ABook, ASheet, ACol, ARow, AutoCreate)

ReportAllEquityChanges2ExcelEx (ABook, ASheet, ACol, ARow, AutoCreate)

ReportCompressedEquityChanges2ExcelEx (ABook, ASheet, ACol, ARow, AutoCreate, CompressType)

ReportTradeSummary2ExcelEx (ABook, ASheet, ACol, ARow, AutoCreate)

ReportSystemSummary2ExcelEx (ABook, ASheet, ACol, ARow, AutoCreate)

System Setting Properties

System settings can only be changed before the first order is placed through the system on a chart unless mentioned otherwise. These parameters control the overall behavior of the trading system and can affect the historical results significantly.

For system settings that affects portfolio system testing, please refer to ***Portfolio System Setting Properties*** (on page 1601).

ClosePositionEOD

For intraday trading system, `ClosePositionEOD` is used to control whether a system will close all open positions at the end of the trading day automatically.

CommissionType

There are two basic types of commission structure supported by the trading system object.

`cmFlatRate` For each transaction to take place, a fixed commission amount will be taken from the cash balance.

`cmPerUnit` For each transaction, a fixed base commission plus a per unit charge will be taken from the cash balance.

CommissionBaseAmount

`CommissionBaseAmount` defines the base amount of commission. This property is used in both `cmFlatRate` and `cmPerUnit` calculations.

CommissionPerUnit

`CommissionPerUnit` defines the per unit commission amount. This property is used in `cmPerUnit` calculations.

DefaultOrderSize

Returns the default order size of the symbol. Default order size is set either in Symbol Info Manager or when applying the trading system.

FillType

`FillType` controls how a trade is filled. By changing the fill behavior you can get a sense of sensitivity of the trading system related to entry conditions. Better trading systems usually are insensitive to fill results, even under the worst condition.

`ftWorst` use the worst possible price of the next bar as the filled price

`ftAverage` use the average of the next bar as the filled price

`ftExact` use the exact price based on the type of order as the filled price

`ftLimitedWorst` Similar to `ftWorst`, except the worst case is limited by the value specified by `Slippage`.

`ftBidLimitedWorst`

`ftAskLimitedWorst` Forex specific. Worst case is estimated using spread. If the Forex trade data comes from bid, use Bid Limited Worst. If the Forex trade data comes from ask, use Ask Limited Worst.

InitialCapital

`InitialCapital` sets or returns the initial capital available for trading. This parameter affects the trade summary report calculations and can be used in a trading system for calculating your order size dynamically

InterestRate

`InterestRate` is the percentage value of the risk-free interest rate that you can collect per year. For example, `InterestRate` at 5 is 5% per year. This parameter is used for the calculation of the Sharpe Ratio.

MarginType

`MarginType` controls how `Trade` object handle savailability of cash and the number of shares or contracts that you can trade at any single point in time.

`mtCash` use the face value of the positions in cash requirement calculation

`mtPercent` use a percentage (MarginValue) of the face value of the positions, usually appliable to marginable stocks

`mtAmount` use an exact amount (MarginValue) per contract of the positions, usually applicable to future contracts

MarginValue

`MarginValue` is used together with `MarginType` to determine the cash requirement of the positions being held.

MinTickSize

`MinTickSize` specifies the minimum change in the symbol. For example stocks are now moving at a minimum of $0.01, while the E-mini SP 500 future has a minimum move of 0.25 point. This parameter affects how `Trade` object fill your trades.

OrderPlacementUsingTradeSimOnly

False by default. If set to true, all orders placed will be sent to Trade Simulator for execution, regardless of Order Interface settings.

Use this property when NeoTicker® is set up to send real-life orders to your broker, but your system is not ready to do that yet.

PriceMultiple

`PriceMultiple` equals to 1 for stocks. For trading instruments like commodities or index futures, the price multiples are different. This parameter affects the way trade summary is calculated.

ScaleOrderSize

Returns the scale order size of the symbol. Scale order size is set either in Symbol Info Manager or when applying the trading system.

SingleEntryPerDirection

Set `SingleEntryPerDirection` to true if no more than one position at a direction is possible. For example, if the system is already long, no more additional long position will be accepted.

Slippage

This property is applicable only if `FillType` property is set to `ftLimitedWorst`. In this case, the fill will be using the worst possible price or slippage, which ever is less severe.

TimeRangeStart

`TimeRangeStart` is a double property. You can assign a specific date to this property to enforce a starting date time for the trade object to accept order placement. It is used if and only if UseTimeRange is set to true.

TimeRangeEnd

`TimeRangeEnd` is a double property. You can assign a specific date to this property to enforce a finishing datetime for the trade object to stop accepting order placement. It is used if and only if UseTimeRange is set to true.

UseTimeRange

`UseTimeRange` is a boolean property. It is default to false. When set to true, you can control the trade object to allow orders to be placed within the date range you have specified using TimeRangeStart and TimeRangeEnd.

SystemStats Object

`Trade.SystemStats` returns the SystemStats object. This object is for querying varies trading system statistics. SystemStats stores the same set of statistics as System Performance Viewer and TradingSystemStats object in OLE automation.

See ***TradingSystemStats Object*** (on page 677) for reference.

System Status Properties

CashUsage, CashUsageEx (IssueId)

CashUsage returns the cash used to enter the current position.

CashUsageEx is the portfolio version. It returns the cash used to enter the current position for IssueId, where IssueId is from 1 to DataSeries.Count, pointing to DataSeries [IssueId].

CurrentEquity

CurrentEquity returns the current equity level. It is based on the total cash on hand at the moment combined with the open position profit/loss.

CurrentCash

CurrentCash returns the cash on hand based on the current equity level taking out the cash requirement of all the positions.

CurrentCashExcludingShortPositions

CurrentCashExcluidngShortPositions returns the cash on hand similar to CurrentCash property, except that cash received form short positions are excluded.

HasTradedThisBar, HasTradedThisBarEx (IssueId)

HasTradedThisBar returns true if a trade has happened in this bar.

HasTradedThisBarEx is the portfolio version. It returns the whether issue represented by IssueId has been traded in this bar, where IssueId is from 1 to DataSeries.Count, pointing to DataSeries [IssueId].

HasTradedToday, HasTradedTodayEx (IssueId)

HasTradedToday returns true if a trade has happened in the day the bar belongs to.

HasTradedTodayEx is the portfolio version. It returns the whether issue represented by IssueId has been traded today, where IssueId is from 1 to DataSeries.Count, pointing to DataSeries [IssueId].

LastTransaction, LastTransactionEx (IssueId)

LastTransaction returns the transaction object of the last transaction. If there is no transaction, an empty pointer is returned.

LastTransactionEx is the portfolio version. It returns the last transaction object of IssueId, where IssueId is from 1 to DataSeries.Count, pointing to DataSeries [IssueId].=

MaxSizeUsingAvailableCash (IssueId)

`MaxSizeUsingAvailableCash` returns the maximum number of shares/contracts that you can take based on available cash on hand. When `IssueId` is set to 0, you are calculating the maximum size for the primary linked series. When `IssueId` is set from 1 to `DataSeries.Count`, then you can calculating the maximum size for `DataSeries [IssueId]`.

MaxSizeAfterClosingPositions (IssueId, CoverId, CoverAll)

`MaxSizeAfterClosingPositions` returns the maximum number of shares/contracts that you can take based on available cash on hand. When `IssueId` is set to 0, you are calculating the maximum size for the primary linked series. When `IssueId` is set from 1 to `DataSeries.Count`, then you can calculating the maximum size for `DataSeries [IssueId]`.

The difference of this method comparing to `MaxSizeUsingAvailableCash` is that if `CoverAll` is set to true, then the parameter `CoverId` is ignored, and the calculation will be based on the assumption that you are going to close out all the current open positions to make available more cash. If `CoverAll` is set to false, while `CoverId` is within valid range of referring to a data series (0 for the primary linked series, 1 to `DataSeries.count` for the data series), then the assumption is that you are going to cover the position held for the specified data series, and those cash is included in the calculation of the maximum size you can take.

Transaction Object

A transaction object encapsulate the detail information of a trade. You can access these detail information through the following properties.

Transaction objects are supporting object of `Trade` object. They are returned by transaction methods of `Trade` object.

UniqueId

Returns an integer representing the unique identification number of the transaction.

OrderId

Returns an integer representing the order that resulted in the execution of this trade.

DateTime

Returns the date and time that this transaction has taken place

LinkId

The underlying data series identifier represents the primary linked series of the indicator. Values greater than 0 represents the underlying series is one of the `DataSeries`

BarsNum

The bar number of the underlying series that the trade took place.

Price

Returns the filled price of the trade.

Size

Returns the filled size of the trade. For stocks it means number of shares.

TransType

Returns the transaction type of the trade. There are two (2) types of transaction.

`ttLong` – a trade resulted from a buy order

`ttShort` – a trade resulted from a sell order

Comment

Returns the comment of the trade.

Commission

Returns the commission paid for this trade.

Symbol

Returns the symbol that this transaction is taken on.

Transaction Properties and Methods

`Trade` object supports the following properties for direct access to detail information of each trade recorded.

TransactionCount

Returns the number of transaction recorded so far by the Trade object.

Transactions [Index]

Returns a specific transaction object located at the Index position. Index ranges from 0 to `TransactionCount` -1.

TransactionById [TransId]

Return the transaction object with the `UniqueId` equals to `TransId`.

TransactionByComment ["comment"]

Returns the last transaction object with the specified comment.

NTLib Object

NeoTicker® provides a rich library of functions for use in all scripting languages. All functions are grouped under the NTLib object for easier access across all scripting and IDL supporting languages.

To use any function listed in this object, you can simply reference it by having NTLib preceding the function call.

For example, you can use the function int2time in VBScript like this,

```
if data1.time (0) <= NTLib.int2time (123000) then
     ...
end if
```

All of the original functions in the built-in function library introduced since version 1.0 are available within NTLib, without the tq_ prefix.

For example, if you have a script using the function tq_ntnow, you can convert it to NTLib.ntnow to get the exact same result in the script.

NTLib replaces the tq functions. Existing tq functions will continue to work, but new library functions will only be added to NTLib.

Trig Functions

function arccos(X: Double): Double;

`arccos` returns the inverse cosine of X. X must be between -1 and 1 inclusive. The returned value will be in the range [0..Pi], in radians.

function arccosh(X: Double): Double;

`arccosh` returns the inverse hyperbolic cosine of X. The value of X must be greater than or equal to 1.

function arcsin(X: Double): Double;

`arcsin` returns the inverse sine of X. X must be between -1 and 1 inclusive. The returned value will be in the range [-Pi/2..Pi/2], in radians.

function arcsinh(X: Double): Double;

`arcsinh` returns the inverse hyperbolic sine of X.

function arctan(X: Double): Double;

`arctan` returns the inverse tangent of X.

function arctanh(X: Double): Double;

`arctanh` returns the inverse hyperbolic tangent of X. The value of X must be between -1 and 1 (inclusive).

function cos(X: Double): Double;

`cos` returns the cosine of the angle X, in radians.

function cosh(X: Double): Double;

Use the `cosh` to calculate the hyperbolic cosine of X.

function sin(X: Double): Double;

The `sin` function returns the sine of the argument. X is a real-type expression. `sin` returns the sine of the angle X in radians.

function sinh(X: Double): Double;

`sinh` calculates the hyperbolic sine of X.

function tan(X: Double): Double;

`tan` returns the tangent of X. `tan(X) = sin(X) / cos(X).`

function tanh(X: Double): Double;

`tanh` calculates the hyperbolic tangent of X.

Conversion Functions

function deg2rad(Degrees: Double): Double;

`deg2rad` converts angles expressed in degrees to the corresponding value in radians, where radians = degrees(pi/180).

function rad2deg(Radians: Double): Double;

`rad2deg` converts angles measured in radians to degrees, where degrees = radians(180/pi).

function double2integer(X: Double): Integer;

`double2integer` converts double number to integer.

function integer2double(X: Integer): Double;

`integer2double` converts integer to double.

Basic Math Functions

function abs(X:Double): Double;

`abs` returns the absolute value of the argument, `X`.

function ceil(X: Double): Double;

`ceil` returns the lowest integer greater than or equal to `X`. For example:

```
NTLib.ceil(-2.8) = -2
NTLib.ceil(2.8) = 3
NTLib.ceil(-1.0) = -1
```

function exp(X: Double): Double;

`exp` returns the value of e raised to the power of `X`, where e is the base of the natural logarithms.

function floor(X: Double): Double;

`floor` returns the highest integer less than or equal to `X`. For example:

```
NTLib.floor(-2.8) = -3
NTLib.floor(2.8) = 2
NTLib.floor(-1.0) = -1
```

function frac(X: Double): Double;

`frac` returns the fractional part of the argument `X`.

`X` is a real-type expression. The result is the fractional part of `X`; that is, `frac(X) = X - floor(X).`

function ln(X: Double): Double;

`ln` returns the natural logarithm `X`.

function log10(X: Double): Double;

`log10` returns the log base 10 of `X`.

function logn(N, X: Double): Double;

`logn` returns the log base `N` of `X`.

function max(A ,B: Double): Double;

`max` compares two numeric values. `max` returns the greater value of the two.

function min(A ,B: Double): Double;

`min` compares two numeric values. `min` returns the smaller value of the two.

function pi: Double;

Use `pi` in mathematical calculations that require pi, the ratio of a circle's circumference to its diameter. Pi is approximated as 3.1415926535897932385.

function power(Base, Exponent: Double): Double;

`power` raises `Base` to any power.

function round(X: Double): Double;

`round` rounds a real-type value.

`X` is a real-type expression. `round` returns a value that is the value of `X` rounded to the nearest whole number. If `X` is exactly halfway between two whole numbers, the result is always the even number.

function sqrt(X: Double): Double;

`sqrt` returns the square root of `X`. `X` is a floating point expression.

function trunc(X: Double): Double;

`trunc` truncates a real-type value towards zero.

function random(Range: Double): Double;

`random` returns a random whole number within the range `0 <= X <= Range.` `Range` is rounded to the nearest whole number before it is used. If `Range` equals to 0, the result is a random number within the range `0 <= X < 1.`

To initialize the random number generator, add a single call `randomize` or `randseed` before making any calls to `random`.

procedure randseed(X: double);

`randseed` assigns a specific value `X` as a random seed to the random number generator. This is useful for applications that deal with data statistics and simulations that need reproducible results.

function randg(Mean, StdDev: Double): Double;

`randg` produces random numbers with Gaussian distribution about the Mean. This is useful for simulating data with sampling errors and expected deviations from the Mean.

procedure randomize;

`randomize` initializes the built-in random number generator with a random value (obtained from the system clock).

Do not combine the call to `randomize` in a loop with calls to the `random` function. Typically, `randomize` is called only once, before all calls to `random`.

Random Functions

function RandomStreamCount : integer;

Returns the number of random streams available.

procedure RandomStreamRandomize(const StreamID : integer);

Initializes a random stream randomly.

procedure RandomStreamSeed(const StreamID : integer; const ASeed : integer);

Initializes a random stream with the provided random seed.

function RandomStreamDouble(const StreamID : integer) : double;

Returns a random double x, where 0 <= x < 1, using the specified random stream.

function RandomStreamInteger(const StreamID : integer; const AInteger : integer) : integer;

Returns a random integer n, where 0 <= n < AInteger, using the specified random stream.

String Conversion Routines

function double2str(Value: Double): String;

`double2str` converts the floating-point value given by `Value` to its string representation. The conversion uses general number format with 15 significant digits.

function str2double(const S: String): Double;

Use `str2double` to convert a string, S, to a floating-point value. S must consist of an optional sign (+ or -), a string of digits with an optional decimal point, and an optional mantissa. The mantissa consists of 'E' or 'e' followed by an optional sign (+ or -) and a whole number. Leading and trailing blanks are ignored.

function str2integer(const S: String): Integer;

`str2integer` converts the string S, which represents an integer-type number in either decimal or hexadecimal notation, into a number.

function integer2str(Value: Integer): String;

`integer2str` converts an integer into a string containing the decimal representation of that number.

function format (const Format: String; const Args: array of const): String;

This function returns a string that formats the open array of constants `Args` using the formatting string `Format`. The format string is similar to C-style format string.

Although `format` works in VBScript and JavaScript, its use in these scripting languages are not recommended. It is because these two languages are loosely typed, where as `format` requires you to specify the exact type of each variable in order for it to work.

For example, the following statements will all produce the string '10 hello 3.5'

```
VBScript      - NTLib.format("%d %s %f", array(10, "hello",
3.5))
JavaScript    - NTLib.format("%d %s %F", [10, "hello", 3.5]);
Delphi Script - NTLib.format('%d %s %f', [10, 'hello', 3.5]);
```

function formatdatetime (const Format : String; DateTime : Double): String;

This function returns a string based on the date time value formatted by the formatting string `Format`. `DateTime` is a Windows internal data time value. In the format string, this function uses y for years, m for months, d for days, h for hours, n for minutes and s for seconds.

For example, the following statements will all produce the string '2001 01 14', assuming `datetime` represents January 14th, 2001:

```
VBScript      - NTLib.formatdatetime("yyyy mm dd", datetime)
JavaScript    - NTLib.formatdatetime("yyyy mm dd", datetime);
Delphi Script - NTLib.formatdatetime('yyyy mm dd', datetime);
```

function formatdouble (const Format : String; Value : Double) : String;

This function formats a floating point value into a string. For example, the following statements will all produce the string ' 1,000.350':

```
VBScript      - NTLib.formatdouble("###,###.000", 1000.35)
JavaScript    - NTLib.formatdouble("###,###.000", 1000.35);
Delphi Script - NTLib.formatdouble('###,###.000', 1000.35);
```

function datetime2str (DateTime : Double) : String;

This function returns a string based on the date time value `DateTime` in Windows short date format.

function str2datetime (const S : String) : Double;

This function returns a double value in internal date time format representing the date time in the string `S`.

function date2str (DateTime : Double) : String;

This function returns a string based on the date value `DateTime` in Windows short date format.

function str2date (const S : String) : Double;

This function returns a double value in internal date format representing the date in the string `S`.

function time2str (DateTime : Double) : String;

This function returns a string based on the time value `DateTime` in Windows time format.

function str2time (const S : String) : Double;

This function returns a double value in internal time format representing the time in the string `S`.

function str2color (const S: String) : Double;

Convert string into color values. If the string is not recognized the default color of black will be returned. Valid string as follows:

```
clAqua
clBlack
clBlue
clDkGray
clFuchsia
clGray
clGreen
clLime
clLtGray
clMaroon
clNavy
clOlive
clPurple
clRed
```

```
clSilver
clTeal
clWhite
clYellow
```

String Handling Routines

function copy(S: String; Index: Integer; Count: Integer): String;

`S` is a string. `Index` and `Count` are integer-type expressions. `copy` returns a substring containing `Count` characters.

If `Index` is larger than the length of `S`, `copy` returns an empty string.

If `Count` specifies more characters than are available, only the characters from `S[Index]` to the end of `S` are returned.

function pos(Substr: String; S: String): Double;

`pos` searches for a substring, `Substr`, in a string, `S`. `Substr` and `S` are string-type expressions.

`pos` searches for `Substr` within `S` and returns an integer value that is the index of the first character of `Substr` within `S`. `Pos` is case-sensitive. If `Substr` is not found, `pos` returns zero.

function length(S: String): Double;

`length` returns the number of characters actually used in the string. `S` is a string expression.

function lowercase(const S: String): String;

`lowercase` returns a string with the same text as the string passed in `S`, but with all the letters converted to lower case.

function trim(const S: String): String;

`trim` removes leading and trailing spaces as well as control characters from the given string `S`.

function trimleft(const S: String): String;

`trimleft` returns a copy of the string `S` with leading spaces and control characters removed.

function trimright(const S: String): String;

`trimright` returns a copy of the string `S` with trailing spaces and control characters removed.

function uppercase(const S: String): String;

`uppercase` returns a copy of the string `S`, with the same text but with all the letters between 'a' and 'z' converted to upper case.

Date Time Functions

function date2int (D : Double) : Integer;

`date2int` converts date value `D` in internal format into an integer representing the date in yyyymmdd format.

function day (D : Double): Integer;

`day` extracts the day of month information from the date value `D`. `D` is in internal format.

function hour (T : Double): Integer;

`hour` extracts the hour information from the time value `T`. `T` is in internal format.

function int2date (I : Integer): Double;

`int2date` converts an integer `I` of format yyyymmdd into a date value of internal format.

function int2time (I : Integer): Double;

`int2time` converts an integer `I` of format hhmmss into a time value of internal format.

function ntnow: Double;

Returns current date time in the time zone NeoTicker® assigned to under server setup.

function minute (T : Double): Integer;

`minute` extracts the minute information from the time value `T`. `T` is in internal format.

function month (D : Double): Integer;

`month` extracts the month of year information from the date value `D`. `D` is in internal format.

function now: Double;

`now` returns the current date time in internal format.

function second (T : Double): Integer;

`second` extracts the second information from the time value `T`. `T` is in internal format.

function secondssincemidnight (T: Double) : Integer;

`secondsincemidnight` converts time value `T` in internal format into an integer representing the number of seconds since midnight of same day. The date information is discarded.

function time2int (T : Double) : Integer;

`time2int` converts time value `T` in internal format into an integer representing the time in hhmmss format.

function today: Double;

`today` returns the current date in internal format.

function weekday (D : Double): Integer;

`weekday` extracts the day of week information from the date value `D`. `D` is in internal format.

function year (D : Double): Integer;

`year` extracts the year information from the date value `D`. `D` is in internal format.

File Functions

function ApplicationDirectory : string

Returns the absolute path of NeoTicker® installation.

function DirectoryExists(DirectoryName : string) : boolean

Returns true if a directory exists.

function DirectoryFileList(DirectoryName : string; var FileList : Variant; var fileCount : integer) : boolean

Returns a list of files in a directory and the file count. The filename returned is stored in a variant array in absolute path. This function will go into all subdirectories recursively to obtain all files.

function FileExists(Filename : string) : boolean

Returns true if a file exists.

function IndicatorDirectory : string

Returns the indicator directory. This is usually the `Indicator` directory in the installation directory, but the location can be modified by the user.

User Information Routines

In general, functions in this section are useful only if you are developing a highly customized protection scheme.

If you are a third party developer, you should consider using NeoTicker® real-time version which includes a sophisticated suite of protection tools. Products developed in real-time version can be easily and securely deployed to both real-time and EOD version of NeoTicker®. For more information, see ***Third Party Developer Guide*** (see "Guide To Protection" on page 1659) in the real-time version documentation

function userid : string;

`userid` returns the unique user id of the NeoTicker® in use.

function blockid : string;

`blockid` returns the unique hardware id of the NeoTicker® in use. This function does not work in the EOD version or demo version.

The encode/decode pair functions provided here is useful for third party developers to create protected script indicators that say expire by certain time period, or, say, enable or disable certain features of the indicator.

Please be advised that while all current real-time version of NeoTicker® (except demo version) has unique hardware id, this is not guaranteed and can change in the future.

The encryption algorithm used in `encode` and `decode` share the same password to encode and decode the data.

function IsEOD : boolean;

Returns true if the running application is NeoTicker® EOD.

function IsDemo : boolean;

Returns true if the running application is NeoTicker® demo version. This function does not work in the EOD version.

function IsLease : boolean;

Returns true if the running application is NeoTicker® with a lease license. This function does not work in the EOD version.

function IsFull : boolean;

Returns true if the running application is NeoTicker® with a permanent license. This function does not work in the EOD version.

function IsNetwork : boolean;

Returns true if the running application is NeoTicker® with a network license. This function does not work in the EOD version.

function encode (InputString, PasswordString) : String;

Returns an encoded version of `InputString` based on encryption key string `PasswordString`.

function decode (InputString, PasswordString) : String;

Returns the decoded version of `InputString` based on encryption key string `PasswordString`.

Miscellaneous Routines

procedure ClearScriptErrorLog;

Clear all messages from the Script Error Log window.

procedure Debug (Value : Variant);

Debug will send the variant `Value` to the script error log window. If the Script Error Log window is set to append logs to file, then you have a complete error tracking tool for tracing the value of a variable.

Remember that using debug and/or using the append logs to file feature in the Script Error Log could slow down NeoTicker®. Thus use it only when necessary.

function GetSymbolSpecs(const Symbol : String; var PriceMultiple : double; var MinTickSize : double; var DefaultSize : integer) : boolean;

Queries ***Symbol Info Manager*** (on page 921) about a symbol. Price multiple, minimum tick size and default order size are returned. Function returns true if symbol definition exists in Symbol Info Manager.

procedure GetUserFields(const Symbol; var Field1, Field2 : variant) : boolean

Retrieves the user fields of a symbol. User fields are set using Symbol Info Manager. See Symbol Info Manager, Field Description.

procedure playsound (Filename : string);

`Playsound` will play the sound file specified by `Filename`. Full path information must be provided.

procedure UDSUpdateWithVolume (Symbol : string; Price : double; Volume : integer);

`UDSUpdateWithVolume` will update the User Define Symbol of External type, `Symbol`, with the latest price set to `Price`, and trade volume set to `Volume`.

Script Editor Reference

Script editor is an integrated editor for the development of indicators for use in NeoTicker®.

You can start the script editor by choosing **Program>Script Editor>New** from the main window.

Script editor works very much like a standard text editor. Extra feature is build-in to support its use within NeoTicker®.

Features Highlight

- Full featured text editor with powerful find and replace, and unlimited undos and redos.
- Supports keyword highlight based on the scripting language .
- Supports built-in verification and dynamic installation into NeoTicker®. You can develop a script, test, install and run it without leaving NeoTicker®.
- You can open multiple script editors. If you open more than one editor on the same file, only the first opened editor can save changes to the file. The other editors are read-only and are used only for browsing.
- Supports password protection and encryption of all scripts.

Edit Area (Pane 1, Pane 2)

This is the area where you can edit your scripts. Pane 1 and pane 2 display the same script, but at different positions. The panes are handy if you work with a large script. You can adjust the behavior with the **Visual** menu.

If you have both panes visible, you can use the pane splitter to adjust the vertical space of the panes. If the **Self-adjusted Splitter** option under the **View** menu is enabled, then the splitter is scaled automatically in ratio to the editor window.

Under the **View** menu, you can turn on line number display for each pane.

Script Headers

Each script has a header associated with it. The header specifies things such as default plot color, number of user parameters, etc.

Normally, the header is hidden, as you are expected not to change the header manually. You can turn on the header by choosing **Visual>Show Script Header**. Showing the header is useful when you want to cut and paste script from one editor to another.

Indicator Menu

The **Indicator** menu contains the commands for setting up and installing scripts.

Item	Function
Setup	Opens the Indicator Specification window for easy indicator definition change
Verify	Verifies the script based on the script language. If an error is encountered, the edit area will jump to the position where the error is found
Install	Installs the script file into the indicator library of NeoTicker®
Disable Now	Disable an indicator for debugging purpose. It will unload its designated external file and ActiveX components

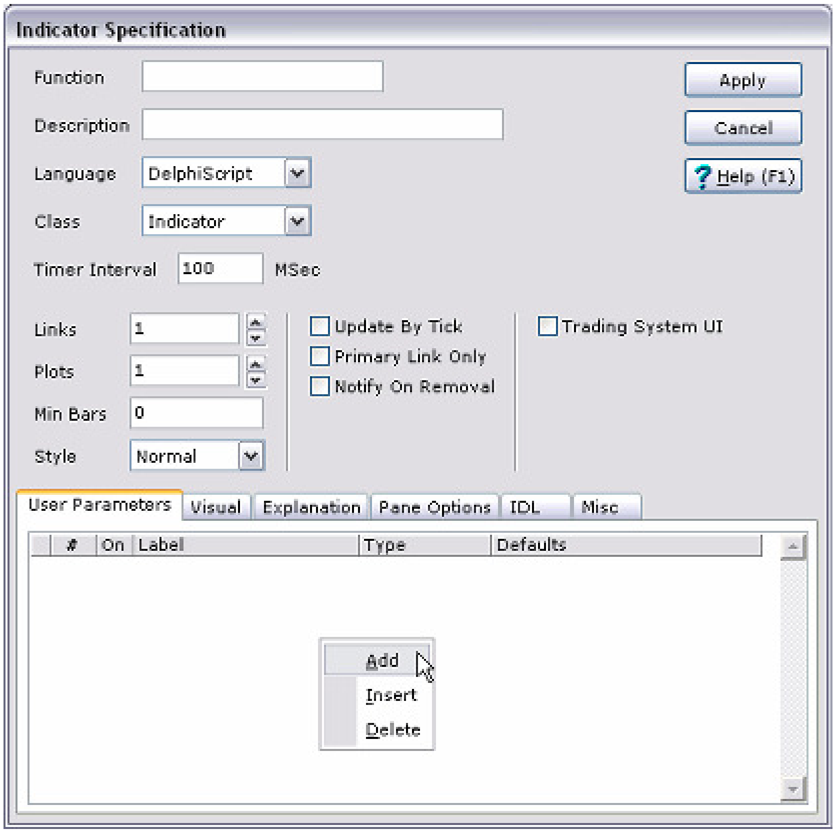
Indicator Specification

This window allows you to edit the indicator specification header of the script.

The indicator specification header is a section in your script that describes how the indicator works with NeoTicker®. This section is common to all types of scripting languages and will not affect your verification process.

You should not modify the indicator specification header directly. Using the Indicator Specification Window to change the header is recommended.

Choose **Setup** from the **Indicator** menu to set up a script as an indicator.


Indicator Specification

Function

Description

Language: DelphiScript

Class: Indicator

Timer Interval: 100 MSec

Links: 1

Plots: 1

Min Bars: 0

Style: Normal

□ Update By Tick
□ Primary Link Only
□ Notify On Removal
□ Trading System UI

#	On	Label	Type	Defaults

Below is the description of the settings available for designing an indicator. Items marked with an asterisk(*) can be overridden by user at run-time.

Top Area

Setting	Function

Function
The function name that NeoTicker® has to call within the script to obtain the indicator value. This is the short name of the indicator.

Description
The description of the function which will appear in the indicator selection area when you try to add the indicator to a chart

Language
The language that you use in writing the script.

Class
Whether it is an normal indicator, updated by timer or a command. See ***Indicator Updated by Timer*** (on page 1467).

Timer Interval
When indicator is updated by timer, the update interval.

Links
The number of data source this indicator will take

Min Bars
The default number of bars the indicator needed before starting its calculation*

Plots
The number of plots in the indicator

Update By Tick
Whether the indicator is updated by tick*

Primary Link Only
When checked, indicator is updated only when there is changes in primary link. See ***Updated by Primary Link Only*** (on page 1478).

Notify On Removal
A removal notice will be sent to indicator for finalization when the indicator is deleted or recalculated. See ***Initialization and Finalization*** (on page 1468).

Trading System UI
Whether the indicator will show trading system setup in the indicator edit window

Style
Special drawing style setting that combines multiple plots into new plot styles*

User Parameters Tab

Lets you add, delete, change user parameters.

Setting	Function
#	Parameter number. This is the same number as the index used by the `ItSelf.Params` property
On	Whether the parameter is enabled
Label	The name you give to this parameter. This label will show up in the Indicator Setup window
Type	Type of parameter. See Parameter Type section below.
Defaults	Default value to be put into the parameter

In Defaults field, you can specify multiple items, separated by vertical slash. The items will be available in a pop up menu to the user of the indicator, for example:

```
None|Percent|Value
```

In Defaults field, for date time parameter, you can use the special keywords `NOW` and `TODAY`. When using this special keywords, the value will be filled in automatically when the indicator is in use.

Parameter Type

A parameter can be of different types, parameter type determines the user interface when user accesses the parameter when the indicator is in use. For parameter type, you have choice of: **string**, **real**, **integer**, **integer.gt.0**, **integer.gt.1**, **date**, **time**, **datetime**, **formula**, **color**. For **date**, **time**, **datetime**, **color**, script editor provides dialogs to help set default values.

Below is the description of the user interface behavior for the different parameter types.

For **string** parameter, the indicator user can enter any string. No checking is performed.

For **real** parameter, the indicator user must enter a floating point number.

For **integer** parameter, the indicator user must enter an integer value.

For **interger.gt.0** parameter, the indicator user must enter an integer value greater than or equal to 0.

For **integer.g1.1** parameter, the indicator user must enter an integer value greater than or equal to 1.

For **date** parameter, the indicator user must enter a valid date. A date chooser dialog is provided to help the user selecting date. For the default value, date is a floating point number. The integral of the date value is the number of days passed since 1899/12/30.

For **time** parameter, the indicator user must enter a valid time. A time chooser dialog is provided to help the user selecting time. For the default value, time is a floating point number. The fraction of the date value is the fraction of a 24-hour day.

For **datetime** parameter, the indicator user must enter a valid date time. A date time chooser is provided to help the user selecting date time. A date time chooser dialog is provided to help the user selecting date time.

For **formula** parameter, the user can enter any string. A formula editor is provided to help the user to enter and validate a formula. For formula parameter, no checking is done on the validity of the formula when the user enters the formula. It's up to the script/IDL to properly use the formula the user enters.

For **color** parameter, the user must enter an integer value representing a color. A color dialog is provided to help the user to select color.

For information on how to interpret the parameter values in script/IDL, refer to ***Param and Params Object*** (on page 1519).

Visual Tab

Lets you modify the settings for each individual plot.

Setting	Function
Plot	The plot number
Name	The plot name
Enabled	Whether the plot is enabled when the indicator is created*
Style	The plot style*
Color	The plot color*
Width	The plot width*

Explanation

Under the explanation tab, you can enter the long explanation of the indicator.

Pane Options

Settings under this tab controls how the indicator is placed in a time chart.

Indicator Placement

This setting tells where to put the indicator. **Smart** setting is the default. Time chart will make the decision where the indicator will be placed based on the value range of the indicator.

Other choices are **Active Pane** (the pane user recently clicks on), **New Pane**, **First Pane**, **Last Pane**, **Specified** (user specified at runtime).

Value Range

This is a hint you give to time chart the value range of the indicator. It is used only if you have **Smart** setting for **Indicator Placement**. The hint is important as time chart need to figure out which pane is most suitable for the indicator, or if a new pane should be created to accommodate the indicator.

IDL Tab

This tab is for specifying indicators implemented using an external programing language. See ***IDL Interface (DLL, ActiveX and .NET indicators)*** (on page 1635).

Misc Tab

This tab is for specifying additional parameters.

It allows you to set a help file for the indicator. See ***Creating Help for Indicator*** (on page 1451).

Drawing Tool Menu

Drawing Tool menu is for specifying a script for use as a customize drawing tool.

Item	Function
Verify	Verifies the script based on the custom drawing tool specifications.
Install As Custom 1-4	Install the current script as one of the custom drawing tool.

Accessing Other Script Editors

All opened script editors are listed under the **Script** menu.

Indicator Wizard

Indicator wizard helps you specify an indicator in formula language.

To open indicator wizard in script editor, choose **Tools>Indicator Wizard**.

Indicator wizard is not applicable if you are writing indicator with scripting language.

Password Protection

This section is obsoleted and is for backward compatibility only.

If you want to protect your scripts for personal use, see ***Protecting Your Work*** (on page 839).

If you want to protect your scripts as a third party developer, see ***Third Party Developer Guide*** (see "Guide To Protection" on page 1659).

NeoTicker® can password protect your scripts from unauthorized access by another user.

The password protected scripts can be loaded into any NeoTicker® for use, but cannot be edited nor read through the script editor nor other regular text editor program.

Important Security Concern

DelphiScript is native to NeoTicker® while VBScript and JavaScript rely on Microsoft's Scripting Technology. This external reliance makes password protecting VBScript and JavaScript technically less secure than password protecting DelphiScript. Therefore, if you require the highest level of security for your scripts, you should develop your scripts in DelphiScript.

Basic Protection

To protect your script, open it using the script editor. Then choose **File>Set Password.**

You cannot use **Set Password** to protect VBScript that have already been encoded using Microsoft's VBScript Encoder. To protect encoded VBScripts, use **Protect File Using Password** instead. This command is discussed in the next section.

Then the Script Password Setup form will show.

Type your password in. Then it will request you to retype the password to confirm, as you may accidentally typed the wrong password in the first place.

If the confirmation is successful, then you will then be prompted to save the changes immediately.

If you choose to save now, the script file will be immediately saved in the protected format.

Protecting a Script File

You can protect a script file on your hard drive by choosing **File>Protect File Using Password**.

In addition to the password, NeoTicker® will ask you for an input file and an output file.

This command is useful when you want to keep an original and an encrypted version of a script file. Also, you should use this command if you want to password protect a VBScript that has been encoded using Microsoft's VBScript encoder.

Opening A Protected Script

When you open a protected script, you will be prompted for the password.

If the password match that of the protected script, you will then be able to edit the script.

Changing Password

To change password choose **File>Set Password**. You will be prompted for the current password. If that is successful, you will then go through the sequence of entering a new password.

Disable Password Protection

To disable password protection, just change the password to nothing. Then the script will be saved in non-protected format.

Full Protection Scheme Consideration

This section is not applicable to NeoTicker® EOD.

The NeoTicker®'s password protection scheme protect your scripts from being modified but does not stop the user from copying the script file to install that onto another NeoTicker®.

To fully protect your script from being used by another user or to control the usage of a script. You can utilize the built-in function of `NTLib.userid` and `NTLib.blockid` in your script to control the script from being accessed by non-authorized NeoTicker®.

You can also add usage control based on pass code entered through a parameter entry. The `NTLib.encode` and `NTLib.decode` methods can help you encoding and decoding passcode for a simple but effective protection.

Each NeoTicker® has a unique user id and hardware protection key id. This information can be used in the script as a filter to stop the script from functioning if the block id does not match.

If you use DLL or other externally programmed code, then you can even protect your work with more complex protection scheme like having your own hardware key protection.

IDL Interface (DLL, ActiveX and .NET indicators)

IDL Overview

IDL interface of NeoTicker® is designed for high-end users and 3rd party developers to develop indicators using programming tools like Visual Studio, Visual C++, Visual Basic, Delphi, C++ Builder, etc. with direct access to NeoTicker®'s indicator object model. Indicator scripts written for NeoTicker® can be cut and pasted into these development environments and compile with minimal code change. The net result is that your indicator scripts will then be able to execute at native code speed.

To use the IDL interface you need a development tool that supports ActiveX / OLE or .NET interface. We support the following tools:

- Microsoft Visual C#
- Microsoft Visual Basic
- Microsoft Visual C++
- Borland Delphi

Your indicator will compile into one of:

- Standard DLL
- ActiveX DLL (Visual Basic only)
- .NET specific DLL (all .NET 2.0 languages)

If possible, you should compile the indicator into standard DLL or early binding ActiveX DLL. Late binding ActiveX DLL runs slower. .NET DLL run slowest.

Sample Projects

The easiest way to get started is to make a copy of the example projects provided, and start from there.

They are stored in the Samples directory (e.g. `C:\Program Files\TickQuest\NeoTicker4\Samples`).

How It Works

An IDL indicator have two parts: Indicator Description File (.IDL) and DLL file.

Indicator Description File

Indicator Description File is like an indicator script. It describes everything about the indicator (e.g. parameters, pane placement, etc) except how the indicator does its calculation (the calculation part is handled by the DLL file). The Indicator Description File also specifies the location of the DLL file.

You can use Script Editor to create Indicator Description File:

1 From Main Window, choose **Program>Script Editor>New** to open Script Editor.

2 Choose **Indicator>Setup** in the script editor.

3 You will need to specific the number of plots, the parameters, etc. This part is identical to setting up a script. You can refer to ***Creating an Indicator, Example 1*** (see "Tutorial: Creating an Indicator, Example 1" on page 1369) for an example. Refer to ***Indicator Specification*** (on page 1627) for the reference on what each setting means.

4 In the indicator setup, set the **Language** field to IDL

5 Under the **IDL** tab, specify the DLL location under the **External Filename** field. The DLL location is either an absolute path, or simply a file name. For the latter, the DLL file must be located in `indicator` directory in the NeoTicker® installation.

6 For Visual Basic (non .NET), under the **IDL** tab, instead of specifying the DLL location, specify the ActiveX object class name under the **ActiveX Class** field.

7 For .NET languages, under the **IDL** tab, instead of specifying the DLL location, specify the **.NET File** location. You will need to provide the .NET class. The format is `Namespace.MainIDLClass. MainIDLClass` is the class that implements the indicator and NameSpace is the name space `that contains the class.`

Early vs. Late Binding

For ActiveX and .NET indicators, you should always turn on **Early Binding**. Late binding is for back compatibility only. For early binding indicators, the class that implements the indicator must inherit the `IIDLIndicator` interface.

Function Entry Point and Interface

The entry point for function is:

- `idlcallex` for standard DLL and early binding ActiveX and .NET DLL.
- (for backward compatibility only) `idl_call` for standard DLL and late binding ActiveX and .NET DLL.

The following example illustrates how you define `idlcallex` in Delphi in the IDL indicator. Exact function prototype is language dependent. Consult the sample projects for examples in other languages.

```
function idlcallex(NTIndicatorObjects : INTIndicatorObjects) :
double; stdcall;
```

The object `NTIndicatorObjects` is the main interface to NeoTicker® objects. You can access all NeoTicker® objects as properties under `NTIndicatorObjects`. For example the equivalent to `ItSelf` object in script is `NTIndicatorObjects.ItSelf` object in IDL.

The entry function `idlcallex` is introduced in NeoTicker® 4.1. IDL indicators written prior to 4.1 use a different interface which is compatible with current version of NeoTicker®. There is no need to recompile the IDL indicator. However, we suggest you to convert actively developing projects to using `idlcallex` so you have access to the latest NeoTicker® objects.

DLL File

You need to compile your indicator code into a DLL file that NeoTicker® can load.

If you are using Visual Basic, you need to register your ActiveX class before NeoTicker® can recognize your DLL file.

Calling Other Indicators and Compress Series Performance Issues

In IDL, it is much more efficient to use unique id's that are available in `ItSelf.MakeIndicatorEx`, `ItSelf.UpdateIndicatorEx`, `ItSelf.IndicatorEx`, `ItSelf.CompressSeriesEx`, `ItSelf.UpdateCompressSeriesEx`. See ***Using Other Indicators*** (on page 1479) and ***Creating Higher Time Frame Series*** (on page 1452).

Special Considerations for Visual Basic (non .NET)

Registration of an ActiveX DLL

To make the ActiveX object in an ActiveX DLL available to NeoTicker®, you need to register the ActiveX DLL using the command regsvr32.

To register an ActiveX DLL:

1. Open a command prompt window (for Windows XP) or MS DOS window (for Windows NT, 2000).
2. Switch to the proper directory with the command cd (e.g. your DLL is copied to the NeoTicker® directory), e.g.

   ```
   cd c:\Program Files\TickQuest\NeoTicker3\indicator
   ```
3. Type the following command (assuming your ActiveX DLL is called `MyIndicator.DLL`)

```
regsvr32 MyIndicator.DLL
```

4 Windows should pop up a message window saying that the registration is completed successfully.

To unregister an ActiveX DLL:

1 Open a command prompt window (for Windows XP) or MS DOS window (for Windows NT, 2000).

2 Switch to the proper directory with the command cd (e.g. your DLL is copied to the NeoTicker® directory), e.g.

```
cd c:\Program Files\TickQuest\NeoTicker3\indicator
```

3 Type the following command

```
regsvr32 /u MyIndicator.DLL
```

4 Windows should pop up a message window saying that the un-registration is done.

Special Considerations for Delphi

ShareMem

If you plan to load and unload DLL for debugging purpose, you must include the ShareMem unit in your code. Otherwise the DLL will not be successfully unloaded and can cause memory corruption problems.

Including ShareMem unit is optional for DLL's that are intended for deployment because unloading is not a common user operation.

Inline Array vs. Variant Array

In Delphi Script, you can create inline array with the constructor [ ]. When you compile in Delphi, this same construct would be recognized as set operator. To bypass this problem, use the `VarArrayOf` function.

For example, in Delphi Script you can write a makeindicator call as follows,

```
MyFunc := Itself.MakeIndicator ('a', 'average', ['1'],
[param1.str]);
```

In Delphi, you will need to modify it to,

```
MyFunc := NTIndicatorObjects.Itself.MakeIndicator ('a',
'average', VarArrayOf (['1']), VarArrayof ([param1.str]));
```

Types

When you declare a variable in Delphi Script, you do not have to provide a type. Type is mandatory in Delphi. The sections below describe how to declare variable with appropriate types when you use Delphi.

Declaration of Nested Indicators

For the example above, the variable MyFunc should be declared as INestedIndicator so that it has the same type as the returning indicator series.

Declaration of Data Series

To access data series like those resulted from property calls CompressSeries, DataSeries.items [1], etc. you need to declare the receiving variable as IDataObj so that you can access to all the data series properties and methods.

For example,

```
var my45minbar : IDataObj;
...
my45minbar := NTIndicatorObjects.Itself.compressseries ('a',
'1', ppMin, 45);
```

Declaration of Orders

To access individual property of an order there is the property NTIndicatorObjects.trade.orders [N].

If you need to assign this property to a variable, the variable must be declared as IOrderObj.

Declaration of Closed Positions

To access individual closed position data there are the properties NTIndicatorObjects.trade.closedpositions [N] and NTIndicatorObjects.trade.closedpositionsex [N].

If you need to assign this property to a variable, the variable must be declared as IPositionObj.

Special Considerations for Visual C++

Fundamentally there is no different to program an IDL indicator using Visual C++ compared to other languages.

The main issue - you have to manually construct variant arrays in Visual C++. Variant arrays are required by some NeoTicker® methods as parameters. For example, the `MakeIndicator` method uses variant arrays.

To create an variant array as a parameter:

1. Use SAFEARRAY functions to create a safe array of VT_BSTR.
2. Create a SafeArrayAccessData for assigning elements into the safe array.
3. For each element, allocate BSTR strings to assign into the safe array.
4. Create a variant array using the safe array.
5. Call the NeoTicker® method.
6. Free SafeArrayAccessData.
7. Clear the variant array.

NeoTicker® comes with Visual C++ examples to get you started. You can read the code of the TEMA C++ example to see how the above is implemented.

Special Considerations for .NET

.NET DLL's do not require ActiveX style registration.

Porting Scripts to IDL

NeoTicker® objects are identical for scripts and IDL. Porting a script involves no change in your execution logic, you only need to take care of the syntactic differences and the interface.

NTIndicatorObjects

The main object NeoTicker® passes to IDL is `NTIndicatorObjects`. All NeoTicker® objects are properties of this object. For example:

- `ItSelf` in script translates to `NTIndicatorObjects.ItSelf` in IDL.
- `Trade` in script translates to `NTIndicatorObjects.Trade` in IDL.
- `Heap` in script translates to `NTIndicatorObjects.Heap` in IDL, and so on.

Porting VBScript

If you develop your indicator using VBScript, you can use the sample Visual Basic project provided as a starting point.

1. Make a copy of the sample Visual Basic IDL project.
2. For the entry function of the script (function name defined by the **Function** field in the script's setup), copy the body of the function into the `idlcallex` function of the Visual Basic project.
3. If you have supporting functions, copy them into the Visual Basic project.
4. Translate all NeoTicker® objects to under `NTIndicatorObjects`.
5. Compile and correct any problems.
6. Register the ActiveX class of the Visual Basic project.
7. Define an IDL indicator in NeoTicker® that calls the ActiveX class. See ***Overview*** (see "IDL Overview" on page 1635) for more information.

Porting Delphi Script

If you develop your indicator using Delphi Script, you can use the sample Delphi IDL project provided as a starting point.

1. Make a copy of the sample Delphi IDL project.
2. For the entry function of the script (function name defined by the **Function** field in the script's setup), copy the body of the function into the `idlcallex` function of the Delphi project.
3. If you have supporting functions, copy them into the Delphi project.
4. Translate all NeoTicker® objects to under `NTIndicatorObjects`.

5 Compile and correct any problems.

6 Define an IDL indicator in NeoTicker® that calls the DLL compiled by the Delphi project . See ***Overview*** (see "IDL Overview" on page 1635) for more information.

External Interface

Overview

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

External interface of NeoTicker® is designed for high-end users and 3rd party developers to develop tools to work with NeoTicker®. External interface allows developers to call external programs and libraries from a NeoTicker® indicator script. The metaphor is simple. n the NeoTicker® indicator script, NeoTicker® sends data to the external object for processing and the external object returns the result to the NeoTicker® script.

Do not mix up external interface with IDL interface which allows developers to implement indicators with standard development tools like Visual Basic, VC++, Delphi, etc. directly. Refer to the IDL Interface section for more details.

NeoTicker®'s external interface is extremely flexible. It can access external objects from DLL's to full feature applications. Thus program interface is useful for:

- Graphical applications for specialized visualization
- Wrappers to call DLL's developed for other trading applications
- Automatic order entry system

To effectively use the external interface, users are expected to have throughout understand of scripting language in NeoTicker® and is proficient in at least one regular Windows programming environment like Visual Basic, C++, or Delphi. Familiar with ActiveX programming will make it easier to understand the information in this section.

IDL vs. External Interface

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

IDL provides all the programming objects to the external program. Thus IDL is suitable for indicator and trading system programming.

External interface relies on ActiveX to communicate to the external program. It is suitable if you require interface flexibility.

External Interface Types

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

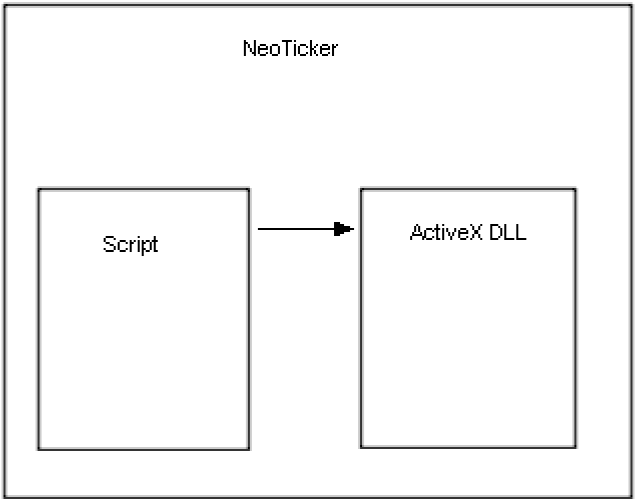
NeoTicker® uses Microsoft's ActiveX technology to communicate with external objects. NeoTicker® will act as an ActiveX client and the external object will act as the ActiveX interface to the actual server, be it a DLL or application.

When designing the ActiveX server for NeoTicker®, you have two choices. You can design an ActiveX DLL or you can design an external program that is an ActiveX EXE. The ways to access ActiveX DLL and ActiveX EXE are identical in NeoTicker® script. However, because ActiveX DLL and ActiveX EXE differ in their implementation and the way they are connected to NeoTicker®, they have different strength and weakness as external objects.

ActiveX DLL is Microsoft's new standard of DLL. Unlike legacy DLL, ActiveX DLL can be registered as an ActiveX server. Therefore, ActiveX DLL is a specialized type of ActiveX server. ActiveX DLL resides in NeoTicker®'s memory so it offers performance advantage over general ActiveX server. In general, use ActiveX DLL when:

- You are developing a new or existing DLL and can change the code to make the DLL an ActiveX DLL
- You do not need visual objects in the ActiveX DLL. It is not safe to create visual objects in ActiveX DLL
- You require the performance advantage of ActiveX DLL

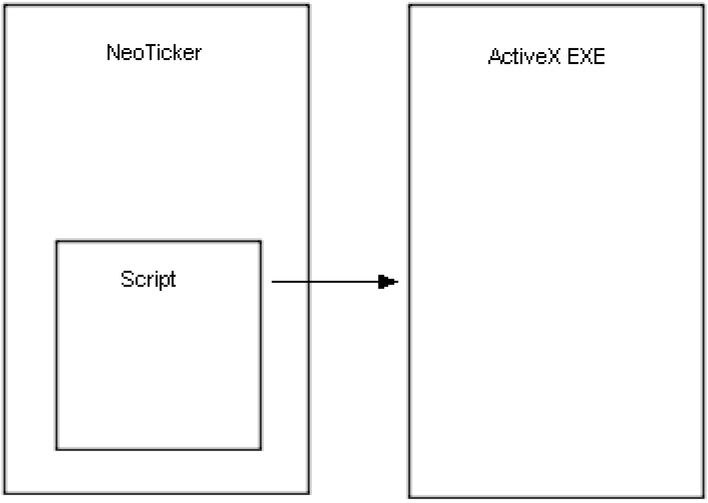
The following figure illustrates NeoTicker®'s relationship with an ActiveX DLL:



ActiveX EXE is an external program that supports an ActiveX interface. Because an external program has its own memory space, this type of external interface offers maximum flexibility. In general, use ActiveX EXE when:

- You require visual objects. For example, your application is a specialized visualization tool
- The construction your application is complex. For example, if extra DLL's are required
- You require advanced protection scheme for proprietary indicators or trading systems

The following figure illustrates NeoTicker®'s relationship with an ActiveX EXE:



Communicate with External Objects

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

NeoTicker® scripts expect the external objects to be ActiveX servers. A script will call exposed methods in the ActiveX server to execute an operation.

Data communication is handled by passing parameters to the exposed method. The parameter type must be COM safe. When possible, you should pass result from an Object as variants. The parameters passed to an Object must be variants (for C++ and Delphi, pointers to variant). Variants are desirable because they are often more efficient in ActiveX communication and type mismatch can cause serious difficulties when debugging.

NeoTicker® provides the methods `MakeArray`, `MakeArrayEx`, `MakeValidArray` and `MakeValidArrayEx` to help you construct variant array to pass data. See ***Making Arrays*** (on page 1470).

Programming Language Consideration

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

One major obstacle in connecting Microsoft Scripting Technology automation scripts to external code is the requirement to use COM safe types (especially variants) for function parameters. This is not a big issue with Visual Basic as it works very well with ActiveX and variants. However, for other programming languages such as Delphi and C++, it create a complex programming problem because variant is not a native type of these languages.

For Delphi, the issue can be resolved by using Delphi 5.0 which fully supports OLE variant type and automates many tasks related to variant array access.

For C++ users, please refer to the Win32 library of variant conversion routines.

ActiveX DLL Example

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

You can find an example of ActiveX DLL in the `Samples\Delphi\DLL Sample` directory in the NeoTicker® installation. This DLL implements a simple moving average indicator. Inside the directory you will find:

- A Dynamic Linked Library (DLL) example written in Delphi located in directory `sample_dll`
- A script called `sample_dll_caller.pas` containing an indicator called Sample DLL Caller

To install the example:

1. Exit NeoTicker®.
2. Install the indicator into NeoTicker® by copying the indicator script into the `indicator` directory of the NeoTicker® installation.
3. Register the ActiveX DLL object `CoAverage` into the Windows' registry using the regsvr32 program. You can do this by, in command prompt, entering:

   regsvr32 sample_dll.dll
4. For detail usage of regsvr32, check Microsoft's website (Q249873).
5. Start NeoTicker®.

To run the example indicator:

1. Open a time chart and load a 500-day MSFT daily chart (or any other chart you want).
2. Apply the Sample DLL Caller indicator to MSFT.

Let's examine the script `sample_dll_caller.pas` which calls the DLL:

```
function sample_dll_caller : double;
   var obj, prices;
begin
   obj := CreateOleObject ('sample_dll.CoAverage');
   data1.makearray (prices, 'C', Param1.Int, false, -1);
   result := obj.calc (prices, Param1.Int);
end;
```

The first statement:

```
   obj := CreateOleObject('sample_dll.CoAverage');
```

creates an object from which the external functions can be called. This call loads the DLL into the memory space of NeoTicker®. This way of linking an external library is called in-proc linking.This call The VBScript equivalent of `CreateOleObject` is `CreateObject`.

The second statement,

```
data1.makearray (prices, 'C', Param1.Int, false, -1);
```

creates an data array of the closing prices of `Data1` by using the `MakeArray` method.

The third statement:

```
result := obj.calc(prices, Param1.Int);
```

calls the function `calc` in the ActiveX DLL. The function `calc` must be a method exposed as an ActiveX method in the ActiveX DLL. The function returns a variant to the caller.

Try changing the parameters and inspect the source code of the DLL for better understanding of the mechanism of DLL based object creation and calling.

Because the ActiveX DLL is registered in Windows, you should unregister the ActiveX DLL when you no longer need it. You can do this by, in the command prompt, entering:

```
regsvr32 /u sample_dll.dll
```

DLL Consideration

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

Notice that the advantage of using DLL is that the memory space used by the DLL is allocated within the application (NeoTicker®). Thus loading the DLL is fast and the communication between the script and the DLL functions is fast. This leads to a slightly faster calculation speed compared to communicating with external programs. DLL also has the advantage of no visual interface thus less work to the developer if calculation is the only concern.

The disadvantage of using DLL is that the user cannot control the loading and unloading of the DLL. A DLL is loaded every time it is called and is unloaded after each bar finished its calculation. Thus the user cannot retain the state of the DLL unless such information is retained in the heap object of the script somehow.

Another disadvantage is development related. Since the DLL is linked with the main application, whenever you have to modify the DLL, you must exit NeoTicker® before your development tool can generate a new version of the DLL. If the DLL is linked to another application, you must exit that application as well.

DLL also poses a dangerous condition that it can crash the main application because the main program and the DLL use the same memory space.

ActiveX Server Example

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

Our second example is located in the `Samples\Delphi\ActiveX Sample` directory in the NeoTicker® installation. Inside the directory you will find:

- An example of ActiveX server written in Delphi located in directory `sample_activex`
- A script called `sample_activex_caller.pas` containing an indicator called Sample ActiveX Caller

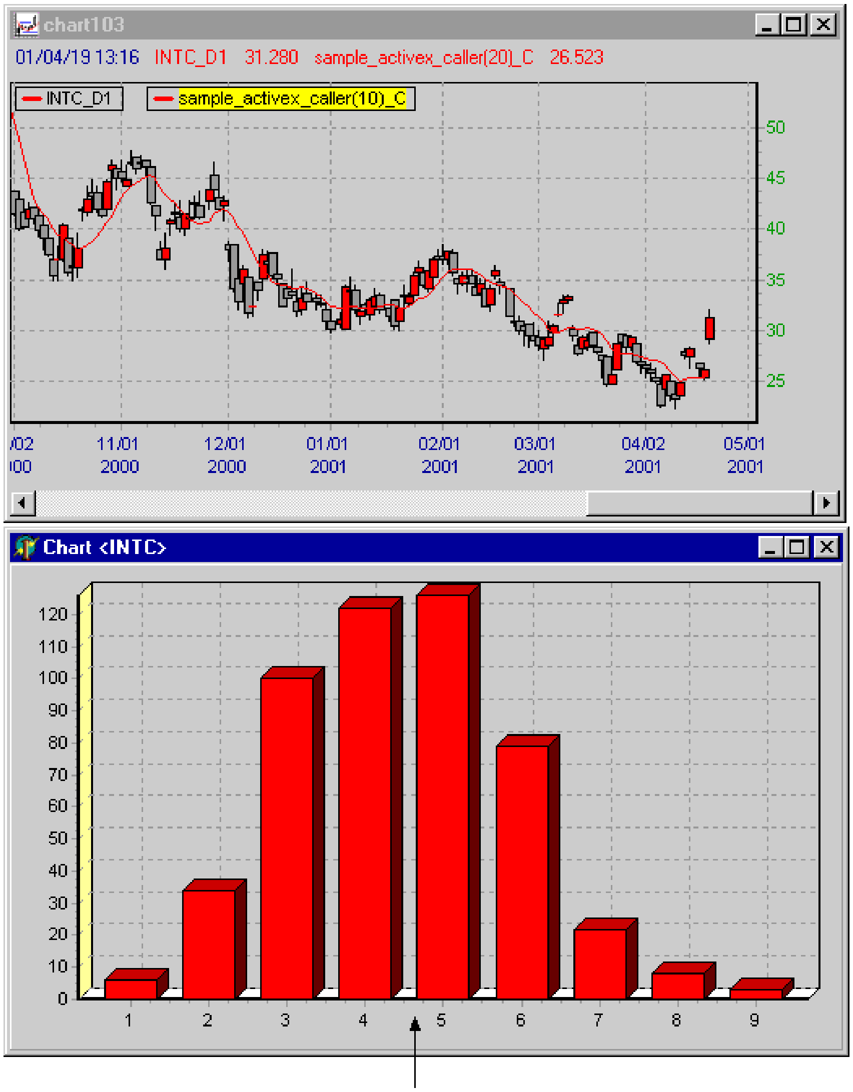
To install the example:

1. Exit NeoTicker®.
2. Install the indicator into NeoTicker® by copying the indicator script into the `indicator` directory of the NeoTicker® installation.
3. Register the ActiveX Server. The example supports self registration. Change to the ActiveX sample directory, and in command prompt, enter:

   ```
   sample_activex /regserver
   ```
4. Start NeoTicker®.

To run the example:

1. Open a time chart and load a 500-day INTC daily chart (or any other chart you want).
2. Apply the Sample ActiveX Caller indicator to INTC



ActiveX Server

Distribution of up closes within 10 bars for 500 days.

The ActiveX example collects statistics about up closes within the specified period, i.e. number of bars that are going up. It accumulates the number of up closes and displays a distribution of the up closes from the available data. The x-axis is the number of up closes over the period parameter, and the y-axis is the number of times that the particular combination has happened.

In the figure above, the chart is a daily chart and the default period is 10. You can see that within 10 days, INTC is most likely to go up for 5 days.

The ActiveX example also returns a simple moving average to the calling script. This illustrates that the ActiveX server can provide a lot more service than simply returning a value.

Consider the script `sample_activex_caller.pas`:

```
function sample_activex_caller : double;
   var obj, prices;
begin
   obj := CreateOleObject ('sample_activex.CoDistribution');

   if heap.size = 0 then // initialization
   begin
      heap.allocate (1);
      obj.init (data1.symbol);
      itself.success := false;
      exit;
   end;

   data1.makearray (prices, 'C', Param1.Int, false, -1);
   result := obj.calc (data1.symbol, prices, Param1.Int);
end;
```

The first statement:

```
   obj := CreateOleObject ('sample_activex.CoDistribution');
```

creates an object from which functions in the ActiveX server can be called. The VBScript equivalent of `CreateOleObject` is `CreateObject`.

In the if block, the statement,

```
   obj.init(data1.symbol);
```

calls an initialization method in the ActiveX server. The if block controls that the initialization is only called once.

The statement:

```
   data1.makearray (prices, 'C', Param1.Int, false, -1);
```

creates an data array of the closing prices of Data1.

The next statement:

```
   result := obj.calc (data1.symbol, prices, Param1.Int);
```

passes the symbol, data array and the period parameter to the ActiveX server to calculate and display distribution. The call also returns a simple moving average value for use as the indicator value in the script.

Remember to unregister the ActiveX server when you no longer need it. The example ActiveX server can self-unregister. To unregister, in command prompt, enter:

```
   sample_activex /unregserver
```

ActiveX Server Side Consideration

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

The advantage of using external ActiveX server is its ability to retain its state if the ActiveX sever is started before being called (if the ActiveX server is not started beforehand, then it would act like a DLL which get loaded and unloaded every time a bar is being calculated). Visual presentation is possible when using an ActiveX server connected to NeoTicker® through the scripts.

If you want to retain state in the ActiveX server, you must be prepared to handle multiple calls from NeoTicker® from within multiple charts. It is not guaranteed that the calls are made sequentially to the ActiveX server. The example server provides an idea as how to resolve this issue - the symbol name is used to identify the caller. An alternative is to use a randomly generated unique identifier to identify the caller.

Also, for visualization purpose, if you choose to develop ActiveX server that can handle multiple calling origins, then you may want to create multiple presentation forms where each form corresponds to one chain of calls from a particular chart.

Overall, ActiveX server application is the preferred way for 3rd party developers to create functionality extension to NeoTicker®.

References

If you are creating an indicator, you should use ***IDL Interface*** (see "IDL Interface (DLL, ActiveX and .NET indicators)" on page 1635). External interface is for interfacing with application that is not designed as an NeoTicker® indicator.

To drive NeoTicker® to do various tasks from an external program, use ***OLE Automation*** (on page 653).

For Delphi, refer to the Delphi Programming Reference for detail description of OLE programming.

For VBScript, refer to Microsoft's ***Windows Script*** (http:\\msdn.microsoft.com\scripting) page, the VBScript reference section for more details.

Custom Drawing Tool

Sample Custom Drawing Tool

The Time Cycle drawing tool is written in VBScript and is part of the installation. For real life example on how to construct a custom drawing tool, feel free to examine the code of Time Cycle. If you are interested in modifying the code, it is recommended that you should save your modification under a new file name before you start modifying the code.

Essential Functions

To define a custom drawing tool, you need to provide the definition of the following two functions.

function numpoints : integer;

It tells NeoTicker® the custom drawing object has how many control points. It can range from 1 to 3.

procedure draw2(boundary, xlist, ylist, price, hlist, vlist, flist, clist, var commands);

The meaning of each parameters is as follows.

boundary

It is an array (0 .. 3) of integer mapping the coordinates of Left, Top, Right, Bottom.

xlist

It is an array (0 .. 2) of integer providing a list of x coordinates of the handles.

ylist

It is an array (0 .. 2) of integer providing a list of y coordinates of the handles.

price

`Price` is the matching price of the y-coordinate for your use.

hlist

It is an array of double providing a list of user defined multiples for your use.

vlist

It is an array of double providing a list of user defined multiples for your use.

flist

It is an array of double providing a list of user defined multiples for your use.

clist

It is an array of double providing a list of user defined multiples for your use.

Returning Drawing Commands

In return, you send a variant array back to the user through the commands variable in the form of

```
{ command parameter_list }*
```

where `command` is an integer and `parameter_list` is the list of parameters required by that particular command.

The available drawing commands are:

0 draw line (p1, p2) - (p3, p4) no extension

1 draw line (p1, p2) - (p3, p4) use extend left option

2 draw line (p1, p2) - (p3, p4) use extend right option

3 draw line (p1, p2) - (p3, p4) use extend left right options

4 draw line (p1, p2) - (p3, p4) force extend left

5 draw line (p1, p2) - (p3, p4) force extend right

6 draw line (p1, p2) - (p3, p4) force extend left right

10 draw rectangle (p1, p2) - (p3, p4)

11 draw ellipse (p1, p2) - (p3, p4)

12 draw arc (p1, p2) - (p3, p4) for p5 degrees

20 draw text (p1, p2) top left corner with string p3

21 draw text (p1, p2) top right corner with string p3

22 draw text (p1, p2) bottom left corner with string p3

23 draw text (p1, p2) bottom right corner with string p3

Sample command list drawing a box from (100, 100) to (200, 200)

```
(
0, 100, 100, 100, 200,
0, 100, 200, 200, 200,
0, 200, 200, 200, 100,
0, 200, 100, 100, 100
)
```

Optional Functions

If an optional function is not defined, default service will be used in place for the missing functions.

function toolname : string;

You can provide a name to the tool and it will be displayed in the tool hint and the tool bar setup

function usehmult : boolean;

If set to true, the drawing tool will be able to use the `hmultiple` data from NeoTicker® and its UI will be available for user input. Default to false if not defined.

function usevmult : boolean

Similar to `usehmult`, if set to true, `vmultiple` data will be available. Default to false if not defined.

function usefmult : boolean;

Similar to `usehmult`, if set to true, `fmultiple` data will be available. Default to false if not defined.

function usecmult : boolean;

Similar to `usehmult`, if set to true, `cmultiple` data will be available. Default to false if not defined.

procedure clicked2(boundary, xlist, ylist, price, hlist, vlist, flist, clist, var commands);

By default, user can select the custom drawing tool by clicking on the lines connecting the handles. The clicked2 function allows you to specify extra coordinates for clicking. This function is useful for complex drawing objects that have multiple places to be clicked on.

You are provided with the same set of parameters as `draw2`. You must return a list of `(x1, y1) - (x2, y2)` coordinates in the commands variable for the checking of your drawing object is clicked by the user

Obsolete Functions

Following functions are provided for backward compatibility only.

function draw(boundary, xlist, ylist, prices, hlist, vlist, flist) : variant;

function clicked(boundary, xlist, ylist, prices, hlist, vlist, flist) : variant;

CHAPTER 14

Guide To Protection

This guide cover protection topics that are related to third party developments

NeoTicker® provides tools to help you develop indicators, windows and templates that are suitable to deploy in a customer site.

In This Chapter

Introduction to Protection

Understand the Role of Protection

Protection enforces legitimate usage of your work. There is no such thing as perfect protection. Given enough resources, any protection scheme is defeatable. As a third party developer, it is your job to provide value in your work that customers are willing to pay you rather than trying to defeat your protection scheme.

NeoTicker® offers a flexible protection scheme. In this section, we will explain what NeoTicker®'s protection scheme does so you can make an informed choice on what to use.

What are Protected

NeoTicker® can protect the following types of work:

- Indicators
- Function Windows (charts, quote windows, etc)
- Templates (template for charts, quote windows, etc)

Parties Involved

We can identify three parties a protection scheme:

- Developers - people who implements the product and choose the protection scheme, i.e. programmers.
- Administrators - people who administrate the product according to the protection scheme, i.e. administrative staffs.
- Users - people who use the product.

In smaller organizations and in new products, developer and administrator tend to be the same person. Administrators come into play only when the product grows to a point that work division becomes necessary.

In the following sections, we will explain the type of protections available in NeoTicker® and how these protections are viewed from the three parties involved.

Source Code Protection

At the core of NeoTicker®'s protection scheme is Source Code Protection.

Source Code Protection eliminates the ability of anyone other than yourself to look at your work at source level. For example, if you develop an indicator using formula, when protected, your customers can use the indicator, but they cannot look at the formula. This prevents your customers from gaining knowledge from the formula to create identical and derivative indicators.

NeoTicker® implements Source Code Protection by encrypting your indicators, windows, templates. Once encrypted, your work is no longer readable by human, and can only be understand by NeoTicker®.

An additional password is used for Source Code Protection within your organization. Personnel with password have full access to your work. Personnel without password cannot view your work (because it is encrypted), but can handle administrative tasks such as activating customers to use your work.

In summary,

- Developers - have full access to source code using password.
- Administrators - do not have access to the source code.
- Users - do not have access to the source code.

Usage Control

Usage Control serves the following functions:

- Eliminates the ability of your customers to distribute your work to other users
- Lets you create time limited usage version of your work for lease and demo purposes

For example, suppose you've developed an indicator. With usage control, you can create a lease plan that lets a customer using the indicator for one year. After one year, the indicator will expire and will no longer work in NeoTicker®. The customer must renew the lease through you in order to continue using the indicator.

NeoTicker® implements Usage Control by activation. Before a customer can use your work, they must first contact you with information of their NeoTicker® copy. Based on this information, you can create a key to activate your work for the customer.

In summary,

- Developers - responsible for choosing a usage control scheme.
- Administrators - responsible for granting usage to users.
- Users - cannot use product until being granted usage.

Defining a Product

Now lets talk about what is being protected here. In this section, we will formalize the concept of a product. A product is an atomic item you want to sell to your customers.

In this section we are concerning ourselves with development of a product, not the administration, which will be covered in later sections.

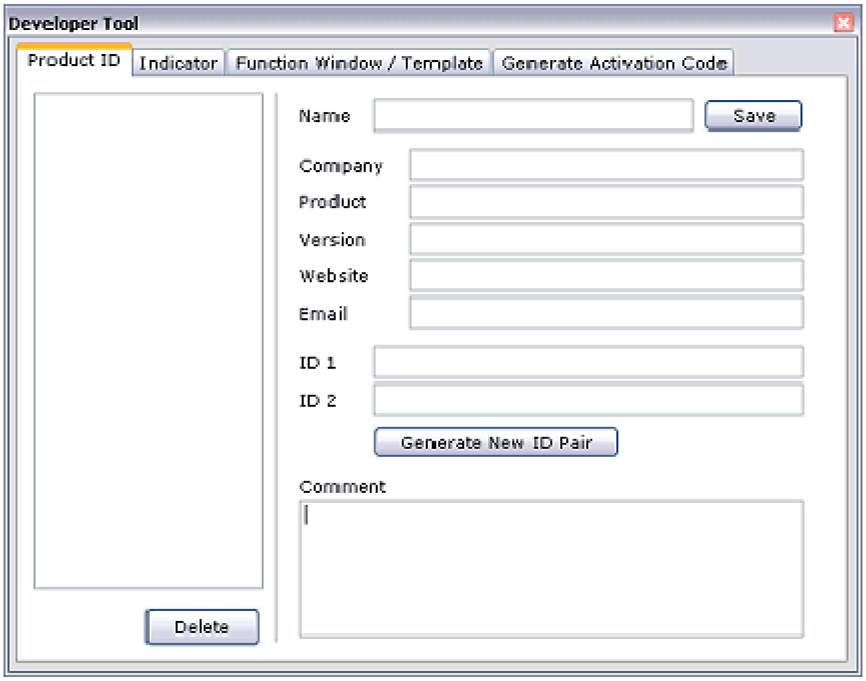
Think in Terms of Product

When you produce work to your customers, the work is not necessary a single item. For example, your work can constitute two indicators and a chart. You should consider the items as a single product. Your customers wouldn't want just the indicator or just the chart. They need the whole product. So it is natural to group the two indicators and the chart into a single product.

What defines a product then? Each product has an unique identity and items in the product share this identity. So, the product items share a unified encryption/activation scheme and different products can have different schemes.

Developer Tool

Product definition is handled by Developer Tool. You can open Developer Tool under **Tool>Developer Tool** in main window.



ID Strings

What uniquely defines a product are two ID strings. To generate ID strings,

1 In Developer Tool, press the **Product ID** tab.

2 Press **Generate New ID Pair** button.

3 The ID strings are shown in **ID 1** and **ID 2**.

4 Give a name to the product and press the **Save** button.

Because the size of the string, **Generate New ID Pair** button practically does not produce the same string twice. Therefore, by pressing this button, you can ensure your product has a unique product id that is different from all other products in the world.

ID strings are essential to encryption/activation. Therefore, you should make sure:

- You keep back ups of ID strings (print it out if necessary).
- They are handled only by trusted personnel and never leak outside of your organization.

Non-ID Strings Information

For a product, other than the ID strings, there are information you should fill in under **Product ID** tab (Company, Product and Version strings). Unlike ID strings, these information are not used for encryption/activation purpose but does help identifies the product.

When you create a new product, you should fill in these strings.

When you release a new version of the product and want activation files generated for previous versions to continue to work, do not change these strings.

When you release a new version of the product and want activation files generated for previous versions to stop working, change the version string.

Example

For example, your company ACME Inc develops an indicator called Zen Predictor. Zen Predictor is a product that has a chart and two indicators. The following steps will create a product for Zen Predictor.

1 In Developer Tool, press the **Product ID** tab.

2 For **Name**, enter Zen Predictor.

3 For **Company**, enter ACME Inc.

4 For **Product**, enter Zen Predictor.

5 For **Version**, enter 1.0.

6 For **Website**, enter www.acmeinc.com.

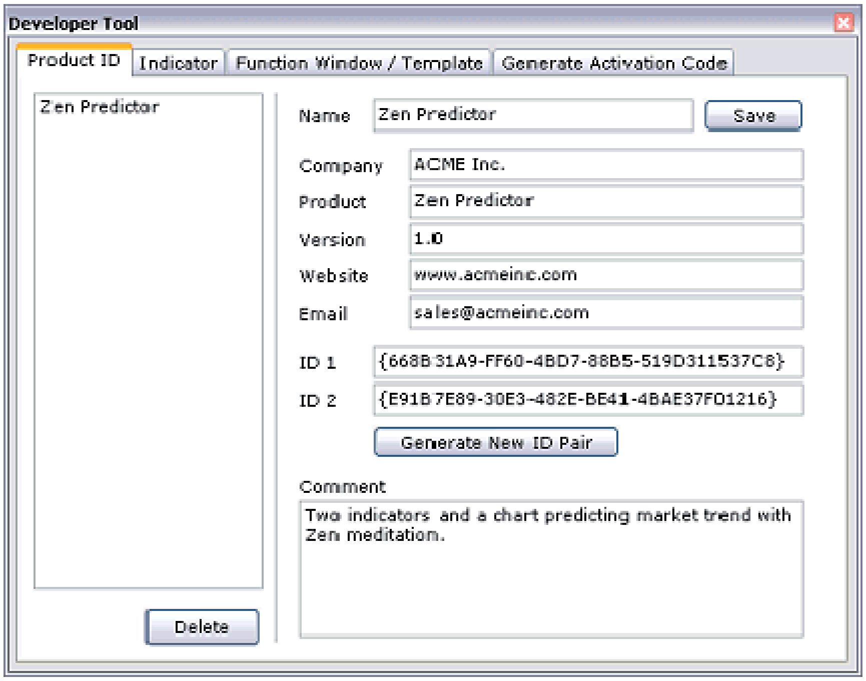
7 For **Email**, enter sales@acmeinc.com.

8 For **Comment**, enter some descriptive text about the product (comments are for developers only, customers will not see the comments).

9 Press **Generate New ID Pair** button.

10 Press **Save** button.

The following figure shows the example product, Zen Predictor. Everytime you need to encrypt/authorize for Zen Predictor, select Zen Predictor in **Product ID** tab.



Zen Predictor

Name Zen Predictor

Company ACME Inc.

Product Zen Predictor

Version 1.0

Website www.acmeinc.com

Email sales@acmeinc.com

ID 1 {668B31A9-FF60-4BD7-88B5-519D311537C8}

ID 2 {E91B7E89-30E3-482E-BE41-4BAE37F01216}

Comment

Two indicators and a chart predicting market trend with Zen meditation.

We will be using Zen Predictor as an example in later sections to illustrates how to protect indicators and windows.

Protecting Indicators

To protect an indicator, a locked version of the indicator is created. The locked version is an encrypted version of the original indicator. The locked version is not human readable, so it is a form of Source Code Protection. Who can use the indicator depends on the options.

To create the locked version of an indicator.

1. Open Developer Tool.
2. Press **Indicator** tab.
3. Under **File**, choose the indicator file you want to lock (e.g. ZenPredictorIndicator1.pas).
4. Under **Prod Id**, choose the Product ID the indicator belongs to (e.g. Zen Predictor).
5. If the indicator requires activation before it can be used, enable **Require Activation Code**. If you leave this option off, the locked indicator will not be in a human readable form, but can be used by any user.

   Disable this option if you want to give away the product but not revealing the internal logic of your work.

   Enable this option if you want to charge money for your product so you want to limit access only to users who paid.
6. Under **Password**, enter a password of your choice. Type the password again under **Confirm Pwd**. Password is required if you want to view/edit a locked indicator.

   With a password, you can view/edit a locked indicator.

   Without a password, your staff can activate the product for customers, without the ability to view the source code of the indicator.
7. Press the **Create Locked Version** button. A locked version of the indicator will be created. The locked version has a .lok suffix (e.g. ZenPredictorIndicator1.lok). This file is not human readable and is to be distributed to customers.

Tips:

- For easy development, you should keep an regular (not locked) version of an indicator and work on that. Only when you are ready to send it customers, should you lock it.
- If you have multiple indicators in a product, you must create locked version one by one.
- Password is not saved with your Product ID or NeoTicker® session for security reason. You should memorize it, or write it down and keep it in a safe place.

IDL Indicators

IDL indicators consist of two parts, a header and a DLL created by a compiler. The header part is encrypted, but the DLL part is not encrypted. You will need to send both parts to your customers. Because DLL is compiled object code, end user cannot read the source code of a DLL indicator. However, you must take extra measure to protect the DLL being called by a header other than your own.

To help you check if your DLL is called by a legitimate IDL header, we provide a mechanism called a developer checksum, which is based on your product ID strings. In the DLL indicator, you must check at least once for the developer checksum. Only when the checksum matches, your DLL should proceed with normal calculation.

To obtain developer checksum:

1. Open Developer Tool.
2. Press **Indicator** tab.
3. Under **Developer Checksum**, choose the Product ID the indicator belongs to.
4. Copy the developer checksum for use in IDL indicator.

To verify developer checksum in IDL indicator:

1. In IDL indicator, use the property `ItSelf.DeveloperChecksum` to obtain the developer check sum.
2. Compared the value with the value obtained. For example, if the developer checksum is 123456, then the code (Delphi) looks like:

```
if ItSelf.DeveloperChecksum <> 123456 then exit;
```

Examples:

- If you want your indicator usable by anyone, but the logic hidden, disable **Require Activation Code**. Send the locked version to your customers. They will be able to use the indicator without activation.
- If you want to create a demo version of an indicator with time limited usage, enable **Require Activation Code**. Your customer will need to contact you for activation. Once the time limit expires, your customer can no longer use the indicator.
- If you want to create a lease version of an indicator, enable **Require Activation Code**. Your customer will need to contact you for activation. Once the lease expires, your customer can no longer use the indicator until they renew the lease from you.
- If you want to create a permanent version of an indicator with no time limited usage, enable **Require Activation Code**. Your customer will need to contact you for activation.

Protecting Windows and Templates

Protecting windows and templates is very similar to protecting indicators. A locked version of the file is created to be distributed to your customers.

The locked version is an encrypted version of the original window/template. The locked version has the following properties:

- The locked file is not human readable, so it is a form of Source Code Protection.
- Who can use the window/template depends on the options.
- Once the file is opened by NeoTicker® as a window/template, the user cannot read any formula in the window/template. For example, if a chart contains an fml indicator, the formula in the fml indicator is not visible.
- If the locked file is saved again by the user, it will be saved in the same protected form as the original.

To create the locked version of an window/template:

1. NeoTicker® allows you to write protect a window. It is recommended that you turn on write protection for a window before you lock it. This way, your customers cannot accidentally overwrite the version you send to them.
2. Open Developer Tool.
3. Press **Function Window / Template** tab.
4. Under **Type**, choose the type of window/template you want to lock, e.g. chart, quote, etc.
5. If you are locking a function window, choose **Window**. If you are locking a template, choose **Template**.

6 Under **File**, choose the window/template file you want to lock (e.g. ZenPredictorChart).

7 Under **Prod Id**, choose the Product ID the window/template belongs to (e.g. Zen Predictor).

8 If the window/template requires activation before it can be used, enable **Require Activation Code**. If you leave this option off, the locked window/template will not be in a human readable form, but can be used by any user.

Disable this option if you want to give away the product but not revealing the internal logic of your work.

Enable this option if you want to charge money for your product so you want to limit access only to users who paid.

9 Under **Password**, enter a password of your choice. Type the password again under **Confirm Pwd**. Password is required if you want to view/edit a locked indicator.

With a password, you can view/edit a locked window/template file.

Without a password, your staff can activate the product for customers, without the ability to view the window/template file.

10 Press the **Create Locked Version** button. A locked version of the indicator will be created. The locked version has a (lock) suffix (e.g. ZenPredictorChart (lock)). This file is not human readable and is to be distributed to customers.

Tips:

- Window/template locking is a weak form of protection. Once a customer opens the window/template, it is not difficult to duplicate the settings manually to reverse engineer your work. Therefore, you must create settings that you deem valuable using formula or a locked indicator, both of which are not visible from the user.
- For easy development, you should keep an regular (not locked) version of the window/template and work on that. Only when you are ready to send it to customers, should you lock it.
- If you have multiple windows/templates in a product, you must create locked version one by one.
- Password is not saved with your Product ID or NeoTicker® session for security reason. You should memorize it, or write it down and keep it in a safe place.
- If your chart contains an indicator you write, and the indicator takes formula as a parameter. When you design the indicator, make sure the parameter is designated as a formula, not a string. This ensures NeoTicker® blocks view of the formula from your customers.

Examples:

- If you want your window/template usable by anyone, but the logic hidden, disable **Require Activation Code**. Send the locked version to your customers. They will be able to use the window/template without activation.
- If you want to create a demo version of an window/template with time limited usage, enable **Require Activation Code**. Your customer will need to contact you for activation. Once the time limit expires, your customer can no longer use the window/template.
- If you want to create a lease version of an window/template, enable **Require Activation Code**. Your customer will need to contact you for activation. Once the lease expires, your customer can no longer use the window/template until they renew the lease from you.
- If you want to create a permanent version of a window/template with no time limited usage, enable **Require Activation Code**. Your customer will need to contact you for activation.

Installation on Customer Site

At the time of writing there is no automatic installation tool for third party products. This is a planned feature for NeoTicker® 4.

If you need to install products on customer site, you or your customers need to install it manually.

Assume NeoTicker® is installed in the directory `C:\Program Files\TickQuest\NeoTicker4`:

- Indicators should be installed in the directory `C:\Program Files\TickQuest\NeoTicker4\indicator`

  For example, `ZenPredictorIndicator1.lok` and `ZenPredictorIndicator2.lok` should be installed inside the `indicator` directory.
- Function windows should be installed in the directory `C:\Program Files\TickQuest\NeoTicker4\Window`, under a sub directory of the matching function window type.

  For example, `ZenPredictorChart (lock)` should be installed in the `Window\chart` directory.
- Templates should be installed in the directory `C:\Program Files\TickQuest\NeoTicker4\Template`, under a sub directory of the matching function window type.

Activating Your Product for Customer Use

Finally, we come to the administration of your product, i.e. activating your product for customers. Notice that for a staff to activate a product, there is no need for the staff to have full access to the indicator/window/template source code. All the staff needs is a copy of NeoTicker®, the product ID, and the locked version of the indicator/window/template files.

In order for you to activate a customer to use your product, you will need the customer to send you information about their copy of NeoTicker®.

Have Customer to Send You NeoTicker® Info

Instruct your customer to choose from main window, **Help>User Info**. This will open a dialog that contains some text message. Have the customer to send you the text.

What are Customer Information

There are three pieces of information:

- User ID - This is the user id of the customer
- HW Key ID - This is the hardware key id
- Machine Code - This is the computer id

You will need to use a combination of the three information to generate a activation key file that you will send back to the customer. You do not need to use all three pieces of information. Different combination will yield different levels of protections.

The following table lists how the customer information affects the level of protection provided, as well as the applicability of the protection to different versions of NeoTicker®.

Information Used	NeoTicker® Version	Description of Key Generated
No information	Real-time, EOD	The key generated can be used by any user to activate your product. Typically, this is suitable for a beta program of you product, when you want a large audience to use your produc do not want usage to extend beyond certain date.

User ID	Real-time, EOD	The key generated can only be used by the user with the ma matching user id. For real-time version, this is a strong protection. For EOD version, this is a weak protection. Customer does not require to re-activate after upgrading cor
HW Key ID	Real-time	The key generated can only be used by NeoTicker® with m hardware key. This is a strong protection but cannot be used with EOD ver Customer does not require to re-activate after upgrading cor
Machine Code	Real-time, EOD	The key generated can only be used on same customer comp This is a medium strength protection. Customer may need to re-activate after upgrading computer
User ID, HW Key ID	Real-time	Do not use this combination. Keys generated is no better tha generated using HW Key ID alone.
User ID, Machine Code	Real-time, EOD	The key generated can only be used by the same customer o same computer. This is a strong protection. Customer may need to re-activate after upgrading computer
HW Key ID, Machine Code	Real-time	Do not use this combination. Keys generated is no better tha generated using HW Key ID alone.
User ID, HW Key ID, Machine Code	Real-time	Do not use this combination. Keys generated is no better tha generated using HW Key ID alone.

Generating Activation Key File for Customer

Once you receive customer user information, you can generate a key to activate the product for your customer:

1 Open Developer Tool.

2 Press **Generate Activation Code** tab.

3 Under **Prod ID**, choose the product.

4 Under **Usage**, choose **Permanent** if the product is activated forever for the customer. Choose either **Next** or **Expiration Date** for creating an activation key file for demo/lease version of your product.

5 Fill in **User ID**, **HW Key ID** and/or **Machine Code**. Consult the table above on what you need to fill in.

6 Specify **Default Directory**. This is where the activation key file saves to.

7 Specify **Filename**. This is the filename of the activation key file. Yan use the **Make Filename** button to help you create a filename.

8 Press the **Gen File** button.

Sending Activation Key File to Customer

The activation key file is a text file. The customer must receive this file in order to activate your product on his/her copy of NeoTicker®.

For example, you can send the activation key file as an email attachment. The activation file contains information for self-consistency check. If the activation key file is corrupted by any mean, NeoTicker® will report the problem to your customer and he/she can re-request the activation key file from you.

For information on how a customer can install the activation key file, see ***Activating Products from Third Party Developers*** (on page 405).

Activating Products from Third Party Developers

There are two steps to activate a product from a third party developer.

Sending Information About NeoTicker® to the Developer

This step provides information to the third party developer so that they can create a key to activate their product for your copy of NeoTicker®.

The third party developer will instruct you on how to send the information to them. Usually, you will need to choose from main window, **Help>User Info** to open a dialog, and send the information in the dialog to the third party developer.

Authorizing the Product

Once the third party developer receives the information, they can generate an activation key file and send it to you.

The activation key file is a text file. After you receive the activation key file from the third party developer:

1 In main window, choose **Tool>Activation Tool** to open Activation Tool.

2 In Activation Tool, press **Install Activation File** tab.

3 You can drag the activation key file to the area labelled **Drag and Drop Activation File Here**. Alternatively, you can press the **Browse For Activation File** button to locate the activation file yourself.

Emergency Recovery

Ideally you should always keep an original, unlocked version of your work.

However, if for some reason the original unlocked version is lost, you can unlock the work if you still have the product id and your password.

To unlock an indicator:

1 Open Developer Tool.

2 Press **Indicator** tab.

3 Under **Unlock Indicator**, enter **File** name of the locked file, **Prod ID** and your **Password**.

4 Press **Recover Original Version** button.

To unlock window/template:

1 Open Developer Tool.

2 Press **Function Window / Template** tab.

3 Under **Unlock Window / Template**, enter **Type**, choose **Window** or **Template**, enter **File** name of the locked file, **Prod ID** and your **Password**.

4 Press **Recover Original Version** button.

CHAPTER 15

Indicator Reference

NeoTicker® provides an extensive set of indicators that you can use within time charts, quote windows, pattern scanners, etc.

The syntax and examples of usage for indicators is under construction and will be expanded to cover all indicators in future releases.

In This Chapter

List in Alphabetical Order

A-G

Absolute Value (abs)

Returns the value of the data series without the sign of each value. Absolute value requires one specified series (Link 1).

Accumulate (accum)

Accumulate is an accumulator : if Link 1 is valid, it will accumulate either the value of Link 1 or count it base on the default values.

Accumulation Swing Index (accswingindex)

Cumulative form of Swing Index. Accumulation Swing Index takes one parameter **Limit** and requires one series (Link 1).

Accumulation Distribution (ad)

Returns Accumulation Distribution based on the volume of the data series. Accumulation Distribution requires one series (Link 1).

Accumulation Distribution Custom (ad_custom)

Returns Accumulation Distribution using the second series as the volume information. Accumulation Distribution requires two series.

Adaptive Moving Average (ama)

Adaptive Moving Average requires one series (Link 1).

Addition (add)

Addition of two series (Link 1 + Link 2).

Aroon Down (aroondown)

Aroon Down for a given time period is calculated by determining how much time elapsed between the start of the time period and the point at which the lowest closing price during that time period occurred.

Aroon Oscillator (aroonosc)

The Aroon Oscillator signals an upward trend is under way when it is above zero and a downward trend is under way when it falls below zero. The further away the oscillator is from the zero line, the stronger the trend.

Aroon Up (aroonup)

Aroon up for a given time period is calculated by determining how much time elapsed between the start of the time period and the point at which the highest closing price during that time period occurred.

Auto Trendline (auto_trendline)

Auto Trendline draws 2 trend lines based on the latest 2 swing highs and 2 swing lows. Auto Trendline requires one series (Link 1).

Average Directional Movement Index (ADX)

A smoothed version of DMI. ADX takes one parameter **Period** and requires one series (Link 1).

Average Range (avgrange)

Returns average range of the data series over Period of bars. Average Range requires one data series (Link 1).

Average True Range (avgtruerange)

Returns the average value of all the true range values over the given **Period** of bars. Average True Range requires one series (Link 1).

Backtest Basic (backtest)

Backtest Basic is a legacy indicator. Use Backtest EZ instead.

Conduct basic system test using Link 1 as the Price series and Link 2 as the Signal series. When the signal series turn from negative to positive, the system will close the existing short position and buy at market, vice versa. If the signal series turn from either positive or negative to 0, then it will just close the current position.

BackTest EZ (backtestez)

A generic system backtesting framework. You just need to supply your buy sell conditions and this indicator will take care of the rest for you. BackTest EZ is one of the power indicators. Refer to the ***Power Indicators Guide> Backtest EZ*** (see "Backtest EZ" on page 1275) for more details. BackTest EZ requires two series (Link 1 and Link 2).

Barsago By Time (barsagobytime)

Barsago By Time returns the number of bars ago for the specified time and days ago input. Barsago By Time requires one data series (Link 1).

Bars Since (barssince)

Calculate the number of periods that have passed since the most recent occurrence that formula condition is true. This indicator require one data series (Link 1).

Bars Since Nth (barssincen)

Calculate the number of periods that have passed since the N-th most recent occurrence that formula condition is true. This indicator require one data series (Link 1).

Bollinger Band (bband)

Given the *Period* and *Offset* in terms of multiple of the standard deviation, returns the Bollinger Band value. Bollinger Band requires one series (Link 1).

Bollinger Bands 3 Lines (bbands3)

Given the *Period*, and *Offset* in terms of multiple of the standard deviation, return 3 bollinger band values. Bollinger Bands 3 Lines requires one series (Link 1).

Bollinger Bands 5 Lines (bbands5)

Given the *Period*, *Inner Offset* and *Outer Offset* in terms of multiple of the standard deviation, return 5 bollinger band values. Bollinger Bands 5 Lines requires one series (Link 1).

Chaikin Oscillator (chaikinosc)

Returns the Chaikin Oscillator. Chaikin Oscillator requires one series (Link 1).

Chaikin Oscillator Custom (chaikinosc_custom)

Returns the Chaikin Oscillator. Chaikin Oscillator Custom requires two series. Link 1 is used for defining the price series while Link2 is used for defining the Volume series.

Chande Momentum Indicator (cmo)

Chande Momentum Indicator as described by Chande and Kroll in The New Technical Trader.

Color Plot Formula (colorplotfml)

Color Plot Formula will plot colors based on the formula and coloring rules provided. Color Plot Formula requires one series (Link 1). For usage detail please refer to ***Power Indicators Guide> Color Plot Formula*** (see "Color Plot Formula" on page 1287).

Commodity Channel Index (cci)

CCI is a price momentum indicator based on a scoring system that helps identify possible cyclical behavior. CCI takes one parameter **Period** and requires one series (Link 1).

Constant (const)

Constant requires one specified series (Link 1), which defines the time period when Constant returns a valid value.

Constant 2 Lines (const2)

Constant 2 Lines requires one specified series (Link 1).

Constant 3 Lines (const3)

Constant 3 Lines requires one specified series (Link 1).

Constant 4 Lines (const4)

Constant 4 Lines requires one specified series (Link 1).

Constant 5 Lines (const5)

Constant 5 Lines requires one specified series (Link 1).

Correlation Coefficient (correl)

Returns the statistics correlation coefficient over the specified *Period.* Correlation Coefficient requires one series (Link 1).

Cross Above (xabove)

Returns 1 when Link 1 crosses above Link 2, 0 otherwise.

Cross Above Constant (xaboveconst)

Returns 1 when Link 1 crosses above parameter Constant, 0 otherwise.

Cross Below (xbelow)

Returns 1 when Link 1 crosses below Link 2, 0 otherwise.

Cross Below Constant (xbelowconst)

Returns 1 when Link 1 crosses below parameter Constant, 0 otherwise.

Crossed (xcross)

Returns 1 when Link 1 crosses above Link 2 and -1 when Link 1 crosses below Link 2, 0 otherwise.

Crossed Constant (xcrossconst)

Returns 1 when Link 1 crosses above parameter Constant and -1 when Link 1 crosses below parameter Constant, 0 otherwise.

Current Day High (currDHigh)

Given a daily or intraday chart, this indicator draws a horizontal line marking the current day high price level and automatically adjust itself when the day has changed. This is a support/resistance indicators. Current Day High requires one series (Link 1).

Current Day Low (currDLow)

Given a daily or intraday chart, this indicator draws a horizontal line marking the current day low price level and automatically adjust itself when the day has changed. This is a support/resistance indicators. Current Day Low requires one series (Link 1).

Current Day Open (currDOpen)

Given a daily or intraday chart, this indicator draws a horizontal line marking the current day open price level and automatically adjust itself when the day has changed. This is a support/resistance indicators. Current Day Open requires one series (Link 1).

CurrMonth Open (currMOpen)

Current Month Open draws a series of horizontal dots marking the opening price level of the current month. It works on series with time frame higher than monthly. It requires one series (Link 1).

CurrWeek Open (currWOpen)

Current Week Open draws a series of horizontal dots marking the opening price level of the current week. It works on series with time frame higher than weekly. It requires one series (Link 1).

Custom 1 Delphi (custom)

Custom 1 Delphi is a legacy indicator. Use formula indicator instead.

Custom 1 Delphi is an indicator that evaluates Delphi Script expression. The expression can access Link 1 and the user specified parameters. To access the value of Link 1, use `Data1.value[n]` in the expression, where `n` is the number of bars ago. Custom Delphi 1 provides three user parameters: `param2`, `param3` and `param4`. To access these parameters in the expression, use `param2.real`, `param3.real` and `param4.real`.

Custom 2 Delphi (custom2)

Custom 2 Delphi is a legacy indicator. Use formula2 indicator instead.

Custom 2 Delphi is identical to Custom 1 Delphi except it can access Link 2 in addition to Link 1.

Data Breadth (databreadth)

Data Breadth returns 4 basic real time market breadth data (Advance Issues, Decline Issues, Advance Volume, Decline Volume) based on the data series from the First Series to the Last Series. For detail on usage please refer to ***Power Indicator Guide> Data Breadth*** (see "Data Breadth" on page 1301).

Day Percent (DayPercent)

Day Percent returns in percentage (0 to 100) of the current bar time relative to the trading time range of the hosting function window

Days Since (dayssince)

Days Since returns the number of days since the given date. This indicator must be applied on a series that has a time frame that is 1 day or lower. Days Since requires one specified series (Link 1).

Directional Movement Index (DMI)

Directional Movement Index is a trend indicator. DMI will rise when the trend is developing and will stay at higher value if the trend is intact. DMI is calculated based on DMI Plus and DMI Minus. DMI takes one parameter **Period** and requires one series (Link 1).

Directional Movement Index Minus (DMIminus)

It is the down component of the DMI. DMI Minus takes one parameter **Period** and requires one series (Link 1).

Directional Movement Index Plus (DMIplus)

It is the up component of the DMI. DMI Plus takes one parameter **Period** and requires one series (Link 1).

Distribution Plot (distplot)

Returns a distribution graph based on formula-driven condition and value to measure. Summary information of the distribution is optionally reported. Real-time update of the last N occurence including marker for the position of the last updated position in the distribution. If **RT LastN** is set to 0, the real-time line plot will be disabled. This indicator will refuse to run if set to update-by-tick. Distribution Plot is one of the power indicators. Refer to the ***Power Indicators Guide> Distribution Plot*** (see "Distribution Plot" on page 1307) for more details. Distribution Plot requires two series (Link 1 and Link 2).

Division (div)

Divides one series from another (Link 1 / Link 2).

Double Exponential Moving Average (dema)

Returns DEMA as described in Technical Analysis Stocks and Commodities magazine, V12.

Ease of Movement (ease)

Ease of Movement indicator indicates possible strength when it crosses from below zero to above zero and vice versa. Other usage includes finding specific levels of exhaustion and using that as a hint to identify turning points. Ease of Movement requires one series (Link 1).

Equi Volume Overlay (Equivol)

Equi Volume Overlay draws overlay region of the data series based on fixed volume or ticks. Options to plot open, close, and even volume based information like VWAP of each region. Equivol requires one series (Link 1).

Exponential Average Range (xavgrange)

Returns exponential moving average of the range of the data series. Exponential Average Range requires one series (Link 1).

Exponential Moving Average (xaverage)

Exponential Moving Average is effectively the mean of the complete data series from the beginning up to the point of calculation. Exponential Moving Average takes one parameter *Period* and requires one series (Link 1).

Formula (fml)

Returns the evaluation of the expression defined in plot1. Formula requires one series (Link 1).

Formula 2 (fml2)

Returns the evaluation of the expression defined in plot1. Formula 2 requires two series.

Gap Down (gapdown)

This indicator returns a +1 when the security's price gap down. Otherwise it returns a 0. A gap down occurs when today's open is lower than yesterday's low. This indicator require one data series (Link 1)

Gap Up (gapup)

This indicator returns a +1 when a security's price gap up. Otherwise it returns a 0. A gap up is when today's open is higher then previous day high. This indicator requires one data series (Link 1).

Gap Value (gap)

This indicator calculates the day gap value of a security. It return positive value when the security is gapped up, negative value if the security is gapped down, otherwise it returns 0. A gap occurs if yesterday's high is greater than today's open or yesterday's low is less than today's open. This indicator requires one data series (Link 1).

H-N

Highest (highest)

Calculates the highest value in the data series since the first bar loaded in the chart. This indicator requires one data series (Link 1).

Highest Bar (highestb)

Calculates the number of bars has passed since the data series highest value. This includes all data loaded in the chart. This indicator requires one data series (Link 1).

Highest High Bar (hhb)

Returns the Highest High Bar over **Period**. Highest High Bar requires one series (Link 1).

Highest High Value (hhv)

Returns the Highest High Value over **Period**. Highest High Value requires one series (Link 1).

Highlight 1 Delphi (highlight)

Highlight 1 Delphi is a legacy indicator. Use Highlight Formula instead.

Highlight 1 Delphi evaluates the condition **Condition**. The condition is a boolean Delphi Script expression. If the condition evaluates to true, the result of the indicator is Link 1 plus **Offset**. If the condition evaluates to false, the result is invalid. In the condition, you can access the value of Link 1 by `Data1.value[n]`, where `n` is the number of bars ago. Highlight 1 Delphi provides two user parameters: `param3` and `param4`. To access these parameters in the condition, use `param3.real` and `param4.real`.

Highlight 2 Delphi (highlight2)

Highlight 2 Delphi is a legacy indicator. Use Highlight 2 Formula instead.

Highlight 2 Delphi is identical to Highlight 1 Delphi except it can access Link 2 in addition to Link 1. To access the value of Link 2, use `Data2.Value[n]` in the condition, where `n` is the number of bars ago.

Highlight Bar Formula (highlightbarfml)

Highlight the complete bar from High to Low if **Condition** is true. Highlight Bar Formula requires one series (Link 1).

Highlight Formula (highlightfml)

Draw a marker onto **Position** High or Low if **Condition** is true. Highlight Formula requires one series (Link 1).

Historical Test Moving Average Crossover System (sys_hist_mov_avg)

Reports a table of scanned issues with their profitability information to a report window. The profitability is returned as the indicator value.

Historical TestXAverage Crossover (sys_hist_mov_avg)

A basic moving average crossover system is implemented with full reporting. You can use it as a starting point to construct system tests and reports. Moving Avg. Crossover takes two parameters: Period Fast and Period Slow. Moving Avg. Crossover requires one series (Link 1) and reports to the first report window in the active group.

Historical Volatility (historical_vola)

Returns the annualized historical volatility. Historical Volatility requires one series (Link 1).

Historical Volatility Ratio (histvol_ratio)

Returns the ratio between two periods of annualized historical volatility. Historical Volatility Ratio requires one series (Link 1).

Inside Bar (insidebar)

Returns 1 for inside bar identified at the current bar position and 0 for non inside bar. Inside Bar requires one series (Link 1).

Keltner Channel (keltner)

Returns Keltner Channel. Keltner Channel is a channel indicator based on exponential moving average and true range of the underlying price series. Keltner Channel requires one data series (Link 1).

Keltner Channels 3 Lines (keltner3)

Keltner Channels 3 Lines is a channel indicator based on exponential moving average and true range of the underlying price series. Keltner Channel 3 returns 3 plot series. This indicator requires one data series (Link 1).

Linear Regression Channel (linregchannel)

Returns the value of SD standard deviation away from the specified linear regression projection. Linear Regression Channel requires one data series (Link 1).

Linear Regression Channel 3 Lines (linregchannel3)

Returns the complete channel of SD standard deviation away from the specified linear regression projection. Linear Regression Channel 3 Lines requires one data series (Link 1).

Linear Regression Constant (linconst)

Similar to linear regression value, it returns the constant of the approximated linear equation. Linear Regression Constant takes one parameter **Period** and requires one series (Link 1).

Linear Regression Slope (linslope)

Similar to linear regression value, it returns the slope of the approximation. Linear Regression Slope takes one parameter **Period** and requires one series (Link 1).

Linear Regression Value (linreg)

Given **Period** of bars and number of bars to project into the future (**Projection**), linear regression value returns the projected value based on the least-square approximation method. Linear Regression Value requires one series (Link 1).

Lowest (lowest)

Returns the lowest value of a data series since the first bar loaded in the chart. This indicator require one data series (Link 1).

Lowest Bars (lowestb)

Calculates the number of bars that have passed form the lowest value of a data series. This indicator require one data series (Link 1)

Lowest Low Bar (llb)

Returns the Lowest Low Bar over Period. Lowest Low Bar requires one series (Link 1).

Lowest Low Value (llv)

Returns the Lowest Low Value over Period. Lowest Low Value requires one series (Link 1).

Mass index (mass)

Returns the Mass Index. Period1 is the exponential moving average period of the daily ranges. Period2 is the smoothing factor period. Mass Index requires one series (Link 1).

Maximum (max)

Returns the largest of the two series

Mean Deviation (meandev)

Returns the statistics mean deviation over the specified **Period**. Mean Deviation requires one series (Link 1).

Median (median)

Median returns the middle value of the data series sorted over **Period** of bars. Median requires one series (Link 1).

Midpoint (midpoint)

Midpoint returns the middle point of the price range from the lowest low to the highest high over the specified **Period** of bars. Midpoint requires one series (Link 1).

Minimum (min)

Returns the smallest of the two data series

Momentum (mo)

Momentum is the raw price difference between the current value of the data series and the value of the data series **Period** ago. It is similar to the Rate of Change indicator. Momentum requires one series (Link 1).

Money Flow Index (moneyflow)

Money Flow Index is similar to RSI with volume taken into consideration. It can be interpreted in a way similar to the RSI. Money Flow Index requires one series (Link 1).

Moving Average (mov)

All Moving Average Type of a specified period. Moving Average requires one specified series (Link 1).

Moving Averages 2 Lines (mov2)

Moving averages for 2 different periods. Moving Average 2 Lines requires one specified series (Link 1).

Moving Averages 3 Lines (mov3)

Moving averages for 3 different periods. Moving Average 3 Lines requires one specified series (Link 1).

Moving Average Convergence Divergence (MACD)

MACD is a price momentum indicator. It is based on the difference between two moving averages. Usually a moving average is applied to the MACD as a signal line. MACD takes four parameters. **MA1_type** is the type of the first moving average. **MA1_period** is the period of the first moving average. **MA2_type** is the type of the second moving average. **MA2_period** is the period of the second moving average. MACD requires one series (Link 1).

Moving Avg. Crossover (mov_crossover)

This is a sample trading system that demonstrates the NeoTicker® scripting language's flexibility and capability. A basic moving average crossover system is implemented and the testing result is reported to the first report window in the active group. You can use it as a starting point to construct system tests and reports. Moving Avg. Crossover takes two parameters: **Period Fast** and **Period Slow**. Moving Avg. Crossover requires one series (Link 1) and reports to the first report window in the active group.

Multiplication (mult)

Multiplication of two series (Link 1 * Link 2).

Narrow Range (nr)

Returns 1 for narrowest range within the specified Period and 0 otherwise. Narrow Range requires one series (Link 1).

Narrow Range N Bar (nrn)

Narrow Range returns 1 when the current N bars is the narrowest range over the **Period** specified by the parameter. Narrow Range requires one series (Link 1).

NowPercent (NowPercent)

NowPercent returns in percentage (0 to 100) of the time relative to the trading time range of the hosting function window.

O-Q

On Balance Volume (obv)

On Balance Volume is a volume trend indicator. Its main usage is for detecting non-confirmation of price moves. On Balance Volume requires one series (Link 1).

Open Interest (openint)

Returns the open interest of Link 1. The value is undefined if Link 1 does not contain any open interest information, e.g. Link 1 is a stock.

Outside Bar (outsidebar)

Returns 1 for outside bar identified at the current bar position and 0 for non outside bar. Outside Bar requires one series (Link 1).

Parabolic SAR (parabolic)

Parabolic SAR should always be plotted directly onto the prices. It is a stop and reverse (SAR) model based on the concept that, in the beginning of a trend, the fluctuation tends to be high and requires larger tolerance, whereas in a mature trend, tighter stop is preferred because reversal may happen any time. Parabolic SAR takes one parameter AFactor and requires one series (Link 1).

Parabolic SAR EX (parabolicex)

Parabolic SAR EX should always plotted directly onto the prices. Parabolic SAR EX is a stop and reverse (SAR) model based on the concept that in the beginning of a trend the fluctuation tends to be high and requiring larger tolerance, while in a mature trend, tighter stop is preferred because reversal may happen any time. Parabolic SAR EX takes two parameters Factor and Max and requires one series (Link 1).

Parabolic SAR EX is the extended version of ***Parabolic SAR (parabolic)*** (on page 1997). The EX version lets you adjust the Max parameter. If Max parameter is set to the classic value of 0.2, Parabolic SAR EX returns the same value as Parablic SAR.

Percentage Bands 3 Lines (percentageband3)

Percentage Bands 3 Lines. The specific moving average of the percentage offsets are plotted. This indicator requires one series (Link 1).

Plot Count (plotcount)

Plot Count returns the number of plots in the input. Plot Count requiresone link (Link 1).

Plot Value (plotvalue)

Given the plot number. Plot Value returns the value of a specific plot. Plot Value requires one link (Link 1).

Previous Day Average (prevDAverage)

Previous Day Average is a day trading support resistance indicator that marks the average of high, low and close of previous trading day with horizontal dots. Previous Day Average works on series with minute or tick time frame. It requires one series (Link 1).

Previous Day Close (prevDClose)

Previous Day Close draws a series of horizontal dots marking the closing price level of the previous trading day. It works on series with minute or tick time frame. It requires one series (Link 1).

Previous Day High (prevDHigh)

Previous Day High draws a series of horizontal dots marking the high price level of the previous trading day. It works on series with minute or tick time frame. It requires one series (Link 1).

Previous Day Low (prevDLow)

Previous Day Low draws a series of horizontal dots marking the low price level of the previous trading day. It works on series with minute or tick time frame. It requires one series (Link 1).

Previous Day Open (prevDOpen)

Given a daily or intraday chart, this indicator draws a horizontal line marking the previous day opening price level and automatically adjust itself when the day has changed. This is a support/resistance indicators. Previous Day Open requires one series (Link 1).

Previous Day Resistance (prevDResistance)

Previous Day Resistance is a day trading technique called support/resistance based on the previous trading day's traded range. The Previous Day Resistance calculates and plots two resistance levels. It is usually used with Previous Day Average and Previous Day Support. Previous Day Resistance works on series with minute or tick time frame. It takes one parameter Type, which is either Upper or Lower. Previous Day Resistance requires one series (Link 1).

Previous Day Support (prevDSupport)

Previous Day Support is a day trading technique called support/resistance based on the previous trading day's traded range. The Previous Day Support calculates and plots the two support levels. It is usually used with Previous Day Average and Previous Day Resistance. Previous Day Support works on series with minute or tick time frame. It takes one parameter Type, which is either Upper or Lower. Previous Day Support requires one series (Link 1).

Previous Month Average (prevMAverage)

Previous Month Average is a support resistance indicator that marks the average of high, low and close of previous month with a dotted line and can automatically "jump" to the right level when the data switch over to the next month. Previous month average requires one series (Link 1).

Previous Month Close (prevMClose)

Given a daily or weekly chart, this indicator draws a series of horizontal dots marking the previous month closing price level and automatically adjust itself when the month has changed. One of the best support/resistance indicators available. Usually the complete group of Previous Month indicators are used together on the same chart. Previous Month Close requires one series (Link 1).

Previous Month High (prevMHigh)

Similar to Previous Month Close, this indicator plots the previous month high in a series of dots on daily and weekly charts. Previous Month High requires one series (Link 1).

Previous Month Low (prevMLow)

Similar to Previous Month Close, this indicator plots the previous month low in a series of dots on daily and weekly charts. Previous Month Low requires one series (Link 1).

Previous Month Resistance (prevMResistance)

Borrowed from a day trading technique called support/resistance based on the previous trading day's traded range, the Previous Month Resistance calculates and plots the two resistance levels based on previous month's traded range. It is usually used with Previous Month Support. Previous Month Resistance takes one parameter **Type**, which is either Upper or Lower. Previous Month Resistance requires one series (Link 1).

Previous Month Support (prevMSupport)

Borrowed from a day trading technique called support/resistance based on the previous trading day's traded range, the Previous Month Support calculates and plots the two support levels based on previous month's traded range. It is usually used with Previous Month Resistance. Previous Month Support takes one parameter **Type**, which is either Upper or Lower. Previous Month Support requires one series (Link 1).

Previous Week Average (prevWAverage)

Previous Week Average is a support resistance indicator that marks the average of high, low and close of previous week with a dotted line and can automatically "jump" to the right level when the data switch over to the next week. Previous week average requires one series (Link 1).

Previous Week Close (prevWClose)

Given a daily or intraday chart, this indicator draws a series of horizontal dots marking the previous week closing price level and automatically adjust itself when the week has changed. One of the best support/resistance indicators available. Usually the complete group of Previous Week indicators are used together on the same chart. Previous Week Close requires one series (Link 1).

Previous Week High (prevWHigh)

Similar to Previous Week Close, this indicator plots the previous week high in a series of dots on lower time frame charts. Previous Week High requires one series (Link 1).

Previous Week Low (prevWLow)

Similar to Previous Week Close, this indicator plots the previous week low in a series of dots on lower time frame charts. Previous Week Low requires one series (Link 1).

Previous Week Resistance (prevWResistance)

Borrowed from a day trading technique called support/resistance based on the previous trading day's traded range, the Previous Week Resistance calculates and plots the two resistance levels based on previous week's traded range. Usually used with Previous Week Support. Previous Week Resistance takes one parameter Type, which is either Upper or Lower. Previous Week Resistance requires one series (Link 1).

Previous Week Support (prevWSupport)

Borrowed from a day trading technique called support/resistance based on the previous trading day's traded range, the Previous Week Support calculates and plots the two support levels based on previous week's traded range. Usually used with Previous Week Resistance. Previous Week Support takes one parameter Type, which is either Upper or Lower. Previous Week Support requires one series (Link 1).

QC Bollinger Bands

This indicator emulate the behaviour of the "Bollinger Bands" studys from quote.com's charting application. It will automatic have band plot around a moving average.

QC Choppiness

The Choppiness Index measures the ratio of the average price range over a Length of interval to the price range over the entire length.

QC Directional Movement

In Directional Movement the +DI line shows the average percentage of the range that is upward movement; -DI line shows the percentage that is downward movement. The crossing ot these lines indicates a change in trend. The ADXR line indicates how much the instrument is trending - below 20 indicates a non-trending environment.

QC Donchain Channel

This indicator emulate the "Donchain Channel" study from quote.com charting application. Donchain channel plot the highest high and the lowest low over a specified interval Length.

QC Envelopes

Envelopes create bands at a chosen percentage plus and minus a moving average. The default percent of 10 yeilds bands at 110% and 90% of the moving average values.

QC Exponential Moving Average

Exponential Moving Average is effectively the mean of the complete data series since the beginning up to the point of calculation

QC MACD

Moving Average Convergence/Divergence provides early clues of a trend reversal. The MACD line represents the difference between a fast exponential moving average (Length1) and a slow one (Length2) - check Simple Moving Average to make this line an oscillator. The Signal Line is an exponential moving average of the MACD line. A buy signal occurs when the MACD line crosses above the Signal line; crossing below is a sell signal. A histogram plots their difference. Crossing the zero line confirms the trend.

QC Momentum

Momentum calculates the spread between the current price and the price Length intervals ago. A line of momentum calculations indicates if prices are changing at an increasing or decreasing rate. Crossing the zero line indicates a chane in trend.

QC Moving Average

This indicator emulate the behaviour of the "Moving Average" study from quote.com charting application. All parameters from the orignal study are retained except for "offset", which can be set using the "Displace" under the "Visual" tab in the edit indicator window.

QC ROC

Rate of Change performs the same calculations as the Momentum study, but expresses the spread as a percentage of the instrument's price.

QC RSI

Relative Strength Index is a countertrend Indicator that measures the ratio of upward closes to downward closes as a moving average. Fixed lines at 70 and 30 indicate overbought and oversold conditions. Traditional charting techniques like failure swings support and resistance, and divergence are used with the RSI line to identify trend reversals.

QC StochRSI

The Stochastic RSI is applying %K on to RSI and in turn apply %D smoothing on to %K.StochRSI requires one data series (Link 1).

R-Z

Random (rand)

Random takes one parameter **Range** and returns a random series of integers from 0 to **Range** - 1. Random requires one specified series (Link 1), which defines the time period when Random returns a valid value.

Range (range)

Returns the range of the current bar of the data series. Range equals to the difference between high and low of a price bar. Range requires one data series (Link 1).

Rate of Change (roc)

Rate of Change is the percentage change in value based on **Period** bars ago. It is similar to the Momentum indicator. Rate of Change requires one series (Link 1).

Reference (ref)

Returns the value of Link 1 shifted by *Offset* number of periods. A negative offset refer to values from previous periods. Positive offset is not allowed. Reference requires one data series (Link 1).

Reference Time (reftime)

Reference Time returns the open, high, low, close, volume of the user defined time range. You can also specify the number of trading days ago to return values for. Reference Time requires one data series (Link 1).

Reference Valid

Returns the value of Link 1 shifted by Offset number of valid periods. A negative offset refer to values from previous periods. Positive offset is not allowed. Reference requires one data series (Link 1).

Region Coloring (Region)

Paint region marked between Param 1 and Param 2 with user specified color. Param 1 and Param 2 can be formulas. Region Coloring requires one link. For detail on the usage of this indicator refer to ***Power Indicators Guide> Region Coloring*** (see "Region Coloring" on page 1327).

Relative Strength (rs)

Relative Strength indicator can "normalize" a source series, say prices of a stock, to a specific starting point in time (e.g. Jan 1, 1995) and plot the percentage or absolute changes since that time period. The starting point for measuring percentage change is 100% while the starting point of absolute change is 0. Relative Strength requires one specified series (Link 1).

Relative Strength Index (rsindex)

Relative Strength Index is a price momentum indicator which depends on a single data series. You can also apply RSI to an indicator to study their momentum behavior. RSI takes one parameter *Period* and requires one series (Link 1).

Relative Strength Index Modified (rsindexMod)

The modified version of RSI has a slightly different interpretation of the original RSI. It moves more smoothly than the original version. RSI Modified takes one parameter *Period* and requires one series (Link 1).

Relative Strength Multiple Data (rs_multi)

Relative Strength Multiple Data can plot upto 10 data series in relative to their price from a specific date time. Relative Strength Multiple Data requires one series (Link 1) for time frame purpose only. For detail please refer to the ***Power Indicators Guide> Relative Strength Multiple Data*** (see "Relative Strength Multiple Data" on page 1335).

Sampling Daily (SamplingDaily)

Sampling Daily returns the first value of each day from an intraday data series (Link 1) and returns invalid for the all the rest of the data points. It requires one series (Link 1).

Sampling Monthly (SamplingMonthly)

Sampling Monthly returns the first value of each day from lower time frame data series (Link 1) and returns invalid for the all the rest of the data points. It requires one series (Link 1).

Sampling Weekly (SamplingWeekly)

Sampling Weekly returns the first value of each week from lower time frame data series (Link 1) and returns invalid for the all the rest of the data points. It requires one series (Link 1).

Scan Sample Daily Report (scan_sample_daily)

Reports a table of scanned issues with their current technical indicator values to a report window.

Sign (sign)

Returns 1, 0, -1 based on the sign of Link 1. If Link 1 is positive then Sign returns 1. If Link 1 is 0, then Sign returns 0. If Link 1 is negative, then Sign returns -1.

Signal (signal)

Signal is a conditional indicator: if Link 1 equals 0, then **Signal** does not return a value; if Link 1 is not 0, Signal returns Link 2 + **Offset**.

Simple Moving Average (average)

Returns the simple moving average over **Period**. Simple Moving Average requires one series (Link 1).

Standard Deviation (stddev)

Returns the Standard Deviation assuming a sample space is given. Standard Deviation takes one parameter *Period* and requires one series (Link 1).

Standard Deviation Variable Length (stddewar)

Returns the Standard Deviation. The length of the calculation is all the data available in the series excluding invalid points.

Stochastic FastD (fastd)

Stochastic FastD indicator of a specified period. Stochastic FastD requires one specified series (Link 1).

Stochastic FastK (fastk)

Stochastic FastK indicator of a specified period. Stochastic FastK requires one specified series (Link 1).

Stochastic SlowD (slowd)

Stochastic SlowD indicator of a specified period. Stochastic SlowD requires one specified series (Link 1).

Stochastic SlowK (slowk)

Stochastic SlowK indicator of a specified period. Stochastic SlowK requires one specified series (Link 1).

Sub Series (subseries)

Sub Series returns a meta data series having only bars that meet the filter parameter condition. Sub Series requires one series (Link 1).

Subtraction (diff)

Subtracts one series from another (Link 1 - Link 2).

Summation (summation)

Returns the Summation over **Period**. Summation requires one series (Link 1).

SumXY (sumxy)

Returns the sum of x multiply by y where x is 1 to N and y is the data series value. Sum XY is useful for the calculation of linear regression and other similar type of mathematical formulas.

Swing High (swinghigh)

Swing High returns the Nth occurrence (Occur) of the swing high point of **Strength** bars over **Length** period.

Swing High Bar (swinghighbar)

Swing High Bar returns the bars ago of the Nth occurrence (Occur) of the swing high point of **Strength** bars over **Length** period.

Swing Index (swingindex)

Swing Index is a special indicator developed by Welles Wilder as described in his book New Concepts in Technical Trading Systems. It is usually used through its derivative, Accumulation Swing Index. Swing Index takes one parameter **Limit** and requires one series (Link 1).

Swing Low (swinglow)

Swing Low returns the Nth occurrence (Occur) of the swing low point of **Strength** bars over **Length** period.

Swing Low Bar (swinglowbar)

Swing Low Bar returns the bars ago of the Nth occurrence (Occur) of the swing low point of **Strength** bars over **Length** period.

Three Line Break (threelinebreak)

(Obsolete) Three Line Break Chart with custom formula for the price. Three Link Break requires one series (Link 1).

Three Line Break Ex (threelinebreakex)

(Obsolete) Three Line Break Chart with extra option to break sequence on day change. Three Link Break Ex requires one series (Link 1).

Tick (tick)

Returns the tick volume of Link 1. Tick volume is the number of data units that constitute to a bar. The value is undefined if Link 1 does not contain any tick volume information, e.g. Link 1 is an indicator.

This indicator is the same as TickVol.

TickVol (tickvol)

Returns the tick volume of Link 1. Tick volume is the number of data units that constitute to a bar. The value is undefined if Link 1 does not contain any tick volume information, e.g. Link 1 is an indicator.

This indicator is the same as Tick.

Time Frame Period (timeframeperiod)

Time Frame Period returns the bar size of the linked series. Time Frame Period requires one series (Link 1).

Time Series Forecast (tsf)

It is equivalent to linear regression value. Time Series Forecast takes two parameters **Period** and **Projection**. Time Series Forecast requires one series (Link 1).

TP Tick N (tptickn)

TP Tick Vol N (tptickvoln)

TP Bid Ask Trade N (tpbidasktraden)

TP Bid Ask Trade Vol N (tpbidasktradevoln)

These are Tick Precise Indicators. See ***Tick Precise Indicators*** (on page 996).

Trade Simulator (tradesim)

Trade Simulator indicator is a trading system that place orders you have made in the actual Trade Simulator based on the symbol of Link 1. Trade Simulator requires one series (Link 1).

Trade Simulator Portfolio (tradesim_port)

Trade Simulator Portfolio indicator is a trading system that place orders you have made in the actual Trade Simulator Portfolio based on all the symbol within the current context (chart, pattern scanner, etc.). Trade Simulator Portfolio requires one series (Link 1).

Triangle Moving Average (taverage)

Returns the triangle moving average over **Period**. Triangle Moving Average requires one series (Link 1).

Triple Exponential Moving Average (tema)

Returns TEMA as described in Technical Analysis Stocks and Commodities magazine.

TRIX (trix)

Returns TRIX, the percent rate-of-change of a triple smoothed moving average. TRIX requires using one series (Link 1).

True High (truehigh)

Returns True High of the current bar. True High requires one data series (Link 1).

True Low (truelow)

Returns True Low of the current bar. True Low requires one data series (Link 1).

True Range (truerange)

Returns the current true range. True Range requires one series (Link 1).

Typical Price (typicalprice)

Returns Typical Price ((High + Low + Close) / 3). Typical Price requires one series (Link 1).

Ultimate Oscillator (ultimate)

An oscillator based on the weighted combination of oscillator values from three different time periods. Ultimate Oscillator takes three parameters **Period1**, **Period2** and **Period3**. Ultimate Oscillator requires one series (Link 1).

Up Down Tick (UpDownTick)

Up Down Tick returns 4 plots for Up Tick, Down Tick, Up Volume, Down Volume. As most data vendors do not send these information with their historical data, the information is generated from tick data or real-time tick-by-tick update only. To get correct historical calculation, use tick replay in Chart Manager. Up Down Tick requires one series (Link 1).

Valid (valid)

Returns the valid state of Link 1 of Period ago. The value is either 1 (for valid) or 0 (for invalid). Valid requires one series (Link 1).

Valid Bar (validb)

Returns the number of bars ago that a valid bar exists. When the current bar is valid, the returning value is always 0. If there is no valid value since the start of the data series, then the returning value is negative (-1). Valid Bar requires one series (Link 1).

Valid Count (validc)

Returns the number of valid bars out of the current Period of bars. Valid Count requires one series (Link 1).

Valid Percent (validp)

Returns the percentage of number of valid bars out of the current period of bars. Valid Percent requires one series (Link 1).

Value When (valuewhen)

Returns the value of data series when the n-th most recent occurrence of the formula condition is true. ValueWhen requires one series (Link 1).

Variable Highest High Bar (varhhb)

Variable Highest High Bar returns the bars ago of the highest high value in Link 1 based on the number of bars (Link 2) to include in the calculation. Variable Highest High Bar requires two series.

Variable Linear Regression Slope (varlinregslope)

Variable Linear Regression Slope returns the slope value based on the number of bars (Link 2) to include in the linear regression calculation using least-square approximation method. Variable Linear Regression Slope requires two series.

Variable Linear Regression Value (varlinreg)

Variable Linear Regression returns the projected value based on the number of bars (Link 2) to include in the linear regression calculation using least-square approximation method. Variable Linear Regression Value requires two series.

Variable Lowest Low Bar (varllb)

Variable Lowest Low Bar returns the bars ago of the lowest low value in Link 1 based on the number of bars (Link 2) to include in the calculation. Variable Lowest Low Bar requires two series.

Variable Persist Condition (varpersistcond)

Variable Persist Condition returns 1 when Link 1 is greater than 0 for the number of bars specified in Link 2. Variable Persist Condition requires two series.

Variable RSquare (varrsquare)

Variable RSquare returns the R2 value of data in Link 1 based on the number of bars (Link 2) to include in the calculation. Variable RSquare requires two series.

Variable Summation (varsum)

Variable Summation returns the sum of values in Link 1 based on the number of bars (Link 2) to include in the summation calculation. Variable Summation requires two series.

Volatility (vola)

Volatility returns the Welles Wilder's version of volatility measurement. Volatility takes one parameter **Period** and requires one series (Link 1).

Volume (vol)

Returns the volume of Link 1. The value is undefined if Link 1 does not contain any volume information, e.g. Link 1 is an indicator.

Volume Accumulation Oscillator (vol_acc_osc)

Volume Accumulation Oscillator is a volume momentum indicator developed by Mark Chaikin. Volume Accumulated Oscillator takes one parameter **Period** and requires one series (Link 1).

Volume Distribution (voldist)

Volume Distribution returns the normal distribution region covering Mean +/- SD. You can choose from one of the calculation model. Normal model returns the historical average as the approx. mean and historical standard deviation as the approx. SD. Gaussian model returns the predicted Normal Distribution based on the distribution cumulated so far. Volume Distribution is one of the power indicators. Refer to the ***Power Indicators Guide> Volum Distribution*** (see "Volume Distribution" on page 1345) for more details. Volume Distribution requires one link (Link 1).

Volume Formula (volfml)

Color changing volume bars based on parameter Condition evaluation. Volume Formula requires one series (Link 1).

Volume Oscillator (vosc)

Volume Oscillator is the difference between two moving averages of the volume. Volume Oscillator takes two parameters: **Period Fast** and **Period Slow**. Volume Oscillator requires one series (Link 1).

Volume Rate of Change (vroc)

Similar to the Rate of Change indicator except it uses volume instead of the price value. Volume Rate of Change takes one parameter **Period** and requires one series (Link 1).

Weighted Close (wclose)

Returns Weighted Close. Weighted Close requires one series (Link 1).

Weighted Index (weighted_index)

Weighted Index returns a virtual data series based on the symbol weighting you have provided. For each weight parameter, you can specify the symbol and weighting using the format of SYM,WEIGHT where WEIGHT is any valid numeric value. You can assign up to 50 different weights. Weighted Index requires one series (Link 1) for time frame purpose only. For detail on usage of this indicator please refer to ***Power Indicators Guide> Weighted Index*** (see "Weighted Index" on page 1359).

Weighted Moving Average (waverage)

When calculating moving average, Weighted Moving Average applies weights onto the current **Period** of bars with the heaviest weight on the latest bars. Weighted Moving Average takes one parameter **Period** and requires one series (Link 1).

Weighted Spread (wspread)

Spread analysis with weighted parameters adjustment. Weighted Spread takes two parameters: **Weight1** and **Weight2**. Weighted Spread requires one series (Link 1).

Wide Range (wr)

Returns 1 for widest range within the specified Period and 0 otherwise. Wide Range requires one series (Link 1).

Wide Range N Bar (wrn)

Wide Range N Bars returns 1 when the current N bars is the widest range over the **Period** specified by the parameter. Wide Range N Bars requires one series (Link 1).

William Percent R (percentr)

William Percent R equals to 100 - Stochastic FastK indicator. William Percent R requires one specified series (Link 1).

Zig Zag (zigzag)

Zig Zag automatically draws trendlines using either percentage-driven or point-driven minimum change specified by the user. Non-confirmed move at the end of the data series is represented by special color for easy identification. Zig Zag returns the swing values in the first plot, swing bar location (bars ago) in the second plot, and the current swing direction in the third plot. Zig Zag requires one data series (Link 1).

Zig Zag High (zigzaghigh)

Zig Zag High returns the Nth High occurrence using either percentage-driven or point-driven minimum change specified by the user. Zig Zag High requires one data series (Link 1)

Zig Zag High Bar (zigzaghighbar)

Zig Zag High Bar returns the number of bars ago of the Nth High occurrence using either percentage-driven or point-driven minimum change specified by the user. Zig Zag High Bar requires one data series (Link 1)

Zig Zag Low (zigzaglow)

Zig Zag Low returns the Nth Low occurrence using either percentage-driven or point-driven minimum change specified by the user. Zig Zag Low requires one data series (Link 1)

Zig Zag Low Bar (zigziglowbar)

Zig Zag Low Bar returns the number of bars ago of the Nth Low occurrence using either percentage-driven or point-driven minimum change specified by the user. Zig Zag Low Bar requires one data series (Link 1)

Convention in this Chapter

The following sections are the indicator reference. In the Syntax part of the reference, the syntax of the language is explained under different usages.

Delphi Script

This format is used by Delphi Script and Borland Delphi. For example, the indicator Accumulation Swing Index has the following syntax:

```
ItSelf.MakeIndicator ('string', 'accswingindex', ['N'],
['n']);
```

The quoted `'string'` is the name you give to the instance of the indicator for later reference.

The capital case `N` is the link set to Accumulation Swing Index.

The lower case `n` is the numeric parameter sent to Accumulation Swing Index.

VBScript

This format is used by VBScript and Visual Basic. For example, the indicator Accumulation Swing Index has the following syntax:

```
ItSelf.MakeIndicator "string", "accswingindex",
Array("N"), Array("n")
```

The quoted `"string"` is the name you give to the instance of the indicator for later reference.

The capital case `N` is the link set to Accumulation Swing Index.

The lower case `n` is the numeric parameter sent to Accumulation Swing Index.

Formula Language

This format is used when you call an indicator in a formula. This formula is used as a parameter in a hosting indicator in time chart or pattern scanner. For example, the plot1 parameter (the parameter that can accept formula) of the formula indicator (the hosting indicator).

For example, the indicator Accumulation Swing Index has the following syntax:

```
accswingindex (dataN, n)
```

The field `dataN` references to the links of the hosting indicator, e.g. in actual use data1 references to Link 1 of the hosting indicator.

The lower case `n` is the numeric parameter sent to Accumulation Swing Index.

We do not include the optional bars ago parameter in the specification. The bars ago parameter can be optionally added as the first parameter of the indicator.

Formula Column

This format is used when you call the indicator in a formula column in quote window. For example, the indicator Accumulation Swing Index has the following syntax:

```
accswingindex (Tn, n)
```

The field `Tn` is a time frame specification. For example, `M5` represents 5-minute bars.

The lower case `n` is the numeric parameter sent to Accumulation Swing Index.

We do not include the optional bars ago parameter in the specification. The bars ago parameter can be optionally added as the first parameter of the indicator.

Other Parameters

We use clColor to represent color parameter. A color parameter is either a pre-defined constant like clRed, or an integer value representing the color (decimal representation of the hex number BBGGRR, where BB represents blue, GG represents green, RR represents red).

Absolute Value (abs)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'abs', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "abs", Array("N"), Array("")
```

Formula Language*

```
abs (dataN)
```

Formula Column*

```
abs (Tn)
```

Definition

Returns the value of the data series without the sign of each value. Absolute value requires one specified series (Link 1) .

Examples

Delphi Script

```
function abstest : double;
var myabs;
begin
  ItSelf.MakeIndicator ('absstr', 'abs', ['1'], ['']);
  myabs := ItSelf.Indicator ('absstr');
  result := myabs.Value [0];
end;
```

VBScript

```
function abstest ( )
  dim mybas

  ItSelf.MakeIndicator "asbstr", "abs", Array("1"), Array("")
  myabs = ItSelf.Indicator ("asbstr")
  abstest = myabs.Value (0)
end function
```

Formula Language*

```
plot1 := abs(data1);
```

Formula Column*

```
abs(s13)
```

*Note formula and formula language have a built-in abs function which will take precedence over this indicator. For detail please see ***Generic Functions*** (on page 295).

Accumulation Swing Index (accswingindex)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'accswingindex', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "accswingindex", Array("N"),
Array("n")
```

Formula Language

```
accswingindex (dataN, n)
```

Formula Column

```
accswingindex (Tn, n)
```

Definition

Cumulative form of Swing Index. Accumulation Swing Index takes one parameter Limit and requires one series (Link 1).

Examples

Delphi Script

```
function accswingindextest : double;
var myaccswingindex;
begin

    Itself.MakeIndicator ('accswingindexstr', 'accswingindex',
['1'], ['9999']);
    myaccswingindex := Itself.Indicator ('accswingindex');
    result := myaccswingindex.Value [0];

end;
```

VBScript

```
function accswingindextest ()

  dim myaccessingindex

  ItSelf.MakeIndicator "accswingindexstr", "accswingindex",
Array("1"), Array("9999")
```

```
  myaccswingindex = Itself.Indicator ("accswingindex")
  accswingindextest = myaccswingindex.Value (0);
end function
```

Formula Language

```
plot1 := accswingindex (data1, 9999);
```

Formula Column

```
accswingindex (m10, 9999)
```

Accumulate (accum)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'accum', ['1'], ['']);
```

VBScript

```
Itself.MakeIndicator "string", "accum", Array("1"), Array("")
```

Formula Language

```
accum (dataN, TypeStr, N, N)
```

Formula Column

```
accum (Tn, TypeStr, N, N)
```

Definition

Accumulate is an accumulator : if Link 1 is valid, it will accumulate either the value of Link 1 or count it base on the default values. This indicator reset to 0 on every invalid bar.

Examples

Delphi Script

```
function accumtest : double;
var myaccum : variant;
begin

  Itself.MakeIndicator ('accumstr', 'accum', ['1'], ['Value',
'1', '0']);
  myaccum := ItSelf.Indicator ('accumstr');
  result := myaccum.Value [0];

end;
```

VBScript

```
function accumtest ()
dim myaccum

  ItSelf.MakeIndicator "accumstr", "accum", Array("1"), Array
("Value","1","0")
  myaccum = ItSelf.Indicator ("accumstr")
  accumtest = myaccum.Value (0)
```

```
end function
```

Formula Language

```
plot1 := accum(data1, Value, 1, 0);
```

Formula Column

```
accum(m1, Value, 1, 0)
```

Accumulation Distribution (ad)

Syntax

Delphi Script

```
ItSelf.MakeInidcator ('string', 'ad', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "ad", Array("N"), Array("")
```

Formula Language

```
ad(dataN)
```

Formula Column

```
ad(Tn)
```

Definition

Returns Accumulation Distribution based on the volume of the data series. Accumulation Distribution requires one series (Link 1).

Examples

Delphi Script

```
function adtest : double;
var myad : variant;
begin

    ItSelf.MakeIndicator ('adstr', 'ad', ['1'], ['']);
    myad := ItSelf.Indicator ('adstr');
    result := myad.value [0];

end;
```

VBScript

```
function adtest ()
   dim myad

    ItSelf.MakeIndictor "adstr", "ad", Array("1"), Array("")
    myad = ItSelf.Indicator ("adstr")
    adtest = myad.Value (0)

end function
```

Formula Language

```
plot1 := ad(data1);
```

Formula Column

```
ad(1, m15)
```

Accumulation Distribution Custom (ad_custom)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'ad_custom', ['N','N'], ['']);
```

VBScript

```
Itself.MakeIndicator "string", "ad_custom", Array("N", "N"),
Array("")
```

Formula Language

```
ad_custom (dataN, dataN);
```

Formula Column

Not supported.

Definition

Returns Accumulation Distribution using the second series as the volume information. Accumulation Distribution requires two series.

Examples

Delphi Script

```
function ad_customtest : double;
var myad_custom : variant;
begin

    if not Data1.Valid [0] then
    begin
        ItSelf.Success := false;
        exit;
    end;

    ItSelf.MakeIndicator ('adstr', 'ad_custom', ['1', '1.v'],
['']);
    myad_custom := ItSelf.Indicator ('adstr');

    result := myad_custom.Value [0];
end;
```

VBScript

```
function ad_customtest ()
    dim myad_custom

    if not Data1.valid (0) then
        ItSelf.Success = false
        exit function
    end if

    ItSelf.MakeIndicator "ad_customstr","ad_custom",Array("1",
"1.v"), Array("")
    myad_custom = ItSelf.Indicator ("ad_customstr")

    ad_csutomtest = myad_custom.value (0)

end function
```

Formula Language

```
ad_custom (data1,data1.v)
```

Adaptive Moving Average (ama)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'ama', ['N'], ['Period']);
```

VBScript

```
ItSelf.MakeIndicator "String", "ama", Array("N"),
Array("Period")
```

Formula language

```
ama( data1, Period);
```

Formula Column

```
ama(Tn, Period)
```

Definition

A special exponential moving average using an Efficiency Ratio to modify the smoothing constant which ranges from fastest to slowest. Adaptive Moving Average requires one series (Link 1).

Examples

Delphi Script

```
function dp_amaxover : double;
var xabove_signal, xbelow_signal : variant;
begin
   if not data1.valid [0] then
   begin
      itself.success [0] := false;
      exit;
   end;

   itself.makeindicator ('sama', 'ama', ['1'], [param1.str]);
   itself.makeindicator ('lama', 'ama', ['1'], [param2.str]);

   xabove_signal := itself.makeindicator ('myxabove', 'xabove',
                                                         ['sama',
'lama'], ['']);
   xbelow_signal := itself.makeindicator ('myxbelow', 'xbelow',
                                                         ['sama',
'lama'], ['']);
```

```
   if xabove_signal.value [0] > 0 then
      result := 1
   else if xbelow_signal.value [0] > 0 then
      result := -1
   else
      result := 0;

end;
```

VBScript

```
function vb_amaxover()
dim xabove_signal, xbelow_signal

   if not data1.valid (0) then
      itself.success = false
      exit function
   end if

   itself.makeindicator "sama", "ama", Array("1"),
Array(param1.str)
   itself.makeindicator "lama", "ama", Array("1"),
Array(param2.str)

   xabove_signal = itself.makeindicator ("myxabove", "xabove",
_
                                         Array("sama", "lama"),
Array(""))
   xbelow_signal = itself.makeindicator ("myxbelow", "xbelow",
_
                                         Array("sama", "lama"),
Array(""))

   if xabove_signal.value (0) > 0 then
      vb_amaxover = 1
   elseif xbelow_signal.value (0) > 0 then
      vb_amaxover = -1
   else
      vb_amaxover = 0
   end if

end function
```

Formula Language

```
xabove_signal := xabove(ama(data1, param1), ama(data1,
param2));
xbelow_signal := xbelow(ama(data1, param1), ama(data1,
param2));

plot1 := if (xabove_signal > 0, 1, if(xbelow_signal > 0, -1,
0));
```

Formula Column

```
if (fml(M1,"xabove(ama(M1, 10), ama(M1, 25)") > 0,
1,
if(fml(M1,"xabove(ama(M1, 10), ama(M1, 25)") > 0, -1, 0))
```

Addition (add)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'add', ['N','N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "add", Array("N","N"), Array("")
```

Formula Language

```
add(dataN, dataN);
```

Formula Column

Not supported.

Definition

Addition of two series (Link 1 + Link 2).

Examples

Delphi Script

```
function addtest : double;
var myadd;
begin

  if not data1.valid [0] then
  begin
    ItSelf.Success := false;
    exit;
  end;

  ItSelf.MakeIndicator ('mov10', 'average', ['1'], ['10']);
  ItSelf.MakeIndicator ('mov20', 'average', ['1'], ['20']);

  ItSelf.MakeIndicator ('somestr', 'add', ['mov10', 'mov20'],
['']);
  myadd := ItSelf.MakeIndicator ('smoestr');

  result := myadd.value [0];
end;
```

VBScript

```
function addtest ()
    dim myadd

    if not data1.valid (0) then
        Itself.success = false
        exit function
    end if

    ItSelf.MakeIndicator "mov10", "average", Array("1"),
Array("10")
    ItSelf.MakeIndicator "mov20", "average", Array("1"),
Array("20")

    ItSelf.MakeIndicator "somestr", "add", Array("mov10",
"mov20"), Array("")
    myadd = ItSelf.MakeIndicator ("somestr")

    addtest = myadd.Value (0)

end function
```

Formula Language

```
plot1 := add(average(data1, 10), average(data1, 20));
```

Aroon Down (aroondown)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'aroondown', ['N'],
['Period']);
```

VBScript

```
ItSelf.MakeIndicator "string", "aroondown", Array("N"),
Array("Period")
```

Formula Language

```
aroondown (Data1, Period);
```

Formula Column

```
aroondown(Tn, Period)
```

Definition

Aroon Down for a given time period is calculated by determining how much time elapsed between the start of the time period and the point at which the lowest closing price during that time period occurred. Aroon Down requires one data series (Link 1).

Examples

Delphi Script

```
function dp_ardowntrend : double;
var myaroondown, myaroonosc : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   myaroondown := ItSelf.MakeIndicator ('myaroondown',
'aroondown',
                                         ['1'], [param1.str]);
   myaroonosc  := ItSelf.MakeIndicator ('myaroonosc',
'aroonosc',
                                         ['1'], [param1.str]);

   if (myaroondown.value [0] < 30) and (myaroonosc.value [0] <
0) then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low  [0];
      itself.plot [3] := clRed;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function vb_ardowntrend()
dim myaroondown, myarronosc

    if not data1.valid (0) then
       itself.successall = false
       exit function
    end if

    myaroondown = ItSelf.MakeIndicator ("myaroondown", 
"aroondown", _
                                          Array("1"), 
Array(param1.str))
    myaroonosc  = ItSelf.MakeIndicator ("myaroonosc", 
"aroonosc", _
                                          Array("1"), 
Array(param1.str))

    if (myaroondown.value (0) < 30) and (myaroonosc.value (0) < 
0) then
       itself.plot (1) = data1.high (0)
       itself.plot (2) = data1.low  (0)
       itself.plot (3) = clRed
    else
       itself.successall = false
    end if
end function
```

Formula Language

```
isdowntrend := ((aroondown(data1, param1) < 30) and
                (aroonosc(data1, param1) < 0));

plot1 := if(isdowntrend > 0, high, 0);
plot2 := if(isdowntrend > 0, low, 0);
plot3 := if(isdowntrend > 0, clRed, 0);

success1 := if(isdowntrend > 0, 1, 0);
success2 := if(isdowntrend > 0, 1, 0);
success3 := if(isdowntrend > 0, 1, 0);
```

Formula Column

```
if ((aroondown(M1, 10) < 30) and (aroonosc(M1, 10) < 0), 1, 0)
```

Aroon Oscillator (aroonosc)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'aroonosc', ['N'], ['Period']);
```

VBScript

```
ItSelf.MakeIndicator "string", "aroonosc", Array("N"),
Array("Period")
```

Formula Language

```
aroonosc(Data1, Period);
```

Formula Column

```
aroonosc(Tn, Period)
```

Definition

The Aroon Oscillator signals an upward trend is underway when it is above zero and a downward trend is underway when it falls below zero. The farther away the oscillator is from the zero line, the stronger the trend. Aroon oscillator requires one data series (Link 1).

Examples

Delphi Script

```
function dp_arosctrend : double;
var myaroonup, myaroondown, myaroonosc : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   myaroonup   := itself.makeindicator ('myaroonup', 'aroonup',
                                         ['1'], [param1.str]);
   myaroondown := itself.makeindicator ('myaroondown',
'aroondown',
                                         ['1'], [param1.str]);
   myaroonosc  := itself.makeindicator ('myaroonosc',
'aroonosc',
                                         ['1'], [param1.str]);
   if (myaroonup.value [0] > 70) and (myaroonosc.value [0] > 0)
then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low  [0];
      itself.plot [3] := clGreen;
   end
   else if (myaroondown.value [0] < 30) and (myaroonosc.value
[0] < 0) then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low  [0];
      itself.plot [3] := clRed;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function vb_arosctrend()
dim myaroonup, myaroondown, myaroonosc

   if not data1.valid (0) then
      itself.successall = false
      exit function
   end if

   myaroonup   = itself.makeindicator ("myaroonup", "aroonup",
_
                                          Array("1"),
Array(param1.str))
   myaroondown = itself.makeindicator ("myaroondown",
"aroondown", _
                                          Array("1"),
Array(param1.str))
   myaroonosc  = itself.makeindicator ("myaroonosc",
"aroonosc", _
                                          Array("1"),
Array(param1.str))

   if (myaroonup.value (0) > 70) and (myaroonosc.value (0) > 0)
then
      itself.plot(1) = data1.high (0)
      itself.plot(2) = data1.low (0)
      itself.plot(3) = clGreen
   elseif (myaroondown.value (0) < 30) and (myaroonosc.value
(0) < 0) then
      itself.plot(1) = data1.high (0)
      itself.plot(2) = data1.low  (0)
      itself.plot(3) = clRed
   else
      itself.successall = false
   end if
end function
```

Formula Language

```
isuptrend   := ((aroonup(data1, param1) > 70) and
(aroonosc(data1, param1) > 0));
isdowntrend := ((aroondown(data1, param1) < 30) and
(aroonosc(data1, param1) < 0));

plot1 := if((isuptrend > 0) or (isdowntrend > 0), high, 0);
plot2 := if((isuptrend > 0) or (isdowntrend > 0), low, 0);
plot3 := if(isuptrend > 0, clGreen, if(isdowntrend >0, clRed,
0));

success1 := if((isuptrend > 0) or (isdowntrend > 0), 1, 0);
success2 := if((isuptrend > 0) or (isdowntrend > 0), 1, 0);
success3 := if((isuptrend > 0) or (isdowntrend > 0), 1, 0);
```

Formula Column

```
if ((aroonup (M1, 10) > 70) and (aroonosc(M1, 10) > 0), 1,
if((aroondown(M1,10) < 30) and (aroonosc(M1, 10) < 0), -1, 0))
```

Aroon Up (aroonup)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'aroonup', ['N'], ['Period']);
```

VBScript

```
ItSelf.MakeIndicator "string", "aroonup", Array("N"),
Array("Period")
```

Formula Language

```
aroonup(Data1, Period);
```

Formula Column

```
aroonup(Tn, Period)
```

Definition

Aroon up for a given time period is calculated by determining how much time elapsed between the start of the time period and the point at which the highest closing price during that time period occurred. Aroon up requires one specified series (Link 1).

Examples

Delphi Script

```
function dp_aruptrend : double;
var myaroonup, myaroonosc : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   myaroonup  := itself.makeindicator ('myaroonup', 'aroonup',
                                                  ['1'],
[param1.str]);
   myaroonosc := itself.makeindicator ('myaroonosc',
'aroonosc',
                                                     ['1'],
[param1.str]);

   if (myaroonup.value [0] > 70) and (myaroonosc.value [0] > 0)
```

```
then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low [0];
      itself.plot [3] := clGreen;
   end
   else
      itself.successall := false;

end;
```

VBScript

```
function vb_aruptrend()
dim myaroonup, myaroonosc

   if not data1.valid (0) then
      itself.successall = false
      exit function
   end if

   myaroonup = itself.makeindicator ("myaroonup",   "aroonup",
_
                                         Array("1"),
Array(param1.str))
   myaroonosc = itself.makeindicator ("myaroonosc", "aroonosc",
_
                                         Array("1"),
Array(param1.str))

   if (myaroonup.value (0) > 70) and (myaroonosc.value (0) > 0)
then
      itself.plot (1) = data1.high (0)
      itself.plot (2) = data1.low (0)
      itself.plot (3) = clGreen
   else
      itself.successall = false
   end if

end function
```

Formula Language

```
isup_part1 := aroonup(data1, param1) > 70;
isup_part2 := aroonosc(data1, param1) > 0;

isuptrend := ((isup_part1 > 0) and (isup_part2 > 0));

plot1 := if(isuptrend > 0, h, 0);
plot2 := if(isuptrend > 0, l, 0);
plot3 := if(isuptrend > 0, clGreen, 0);

success1 := if(isuptrend > 0, 1, 0);
success2 := if(isuptrend > 0, 1, 0);
success3 := if(isuptrend > 0, 1, 0);
```

Formula Column

```
if ((aroonup (M1, 10) > 70) and (aroonosc(M1, 10) > 0), 1,
if((aroonup(M1,10) > 70) and (aroonosc(M1, 10) > 0), -1, 0))
```

Auto Trendline (auto_trendline)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'auto_trendline', ['N'], ['n',
'n', 'n', 'clColor', 'clColor']);
```

VBScript

```
ItSelf.MakeIndicator "string", "auto_tredнline", Array("N"),
Array("n", "n", "n", "clColor", "clColor")
```

Formula Language

```
auto_tendline (dataN, n, n, n, clColor, clColor);
```

Formula Column

Not supported.

Definition

Auto Trendline draws 2 trend lines based on the latest 2 swing highs and 2 swing lows. Auto Trendline requires one series (Link 1).

Examples

Delphi Script

```
function trendlinetest :double;
var myauto_trendline : variant;
begin

      ItSelf.MakeIndicator ('tlstr', 'auto_trendline', ['1'],
['8', '8', '100', 'clRed', 'clBlue']);
      myauto_trendline := ItSelf.Indicator ('tlstr');

      result := myauto_trendline.Value [0];
end;
```

VBScript

```
function trendlinetest ()
      dim myauto_trendline

      ItSelf.MakeIndicator "tlstr", "auto_trendline",
Array("1"), Array("8", "8", "100", "clRed", "clBlue")
      myauto_trendline = ItSelf.Indicator ("tlstr")

      trendline := myauto_trendline.value (0)

end function
```

Formula Language

```
plot1 := auto_trendline (data1, 8, 8, 100, clRed, clBlue);
```

Average Directional Movement Index (adx)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'adx', ['N'], ['n']) ;
```

VBScript

```
ItSelf.MakeIndicator "string", "adx", Array("N"), Array ("n")
```

Formula Language

```
adx(dataN, n)
```

Formula Column

```
adx (Tn, n)
```

Definition

A smoothed version of DMI. ADX takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script, Example 1

```
function adxtest : double;
var : adxa, adxb;
begin

    If not Data1.valid [0] then
    begin
        ItSelf.success := false;
        exit;
    end;

    ItSelf.MakeIndicator ('adx1', 'adx', ['1'], ['14']);
    adxa := ItSelf.Indicator ('adx1');

    ItSelf.MakeIndicator ('adx2', 'adx', ['2'], ['14']);
    adxb := ItSelf.Indicator ('adx2');

    result := adxa.Value [0] - adxb.Value [0];
end;
```

Delphi Script, Example 2

```
function dp_ADXR : double;
var myadx : variant;
begin
   if not data1.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   myadx := itself.MakeIndicator ('myadx', 'ADX', ['1'],
[param1.str]);

   result := (myadx.Value [0] + myadx.value [13])/2;
end;
```

VBScript, Example 1

```
function adxtest ()
       dim adxa, adxb

       if not Data1.valid (0) then
         ItSelf.success = false
         exit function
       end

       ItSelf.MakeIndicator "adx1", "adx", Array("1"),
Array("14")
       adxa = ItSelf.Indicator ("adx1")

       ItSelf.MakeIndicator "adx2", "adx", Array("2"),
Array("14")
       adxb = ItSelf.Indicator ("adx2")

       adxtest = adxa.value (0) - adxb.value (0)
end function
```

VBScript, Example 2

```
function vb_ADXR()
dim myadx

  if not data1.valid (0) then
     itself.success = false
     exit function
  end if

  myadx = itself.makeindicator ("myadx", "ADX", Array("1"),
Array(param1.str))

   vb_ADXR = (myadx.value (0) +myadx.value (14))/2
end function
```

Formula Language, Example 1

```
plot := adx(data1, 14) - adx(data2, 14);
```

Formula Language, Example 2

```
plot1 := (adx(Data1, param1)+ adx(14, Data1, param1))/2;
```

Formula Column, Example 1

```
(ADX(M1, 14)+ADX(14, M1, 14))/2
```

Formula Column, Example 2

```
adx([IBM]M5, 14) - adx([INTC]M5, 14)
```

Average Range (avgrange)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'avgrange', 'N', 'n');
```

VBScript

```
ItSelf.MakeIndicator "string", "avgrange", Array("N"),
Array("n")
```

Formula Language

```
avgrange (dataN, n);
```

Formula Column

```
avgrange (Tn, n)
```

Definition

A smoothed version of DMI. ADX takes one parameter Period and requires one series (Link 1).

Example

Delphi Script

```
function avgrangeplus10 : double;
var myavgrange : variant;
begin

    if not Data1.Valid [0] then
    begin
        ItSelf.Success := false;
        exit;
    end;

    ItSelf.MakeIndicator ('avgrstr', 'avgrange', ['1'],
['10']);
    myavgrange := ItSelf.Indicator ('avgrstr');

    if not myadx.valid [0] then
    begin
        ItSelf.Success := false;
        exit;
    end;
    result := myavgrange.value [0] + 10;
end;
```

VBScript

```
function avgrangeplus10 ()
   dim myaavgrange

    if not Data1.Valid (0) then
        ItSelf.Success = false
        exit function
    end if

    ItSelf.MakeIndicator "avgrstr", "avgrange", Array("1"),
Array("10")
    myavgrange = ItSelf.Indicator ("avgrange")

    if not myavgrange.valid (0) then
        ItSelf.Success = false
        exit function
    end if

    adxplus10 = myavgrange + 10

end function
```

Formula Language

```
plot := avgrange(data1, 10) + 10;
```

Formula Column

```
avgrange(m1, 10) + 10
```

Average True Range (avgtruerange)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'avgtruerange', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "avgtruerange", Array("N"),
Array("n")
```

Formula Language

```
avgtruerange (dataN, n);
```

Formula Column

```
avgtruerange (Tn, n)
```

Definition

Returns the average value of all the true range values over the given Period of bars. Average True Range requires one series (Link 1).

Example

Delphi Script

```
function avgtruerangetest : double;
var myavgtruerange : variant;
begin

      if not Data1.Valid [0] then
      begin
        ItSelf.Success := false;
        exit;
      end;

      ItSelf.MakeIndicator ('avgtruerangestr', 'avgtruerange',
['1'], ['10']);
      myavgtruerange := ItSelf.Indicator ('avgtruerangestr');


      if myavgtruerange.value [0] > 0.3 then
         result := myavetruerange.Value [0]
      else
         ItSelf.Success := false;
```

```
end;
```

VBScript

```
function avgtruerangetest ()
      dim myavgtruerange

      if not Data1.Valid (0) then
        ItSelf.Success = false
        exit function
      end if

      ItSelf.MakeIndicator "avgtruerangestr", "avgtruerange",
Array("1"), Array("10")
      myavgtruerange = ItSelf.Indicator ("avgtruerangestr")

      if myavgtruerange.Value (0) > 0.3 then
         avgtruerangetest = myavgtruerange.Value (0)
      else
         ItSelf.Success = false
      end if

end function
```

Formula Language

```
a := avgtruerange ( data1, 10);
plot1 := a;
Success1 := a > 0.3;
```

Formula Column

```
if (avgtruerange(m1, 10) > 0.3, avgtruerange(m1, 10), 0)
```

Barsago By Time (barsagobytime)

This manual page is yet to be completed. If you need help on this indicator, contact support@tickquest.com.

Definition

Barsago By Time returns the number of bars ago for the specified time and days ago input. Barsago By Time requires one data series (Link 1).

Bars Since (barssince)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'barssince', ['N'],
['Formula']);
```

VBScript

```
ItSelf.MakeIndicator "string", "barssince", Array("N"),
Array("Formula")
```

Formula Language

```
Barssince (Data1, “Formula”);
```

Formula Column

```
Barssince (Tn, “Formula”)
```

Definition

BarsSince calculates the number of periods that have passed since the formula condition was true. This indicator require one data series (Link 1).

Examples

Delphi Script

```
function dp_barsince : double;
var lastgap : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   lastgap := itself.Makeindicator ('mybarssince', 'barssince',
['1'], ['l> h(1)']);

   if lastgap.value [0] = param1.int then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low [0];
      itself.plot [3] := clLime;
   end
   else
     itself.successall := false;
end;
```

VBScript

```
function vb_barssinceex()
dim lastgap
   if not data1.valid (0) then
      itself.successall = false
      exit function
   end if

   lastgap = itself.makeindicator ("mybarssince", "barssince",
_
                                       Array("1"), Array("l >
h(1)"))
   if lastgap.value (0) = param1.int then
      itself.plot (1) = data1.high (0)
      itself.plot (2) = data1.low (0)
      itself.plot (3) = clLime
   else
      itself.successall = false
   end if
end function
```

Formula Language

```
lastgap := (param1 = barssince (Data1, "l > h(1)"));

plot1 := if(lastgap > 0, high, 0);
plot2 := if(lastgap > 0, low, 0);
plot3 := if(lastgap > 0, clLime, 0);

success1 := if (lastgap > 0, 1, 0);
success2 := if (lastgap > 0, 1, 0);
success3 := if (lastgap > 0, 1, 0);
```

Formula Column

```
if (barssince(M1, "l > h(1)") = 10, 1, 0)
```

Bars Since Nth (barssincen)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'barssincen', ['N'], ['Period',
'Formula']);
```

VBScript

```
Itself.MakeIndicator "string", "barssincen", Array("N"),
Array("Period", "Formula")
```

Formula Language

```
barssincen (Data1, Period, "Formula");
```

Formula Column

```
barssincen (Tn, Period, "Formula")
```

Definition

BarsSinceN calculates the number of periods that have passed since the N-th most recent formula condition was true. This indicator require one data series (Link 1).

Examples

Delphi Script

```
function dp_barssincen : double;
var lastgapdown : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   lastgapdown := itself.makeindicator ('lgd', 'barssinceN',
                                                      ['1'], ['1', 'h<
l(1)']);

   if lastgapdown.value [0] = param1.int then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low [0];
      itself.plot [3] := clRed;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
   function vb_barssincen()
   dim lastgapdown

      if not data1.valid (0) then
         itself.successall = false
         exit function
      end if

      lastgapdown = itself.makeindicator ("lgd", "barssincen", _
                                                  Array("1"),
   Array("1", "h<l(1)"))

      if lastgapdown.value (0) = param1.int then
         itself.plot (1) = data1.high (0)
         itself.plot (2) = data1.low (0)
         itself.plot (3) = clRed
      else
         itself.successall = false
      end if
   end function
```

Formula Language

```
   lastgapdown := barssincen(data1, 1, "h< l(1)");

   plot1 := if(lastgapdown = param1, high,  0);
   plot2 := if(lastgapdown = param1, low,   0);
   plot3 := if(lastgapdown = param1, clRed, 0);

   success1 := if(lastgapdown = param1, 1, 0);
   success2 := if(lastgapdown = param1, 1, 0);
   success3 := if(lastgapdown = param1, 1, 0);
```

Formula Column

```
   if(barssincen(M1, 1, "h< l(1)")=10, 1, 0)
```

Backtest Basic (backtest)

Backtest Basic is a legacy indicator.

Bollinger Band (bband)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'bband', ['N'], ['n', 'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "bband", Array("N"),
Array("n","n")
```

Formula Language

```
bband (dataN, n, n);
```

Formula Column

```
bband(Tn, n, n)
```

Definition

Given the Period and Offset in terms of multiple of the standard deviation, returns the Bollinger Band value. Bollinger Band requires one series (Link 1).

Examples

Delphi Script

```
function bband_cross : double;
var bbandshort, bbandlong : variant
begin
     if not data1.valid [0] then
     begin
        ItSelf.Success := false;
        exit;
     end;

     ItSelf.MakeIndicator ('bbands', 'bband', ['1'],
['10','1']);
     bbandshort := ItSelf.Indicator ('bbands');

     ItSelf.MakeIndicator ('bbandl', 'bband', ['1'], ['20',
'1']);
     bbandlong := ItSelf.Indicator ('bbandl');

     if bbandshort.valid [0] and bbandlong.valid [0] then
         result := bbandshort.value [0] - bbandlong.value [0]
     else
         ItSelf.Success := false;
end;
```

VBScript

```
function bband_cross ()
      dim bbandshort, bbandlong

      if not data1.valid (0) then
        ItSelf.Success := false
        exit function
      end if

      ItSelf.MakeIndicator "bbands", "bband", Array("1"),
Array("10", "1")
      bbandshort := ItSelf.Indicator ("bbands")

      ItSelf.MakeIndicator "bbandl", "bband", Array("1"),
Array("20", "1")
      bbandlong := ItSelf.Indicator ("bbandl")

      if bbandshort.valid (0) and bbandlong.vaild (0) then
        bband_cross = bbandshort.value (0) - bbandlong.value
(0)
      else
        ItSelf.Success = false
      end if
end function
```

Formula Language

```
plot1 := bband(data1, 10, 1) - bband(data1, 20, 1);
```

Formula Column

```
bband(m5, 10, 1) - bband(m5, 20, 1)
```

Bollinger Bands 3 Lines (bbands3)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'bbands3', ['N'], ['n', 'n']);
```

VBScript

```
ItSelf.MakeIndicatro "string", "bbands3", Array("N"),
Array("n", "n")
```

Formula Language

```
bbands3(dataN, n, n)
```

Formula Column

```
bbands3(Tn, n, n)
```

Definition

Given the Period, and Offset in terms of multiple of the standard deviation, returns 3 Bollinger band values. Bollinger Bands 3 Lines requires one series (Link 1).

Examples

Delphi Script

```
function bbandswmov : double;
var mybband3, mymov :variant;
begin

      if not Data1.valid [0] then
      begin
          ItSelf.SuccessEx [1] := false;
          ItSelf.SuccessEx [2] := false;
          ItSelf.SuccessEx [3] := false;
          ItSelf.SuccessEx [4] := false;
          exit;
      end;

      ItSelf.MakeIndicator ('xbband', 'bbands3', ['1'], ['10',
'1']);
      mybband3 := ItSelf.Indicator ('xbband');

      ItSelf.MakeIndicator ('ymov', 'xaverage', ['1'], ['10']);
      mymov := ItSelf.Indicator ('ymov');
      if mybband3.valid [0] and mymov.valid [0] then
      begin
          ItSelf.plot [1] := mybband3.ValueEx (1, 0);
          ItSelf.plot [2] := mybband3.ValueEx (2, 0);
          ItSelf.plot [3] := mybband3.ValueEx (3, 0);
          ItSelf.plot [4] := mymov.Value(0);
      end
      else
      begin
          ItSelf.SuccessEx [1] := false;
          ItSelf.SuccessEx [2] := false;
          ItSelf.SuccessEx [3] := false;
          ItSelf.SuccessEx [4] := false;
      end;

end;
```

VBScript

```
function bbandswmov ()
      dim mybband3, mymov

      if not Data1.Valid (0) then
         ItSelf.SuccessEx (1) = false
         ItSelf.SuccessEx (2) = false
         ItSelf.SuccessEx (3) = false
         ItSelf.SuccessEx (4) = false
         exit function
      end if

      ItSelf.MakeIndicator "xbband3", "bbands3", Array("1"),
Array("10", "1")
      mybband3 = ItSelf.Indicator ("xbband3")

      ItSelf.MakeIndicator "ymov", "average", Array("1"),
Array("10")
      mymov = ItSelf.Indicator ("ymov")

      if mybband3.valid (0) and mymov.valid (0) then
         ItSelf.plot (1) = mybband3.ValueEx (1, 0)
         ItSelf.plot (2) = mybband3.ValueEx (2, 0)
         ItSelf.plot (3) = mybband3.ValueEx (3, 0)
         ItSelf.plot (4) = mymov.Value (0)
      else
         ItSelf.SuccessEx (1) = false
         ItSelf.SuccessEx (2) = false
         ItSelf.SuccessEx (3) = false
         ItSelf.SuccessEx (4) = false
      end if

end function
```

Formula Language

```
plot1 := bbands3.plot1(data1, 10, 1);
```

```
plot2 := bbands3.plot2(data1, 10, 1);
plot3 := bbands3.plot3(data1, 10, 1);
plot4 := average(10);
```

Formula Column

```
bbands3.plot1(m3,10, 1)
```

Bollinger Bands 5 Lines (bbands5)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'bbands5', ['N'], ['n', 'n',
'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "bbands5", Array("N"),
Array("n", "n", "n")
```

Formula Language

```
bbands5( dataN, n, n, n );
```

Formula Column

```
bbands5( Tn, n, n, n )
```

Definition

Given the Period, Inner Offset and Outer Offset in terms of multiple of the standard deviation, returns 5 Bollinger band values. Bollinger Bands 5 Lines requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mybband5 : variant;
begin

     if not data1.valid [0] then
     begin
       ItSelf.SuccessEx [1] := false;
       ItSelf.SuccessEx [2] := false;
       ItSelf.SuccessEx [3] := false;
       exit;
     end;

     Itself.MakeIndicator ('mybband5', 'bbands5', ['1'],
                          [param1.str, param2.str,
param3.str]);
     mybband5 := ItSelf.Indicator ('mybband5');
     if not mybband5.ValidEx [1, 0] then
     begin
       ItSelf.SuccessEx [1] := false;
       ItSelf.SuccessEx [2] := false;
       ItSelf.SuccessEx [3] := false;
       exit;
     end;

     if (Data1.High [0] > mybband5.ValueEx [5, 0]) and
        (Data1.Low [0] < mybband5.ValueEx [2, 0]) then
     begin
       ItSelf.Plot [1] := Data1.High [0];
       ItSelf.Plot [2] := Data1.Low [0];
       ItSelf.Plot [3] := tq_Str2Color(clWhite);
     end
     else
     begin
       if data1.high [0] > mybband5.ValueEx [5, 0] then
       begin
         ItSelf.Plot [1] := Data1.High [0];
         ItSelf.Plot [2] := Data1.Low [0];
         ItSelf.Plot [3] := tq_Str2Color(clGreen);
       end
       else if data1.Low [0] < mybband5.ValueEx [2, 0] then
       begin
         ItSelf.Plot [1] := Data1.High [0];
         ItSelf.Plot [2] := Data1.Low [0];
         ItSelf.Plot [3] := tq_Str2Color(clRed);
       end
       else
       begin
         ItSelf.SuccessEx [1] := false;
         ItSelf.SuccessEx [2] := false;
         ItSelf.SuccessEx [3] := false;
       end;
    end;
end;
```

VBScript

```
function testvb()
dim mybband5

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mybband5", "bbands5", Array("1"), _
                        Array(param1.str, param2.str,
param3.str)
   mybband5 = ItSelf.Indicator ("mybband5")

   if not mybband5.ValidEx (1, 0) then
```

```
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   if (Data1.High (0) > mybband5.ValueEx (5, 0)) and _
      (Data1.Low  (0) < mybband5.ValueEx (2, 0)) then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = tq_Str2Color(clWhite)
   else

     if Data1.High (0) > mybband5.ValueEx (5, 0) then
        ItSelf.Plot (1) = Data1.High (0)
        ItSelf.Plot (2) = Data1.Low  (0)
        ItSelf.Plot (3) = tq_Str2Color(clGreen)
     elseif Data1.Low  (0) < mybband5.ValueEx (2, 0) then
        ItSelf.Plot (1) = Data1.High (0)
        ItSelf.Plot (2) = Data1.Low  (0)
        ItSelf.Plot (3) = tq_Str2Color(clRed)
     else
        ItSelf.SuccessEx (1) = false
        ItSelf.SuccessEx (2) = false
        ItSelf.SuccessEx (3) = false
     end if

   end if

end function
```

Formula Language

```
myCount := myCount +1;

GTupband    := High > bbands5.plot5 (data1, param1, param2,
param3);
LTlowband   := Low  < bbands5.plot2 (data1, param1, param2,
param3);
mydrawline  := GTupband <> 0 or LTlowband <> 0;

Success1 := if (myCount > param1 and mydrawline <> 0, 1, 0);
Success2 := if (myCount > param1 and mydrawline <> 0, 1, 0);
Success3 := if (myCount > param1 and mydrawline <> 0, 1, 0);

plot1 := high;
plot2 := low;
plot3 := if ( GTupband <> 0 and LTlowband <> 0, clWhite,
              if ( GTupband <> 0, clGreen, if ( LTlowband <> 0,
clRed, 0)));
```

Formula Column

```
if(h > bbands5.plot5 (m5, 10, 2, 1) and
   l < bbands5.plot2 (m5, 10, 1, 2), 3,
     if ( h > bbands5.plot5 (m5, 10, 2, 1), 2,
                    if ( l < bbands5.plot2 (m5, 10, 1, 2), 1, 0)))
```

Chaikin Oscillator (chaikinosc)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'chaikinosc', ['N'], ['n',
'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "chaikinosc", Array ("N"), Array
("n", "n")
```

Formula Language

```
chaikinosc (dataN, n, n);
```

Formula Column

```
chaikinosc (Tn, n, n)
```

Definition

Returns the Chaikin Oscillator. Chaikin Oscillator requires one series (Link 1).

Examples

Delphi Script

```
function MarkReverse : double;
var mychaikin : variant;
begin

     if not data1.valid [0] then
     begin
       ItSelf.SuccessEx [1] := false;
       ItSelf.SuccessEx [2] := false;
       ItSelf.SuccessEx [3] := false;
       exit;
     end;

     ItSelf.MakeIndicator ('mychaikin', 'chaikinosc', ['1'],
                          [param1.str, param2.str]);
     mychaikin := ItSelf.Indicator ('mychaikin');

     if not mychaikin.Valid [0] then
     begin
       ItSelf.SuccessEx [1] := false;
```

```
        ItSelf.SuccessEx [2] := false;
        ItSelf.SuccessEx [3] := false;
        exit;
      end;

      if (mychaikin.Value [1] < 0) and (mychaikin.Value [0] > 0)
then
      begin
        ItSelf.Plot [1] := Data1.High [0];
        ItSelf.Plot [2] := Data1.Low  [0];
        ItSelf.Plot [3] := tq_Str2Color(clGreen);
      end
      else if (mychaikin.Value [1] > 0) and (mychaikin.Value [0]
< 0) then
      begin
        ItSelf.Plot [1] := Data1.High [0];
        ItSelf.Plot [2] := Data1.Low  [0];
        ItSelf.Plot [3] := tq_Str2Color(clRed);
      end
      else
      begin
        ItSelf.SuccessEx [1] := false;
        ItSelf.SuccessEx [2] := false;
        ItSelf.SuccessEx [3] := false;
      end;
end;
```

VBScript

```
function MarkReverseVB()
   dim mychaikin

    if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
    end if

    ItSelf.MakeIndicator "mychaikin", "chaikinosc", Array("1"),
_
                        Array(param1.str, param2.str)
    mychaikin = ItSelf.Indicator ("mychaikin")

    if not mychaikin.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
    end if

    if (mychaikin.Value(1) < 0) and (mychaikin.Value(0) > 0)
then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = tq_Str2Color(clGreen)
    elseif (mychaikin.Value(1) > 0) and (mychaikin.Value(0) <
0) then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = tq_Str2Color(clRed)
    else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
    end if
end function
```

Formula Language

```
myCount := myCount +1;

ReverseUp   := chaikinosc (1, data1, param1, param2) < 0 and
                 chaikinosc (0, data1, param1, param2) > 0;
ReverseDown := chaikinosc (1, data1, param1, param2) > 0 and
                 chaikinosc (0, data1, param1, param2) < 0;
mydrawline  := ReverseUp <> 0 or ReverseDown <> 0;

Success1 := if (myCount > param1 and mydrawline <> 0, 1, 0);
Success2 := if (myCount > param1 and mydrawline <> 0, 1, 0);
Success3 := if (myCount > param1 and mydrawline <> 0, 1, 0);

plot1 := high;
plot2 := low;
plot3 := if( ReverseUp<>0, clGreen, if( ReverseDown<>0, clRed,
0));
```

Formula Column

```
chaikinosc (1, M5, 10, 20) > 0 and chaikinosc (0, M5, 10, 20) <
0
```

Chaikin Oscillator Custom (chaikinosc_custom)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'chaikinosc_custom', ['N',
'N'], ['n', 'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "chaikinosc_custom", Array("N",
"N"), Array("n", "n")
```

Formula Language

```
chaikinosc_custom (n, dataN, dataN.V, n, n);
```

Formula Column

Not supported.

Definition

Returns the Chaikin Oscillator. Chaikin Oscillator Custom requires two series. Link 1 is used for defining the price series while Link2 is used for defining the Volume series.

Examples

Delphi Script

```
function test : double;
var mychaikin, mymov : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mychaikin', 'chaikinosc', ['1', '2'],
                          [param1.str, param2.str]);
   mychaikin := ItSelf.Indicator ('mychaikin');

   ItSelf.MakeIndicator ('mymov', 'average', ['1'],
[param3.str]);
   mymov := ItSelf.Indicator ('mymov');

   if not (mychaikin.Valid [0] and mymov.Valid [0]) then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   if (mychaikin.Value [1] < 0) and (mychaikin.Value [0] > 0)
and
      (Data1.Value [0] > mymov.Value [0]) then
   begin
     ItSelf.Plot [1] := Data1.High [0];
     ItSelf.Plot [2] := Data1.Low  [0];
     ItSelf.Plot [3] := tq_Str2Color(clGreen);
   end
   else if (mychaikin.Value [1] > 0) and (mychaikin.Value [0] <
0) and
           (Data1.Value [0] > mymov.Value [0]) then
   begin
     ItSelf.Plot [1] := Data1.High [0];
     ItSelf.Plot [2] := Data1.Low  [0];
     ItSelf.Plot [3] := tq_Str2Color(clRed);
   end
   else
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
   end;

end;
```

VBScript

```
function testvb()
dim mychaikin, mymov

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mychaikin", "chaikinosc", Array("1",
"2"), _
                        Array(param1.str, param2.str)
   mychaikin = ItSelf.Indicator ("mychaikin")

   ItSelf.MakeIndicator "mymov", "average", Array("1"),
Array(param3.str)
   mymov = ItSelf.Indicator ("mymov")
   if not mychaikin.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   if (mychaikin.Value(1) < 0) and (mychaikin.Value(0) > 0) and
_
      (Data1.Value (0) > mymov.Value (0))then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = tq_Str2Color(clGreen)
   elseif (mychaikin.Value(1) > 0) and (mychaikin.Value(0) < 0)
and _
          (Data1.Value (0) < mymov.Value (0)) then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = tq_Str2Color(clRed)
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
myCount := myCount +1;

ReverseUp   := chaikinosc_custom (1, data1, data2, param1,
param2) < 0 and
               chaikinosc_custom (0, data1, data2, param1,
param2) > 0;
ReverseDown := chaikinosc_custom (1, data1, data2, param1,
param2) > 0 and
               chaikinosc_custom (0, data1, data2, param1,
param2) < 0;
AboveMov    := Data1 > Average(0, data1, param3);
BelowMov    := Data1 < Average(0, data1, param3);

Success1 := if (myCount > param1, if (ReverseUp<>0 and
AboveMov<>0, 1,
            if (ReverseDown<>0 and BelowMov<>0, 1, 0)),  0);
Success2 := if (myCount > param1, if (ReverseUp<>0 and
AboveMov<>0, 1,
            if (ReverseDown<>0 and BelowMov<>0, 1, 0)), 0);
Success3 := if (myCount > param1,  if (ReverseUp<>0 and
AboveMov<>0, 1,
            if (ReverseDown<>0 and BelowMov<>0, 1, 0)), 0);

plot1 := high;
plot2 := low;
plot3 := if( ReverseUp<>0, clGreen, if( ReverseDown<>0, clRed,
                    0));
```

Color Plot Formula (colorplotfml)

See Also

- ***Color Plot Formula*** (on page 1287)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'colorplotfml', ['N'],
['fml', 'n', 'n', 'color', 'color', 'fml', 'color', 'fml',
'color', 'fml', 'color', 'fml', 'color', 'fml'])
```

VBScript

```
Itself.MakeIndicator "string", "colorplotfml", Array("N"),
Array("fml", "n", "n", "color", "color", "fml", "color", "fml",
"color", "fml", "color", "fml", "color", "fml")
```

Formula Language

```
colorplotfml (dataN, "fml", "n", "n", "color", "color", "fml",
"color", "fml",
"color", "fml", "color", "fml", "color", "fml")
```

Formula Column

Not supported.

Definition

Color Plot Formula will plot colors based on the formula and coloring rules provided. Color Plot Formula requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mycolorplot : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
```

```
      ItSelf.SuccessEx [3] := false;
      exit;
    end;

    ItSelf.MakeIndicator ('mycolorplot', 'colorplotfml', ['1'],
                          ['o(1) > c(1)', 'h', 'l', 'None',
                           'clRed',  'formula>0 and c>o and
o(1)>h and c(1)<l',
                           'clBlue', 'formula=0 and c<o and
c(1)>h and o(1)<l',
                           '', '', '', '', '', '']);
    mycolorplot := ItSelf.Indicator ('mycolorplot');

    if not mycolorplot.Valid [0] then
    begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
    end
    else
    begin
      ItSelf.Plot [1] := mycolorplot.ValueEx [1, 0];
      ItSelf.Plot [2] := mycolorplot.ValueEx [2, 0];
      ItSelf.Plot [3] := mycolorplot.ValueEx [3, 0];
    end;

end;
```

VBScript

```
function testvb()
dim mycolorplot

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mycolorplot", "colorplotfml",
Array("1"), _
                 Array("o(1) > c(1)", "h", "l", "None", _
                       "clRed",  "formula>0 and c>o and o(1)>h
and c(1)<l", _
                       "clBlue", "formula=0 and c>o and c(1)>h
and o(1)<l", _
                       "", "", "", "", "", "")
   mycolorplot = ItSelf.Indicator ("mycolorplot")


   if not mycolorplot.ValidEx (1, 0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   else
```

```
      ItSelf.Plot (1) = mycolorplot.ValueEx (1, 0)
      ItSelf.Plot (2) = mycolorplot.ValueEx (2, 0)
      ItSelf.Plot (3) = mycolorplot.ValueEx (3, 0)
   end if

end function
```

Formula Language

```
HighVal  := colorplotfml.plot1 (0, data1, "o(1) > c(1)", "h",
"l", "None",
                           "clRed",  "formula>0 and c>o and
o(1)>h and c(1)<l",
                           "clBlue", "formula=0 and c<o and
c(1)>h and o(1)<l",
                           "", "", "", "", "", "");
LowVal   := colorplotfml.plot2 (0, data1, "o(1) > c(1)", "h",
"l", "None",
                           "clRed",  "formula>0 and c>o and
o(1)>h and c(1)<l",
                           "clBlue", "formula=0 and c<o and
c(1)>h and o(1)<l",
                           "", "", "", "", "", "");
ColorVal := colorplotfml.plot3 (0, data1, "o(1) > c(1)", "h",
"l", "None",
                           "clRed",  "formula>0 and c>o and
o(1)>h and c(1)<l",
                           "clBlue", "formula=0 and c<o and
c(1)>h and o(1)<l",
                           "", "", "", "", "", "");

Success1 := if (HighVal(1) <> HighVal and HighVal > 0, 1, 0);
Success2 := if (LowVal(1) <> LowVal and LowVal > 0, 1, 0);
Success3 := if (ColorVal(1) <> ColorVal and ColorVal > 0, 1,
0);

plot1 := HighVal;
plot2 := LowVal;
plot3 := ColorVal;
```

Commodity Channel Index (cci)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'cci', 'N', ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "cci", Array("N"), Array("n")
```

Formula Language

```
cci(dataN, n);
```

Formula Column

```
cci(Tm, n)
```

Definition

CCI is a price momentum indicator based on a scoring system that helps identify possible cyclical behavior. CCI takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mycci : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mycci', 'cci', ['1'], [params.items
[1].str]);
   mycci := ItSelf.Indicator ('mycci');

   if not mycci.Valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
```

```
        ItSelf.SuccessEx [3] := false;
      end
      else
      begin

        if mycci.value [0] > params.items [3].int then
        begin
          ItSelf.Plot [1] := mycci.Value [0];
          ItSelf.Plot [2] := params.items [3].int;
          ItSelf.Plot [3] := tq_str2color (params.items [2].str);
        end;

        if mycci.value [0] < params.items [5].int then
        begin
          ItSelf.Plot [1] := params.items [5].int;
          ItSelf.Plot [2] := mycci.Value [0];
          ItSelf.Plot [3] := tq_str2color (params.items [4].str);
        end;

      end;

   end;
```

VBScript

```
function testvb()
dim mycci

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mycci", "cci", Array("1"),
Array(params.items (1).str)
   mycci = ItSelf.Indicator ("mycci")


   if not mycci.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   else

      if mycci.value (0) > params.items (3).real then
         ItSelf.Plot (1) = mycci.Value (0)
         ItSelf.Plot (2) = params.items (3).real
         ItSelf.Plot (3) = tq_str2color (params.items (2).str)
      end if

      if mycci.value (0) < params.items (5).real then
         ItSelf.Plot (1) = params.items (5).real
         ItSelf.Plot (2) = mycci.Value (0)
         ItSelf.Plot (3) = tq_str2color (params.items (4).str)
```

```
        end if

    end if

end function
```

Formula Language

```
mycci := cci (data1, param1);

Success1 := if (mycci > param2 or mycci < param3, 1, 0);
Success2 := if (mycci > param2 or mycci < param3, 1, 0);
Success3 := if (mycci > param2 or mycci < param3, 1, 0);

plot1 := if(mycci > param2, mycci,   if(mycci < param3, param3,
0));
plot2 := if(mycci > param2, param2,  if(mycci < param3, mycci,
0));
plot3 := if(mycci > param2, clGreen, if(mycci < param3, clRed,
0));
```

Formula Column

```
cci (1, 5m, 34)
```

Constant (const)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'const', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "const", Array("N"), Array("n")
```

Formula Language

```
Const(dataN, n);
```

Formula Column

```
Const(Tn, n)
```

Definition

Constant requires one specified series (Link 1), which defines the time period when Constant returns a valid value.

Examples

Delphi Script

```
function test : double;
var myconst1, myconst2, myconst3, myconst4 : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myconst1', 'const', ['1'],
[params.items [1].str]);
   myconst1 := ItSelf.Indicator ('myconst1');
   ItSelf.MakeIndicator ('myconst2', 'const', ['1'],
[params.items [2].str]);
   myconst2 := ItSelf.Indicator ('myconst2');
   ItSelf.MakeIndicator ('myconst3', 'const', ['1'],
[params.items [3].str]);
   myconst3 := ItSelf.Indicator ('myconst3');
   ItSelf.MakeIndicator ('myconst4', 'const', ['1'],
[params.items [4].str]);
   myconst4 := ItSelf.Indicator ('myconst4');

   if not (myconst1.Valid [0] and myconst2.Valid [0] and
           myconst3.valid [0] and myconst3.Valid [0]) then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
   end
   else
   begin

     ItSelf.Plot [1] := myconst1.Value [0];
     ItSelf.Plot [2] := myconst2.Value [0];
     ItSelf.Plot [3] := myconst3.Value [0];
     ItSelf.Plot [4] := myconst4.Value [0];

   end;

end;
```

VBScript

```
function testvb()
dim myconst1, myconst2, myconst3, myconst4

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      exit function
   end if

   ItSelf.MakeIndicator "myconst1", "const", Array("1"),
Array(params.items (1).str)
   myconst1 = ItSelf.Indicator ("myconst1")
   ItSelf.MakeIndicator "myconst2", "const", Array("1"),
Array(params.items (2).str)
   myconst2 = ItSelf.Indicator ("myconst2")
   ItSelf.MakeIndicator "myconst3", "const", Array("1"),
Array(params.items (3).str)
   myconst3 = ItSelf.Indicator ("myconst3")
   ItSelf.MakeIndicator "myconst4", "const", Array("1"),
Array(params.items (4).str)
   myconst4 = ItSelf.Indicator ("myconst4")

   if not (myconst1.Valid (0) and myconst2.Valid (0) and _
           myconst3.Valid (0) and myconst4.Valid (0)) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
```

```
         ItSelf.SuccessEx (3) = false
         ItSelf.SuccessEx (4) = false
      else

         ItSelf.Plot (1) = myconst1.Value (0)
         ItSelf.Plot (2) = myconst2.Value (0)
         ItSelf.Plot (3) = myconst3.Value (0)
         ItSelf.Plot (4) = myconst4.Value (0)

      end if
   end function
```

Formula Language

```
plot1 := const(data1, param1);
plot2 := const(data1, param2);
plot3 := const(data1, param3);
plot4 := const(data1, param4);
```

Formula Column

```
const(M1, 15)
```

Constant 2 Lines (const2)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'const2', ['N'], ['Constant',
'Constant']);
```

VBScript

```
ItSelf.MakeIndicator "string", "const2", Array("N"),
Array("Constant", "Constant")
```

Formula Language

```
const2.PlotX (Data1, "Constant", "Constant");
```

Formula Column

```
Const2.PlotX (Tn, Constant, Constant)
```

Definition

Constant 2 Lines draw two lines in chart with two constant values. Constant 2 Lines requires one specified series (Link 1).

Examples

Delphi Script

```
function dp_const2 : double;
var myrsi, myconst2 : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   myrsi    := itself.makeindicator ('myrsi', 'RSIndex',
                                                ['1'],
[param1.str]);
   myconst2 := itself.makeindicator ('myconst2', 'const2',
                                                ['1'], ['70',
'30']);

   if not myrsi.valid [0] then
      itself.successex [1] := false
   else
      itself.plot [1] := myrsi.value [0];

   itself.plot [2] := myconst2.valueex [1, 0];
   itself.plot [3] := myconst2.valueex [2, 0];
end;
```

VBScript

```
   function vb_const2()
   dim myrsi, myconst2

      if not data1.valid (0) then
         itself.successall = false
         exit function
      end if

      myrsi    = Itself.MakeIndicator ("myrsi", "RSIndex", _
                                      Array("1"),
   Array(param1.str))
      myconst2 = Itself.MakeIndicator ("myconst2", "Const2", _
                                      Array("1"), Array("70",
   "30"))

      if not myrsi.valid (0) then
         itself.successex (1) = false
      else
         itself.plot (1) = myrsi.value (0)
      end if

      itself.plot (2) = myconst2.valueex (1, 0)
      itself.plot (3) = myconst2.valueex (2, 0)
   end function
```

Formula Language

```
   plot1 := RSIndex(data1, param1);
   plot2 := Const2.plot1 (data1, 70, 30);
   plot3 := Const2.plot2 (data1, 70, 30);
```

Formula Column

```
   Const2.Plot1 (data1, 70, 30)
```

Constant 3 Lines (const3)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'const3', ['N'], ['n', 'n',
'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "const3", Array("N"), Array("n",
"n", "n")
```

Formula Language

```
const3(dataN, n, n, n);
```

Formula Column

```
const3(Tn, n, n, n)
```

Definition

Constant 3 Lines requires one specified series (Link 1).

Example

Delphi Script

```
function test : double;
var myconst, mystoch : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mystoch', params.items [1].str,
['1'],
                            [ params.items [2].str, params.items
[3].str ]);
   mystoch := ItSelf.Indicator ('mystoch');
   ItSelf.MakeIndicator ('myconst', 'const3', ['1'],
         [params.items [4].str, params.items [5].str,
params.items [6].str]);
   myconst := ItSelf.Indicator ('myconst');

   if not (myconst.ValidEx [1, 0] and mystoch.Valid [0]) then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
   end
   else
   begin
     ItSelf.Plot [1] := mystoch.Value [0];
     ItSelf.Plot [2] := myconst.ValueEx [1, 0];
     ItSelf.Plot [3] := myconst.ValueEx [2, 0];
     ItSelf.Plot [4] := myconst.ValueEx [3, 0];
   end;

end;
```

VBScript

```
function testvb()
dim myconst, mystoch

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      exit function
   end if

   ItSelf.MakeIndicator "mystoch", params.items (1).str,
Array("1"), _
                        Array(params.items (2).str,
params.items (3).str)
   mystoch = ItSelf.Indicator ("mystoch")

   ItSelf.MakeIndicator "myconst", "const3", Array("1"), _
          Array(params.items (4).str, params.items (5).str,
params.items (6).str)
   myconst = ItSelf.Indicator ("myconst")

   if not (myconst.ValidEx (1, 0) and mystoch.Valid (0)) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
   else
      ItSelf.Plot (1) = mystoch.Value (0)
      ItSelf.Plot (2) = myconst.ValueEx (1, 0)
      ItSelf.Plot (3) = myconst.ValueEx (2, 0)
      ItSelf.Plot (4) = myconst.ValueEx (3, 0)
   end if


end function
```

Formula Language

```
plot1 := slowd(data1, param1, param2);
plot2 := const3.plot1 (data1, param3, param4, param5);
plot3 := const3.plot2 (data1, param3, param4, param5);
plot4 := const3.plot3 (data1, param3, param4, param5);
```

Formula Column

```
const3(M1, 0, 50, 100)
```

Constant 4 Lines (Const4)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'Const4', ['N'], ['Constant',
'Constant', 'Constant', 'Constant']);
```

VBScript

```
ItSelf.MakeIndicator "string", "Const4", Array("N'),
Array("Constant", "Constant", "Constant", "Constant"
```

Formula Language

```
Const4.PlotX (Data1, "Constant", "Constant", "Constant",
"Constant");
```

Formula Column

```
Const4.PlotX (Tn, Constant, Constant, Constant, Constant)
```

Definition

Constant 4 Lines requires one specified series (Link 1).

Examples

Delphi Script

```
function dp_const4 : double;
var myrsi, myconst4 : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   myrsi    := ItSelf.MakeIndicator ('myrsi', 'RSIndex',
                                            ['1'],
[param1.str]);
   myconst4 := ItSelf.MakeIndicator ('myconst4', 'const4',
                                            ['1'], ['80', '20',
'70', '30']);

   if not myrsi.valid [0] then
      itself.successex [1] := false
   else
      itself.plot [1] := myrsi.value [0];

   itself.plot [2] := myconst4.valueex [1, 0];
   itself.plot [3] := myconst4.valueex [2, 0];
   itself.plot [4] := myconst4.valueex [3, 0];
   itself.plot [5] := myconst4.valueex [4, 0];
end;
```

VBScript

```
function vb_const4()
dim myrsi, myconst4

   if not data1.valid (0) then
      itself.successall = false
      exit function
   end if

   myrsi    = itself.MakeIndicator ("myrsi", "RSIndex", _
                                    Array("1"),
Array(param1.str))
   myconst4 = itself.MakeIndicator ("myconst4", "const4", _
                                    Array("1"), Array("30",
"70", "20", "80"))

   if not myrsi.valid (0) then
      itself.successex (1) = false
   else
      itself.plot (1) = myrsi.value (0)
   end if

   itself.plot (2) = myconst4.valueex (1, 0)
   itself.plot (3) = myconst4.valueex (2, 0)
   itself.plot (4) = myconst4.valueex (3, 0)
   itself.plot (5) = myconst4.valueex (4, 0)
end function
```

Formula Language

```
plot1 := RSIndex (data1, param1);
plot2 := Const4.plot1 (data1, 30, 70, 20, 80);
plot3 := Const4.plot2 (data1, 30, 70, 20, 80);
plot4 := Const4.plot3 (data1, 30, 70, 20, 80);
plot5 := Const4.plot4 (data1, 30, 70, 20, 80);
```

Formula Column

```
Const4.plot1(M1, 30, 70, 20, 80)
```

Constant 5 Lines (const5)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'const5', ['N'], ['n', 'n',
'n', 'n', 'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "const5", Array("N"), Array("n",
"n", "n", "n", "n")
```

Formula Language

```
const5(dataN, n, n, n, n, n);
```

Formula Column

```
const5(Tn, n, n, n, n, n)
```

Definition

Constant 5 Lines requires one specified series (Link 1).

Examples

Delphi Script

```
function test : double;
var myconst, mystoch : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     ItSelf.SuccessEx [5] := false;
     ItSelf.SuccessEx [6] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mystoch', params.items [1].str,
['1'],
                         [ params.items [2].str, params.items
[3].str ]);
```

```
   mystoch := ItSelf.Indicator ('mystoch');


   ItSelf.MakeIndicator ('myconst', 'const5', ['1'],
         [params.items [4].str, params.items [5].str,
params.items [6].str,
          params.items [7].str, params.items [8].str]);
   myconst := ItSelf.Indicator ('myconst');

   if not (myconst.ValidEx [1, 0] and mystoch.Valid [0]) then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     ItSelf.SuccessEx [5] := false;
     ItSelf.SuccessEx [6] := false;
   end
   else
   begin
     ItSelf.Plot [1] := mystoch.Value [0];
     ItSelf.Plot [2] := myconst.ValueEx [1, 0];
     ItSelf.Plot [3] := myconst.ValueEx [2, 0];
     ItSelf.Plot [4] := myconst.ValueEx [3, 0];
     ItSelf.Plot [5] := myconst.ValueEx [4, 0];
     ItSelf.Plot [6] := myconst.ValueEx [5, 0];
   end;

end;
```

VBScript

```
function testvb()
dim myconst, mystoch

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      ItSelf.SuccessEx (5) = false
      ItSelf.SuccessEx (6) = false
      exit function
   end if

   ItSelf.MakeIndicator "mystoch", params.items (1).str,
Array("1"), _
                        Array(params.items (2).str,
params.items (3).str)
   mystoch = ItSelf.Indicator ("mystoch")

   ItSelf.MakeIndicator "myconst", "const5", Array("1"), _
          Array(params.items (4).str, params.items (5).str,
params.items (6).str, _
                params.items (7).str, params.items (8).str)
   myconst = ItSelf.Indicator ("myconst")
```

```
    if not (myconst.ValidEx (1, 0) and mystoch.Valid (0)) then
       ItSelf.SuccessEx (1) = false
       ItSelf.SuccessEx (2) = false
       ItSelf.SuccessEx (3) = false
       ItSelf.SuccessEx (4) = false
       ItSelf.SuccessEx (5) = false
       ItSelf.SuccessEx (6) = false
    else
       ItSelf.Plot (1) = mystoch.Value (0)
       ItSelf.Plot (2) = myconst.ValueEx (1, 0)
       ItSelf.Plot (3) = myconst.ValueEx (2, 0)
       ItSelf.Plot (4) = myconst.ValueEx (3, 0)
       ItSelf.Plot (5) = myconst.ValueEx (4, 0)
       ItSelf.Plot (6) = myconst.ValueEx (5, 0)
    end if

end function
```

Formula Language

```
plot1 := slowd(data1, param1, param2);
plot2 := const5.plot1 (data1, param3, param4, param5, param6,
param7);
plot3 := const5.plot2 (data1, param3, param4, param5, param6,
param7);
plot4 := const5.plot3 (data1, param3, param4, param5, param6,
param7);
plot5 := const5.plot4 (data1, param3, param4, param5, param6,
param7);
plot6 := const5.plot5 (data1, param3, param4, param5, param6,
param7);
```

Formula Column

```
const5(M1, 0, 25, 50, 75, 100)
```

Correlation Coefficient (correl)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'correl', ['N', 'N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "correl", Array("N", "N"),
Array("n")
```

Formula Language

```
correl(DataN, DataN, n);
```

Formula Column

Not supported.

Definition

Returns the statistics correlation coefficient over the specified Period. Correlation Coefficient requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mycorrel : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mycorrel', 'correl', ['1', '2'],
                        [ params.items [1].str ]);
   mycorrel := ItSelf.Indicator ('mycorrel');

   if not mycorrel.Valid [0] then
   begin
```

```
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
   end
   else if (mycorrel.Value [0] > 0.9) or (mycorrel.value [0] <
-0.9) then
   begin
      ItSelf.Plot [1] := data1.high [0];
      ItSelf.Plot [2] := data1.low  [0];
      ItSelf.Plot [3] := clRed;
   end
   else
   begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
   end;

end;
```

VBScript

```
function testvb()
dim mycorrel

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mycorrel", "correl", Array("1", "2"),
_
                        Array(params.items (1).str)
   mycorrel = ItSelf.Indicator ("mycorrel")

   if not mycorrel.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   elseif (mycorrel.Value (0) > 0.9) or (mycorrel.Value (0) < -
0.9) then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clGreen
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
mycorrel := correl (data1, data2, param1);

success1 := if( mycorrel > 0.9 or mycorrel < -0.9, 1, 0);
success2 := if( mycorrel > 0.9 or mycorrel < -0.9, 1, 0);
success3 := if( mycorrel > 0.9 or mycorrel < -0.9, 1, 0);

plot1 := High;
plot2 := Low;
plot3 := clRed;
```

Countdown Tick (countdown_tick)

Apply this indicator to a chart, the indicator will display the number of ticks remaining to complete the latest bar. This indicator works only in real-time and is intended to be used in time charts only.

Countdown Time (countdown_time)

Apply this indicator to a chart, the indicator will display the time remaining to complete the latest bar. This indicator works only in real-time and is intended to be used in time charts only.

Cross Above (xabove)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'xabove', ['N', 'N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "xabove", Array("N", "N"),
Array("")
```

Formula Language

```
xabove(N, dataN, dataN);
```

Formula Column

Not supported.

Definiton

Returns 1 when Link 1 crosses above Link 2, 0 otherwise.

Examples

Delphi Script

```
function test : double;
var myxabove : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mymov1', 'average', ['1'],
[param1.str]);
   ItSelf.MakeIndicator ('mymov2', 'average', ['1'],
[param2.str]);
   ItSelf.MakeIndicator ('myxabove', 'xabove', ['mymov1',
'mymov2'], ['']);
   myxabove := ItSelf.Indicator ('myxabove');
   if not myxabove.Valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
   end
   else if myxabove.Value [0] <> 0 then
   begin
     ItSelf.Plot [1] := data1.high [0];
     ItSelf.Plot [2] := data1.low  [0];
     ItSelf.Plot [3] := clRed;
   end
   else
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
   end;

end;
```

VBScript

```
function testvb()
dim myxabove

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mymov1", "average", Array("1"),
Array(param1.str)
   ItSelf.MakeIndicator "mymov2", "average", Array("1"),
Array(param2.str)
   ItSelf.MakeIndicator "myxabove", "xabove", _
                        Array("mymov1", "mymov2"), Array("")
   myxabove = ItSelf.Indicator ("myxabove")

   if not myxabove.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   elseif myxabove.Value (0) <> 0 then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clGreen
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if
end function
```

Formula Language

```
mymov1    := average(data1, param1);
mymov2    := average(data1, param2);
myxabove := xabove (mymov1, mymov2);


success1 := if( myxabove <> 0, 1, 0);
success2 := if( myxabove <> 0, 1, 0);
success3 := if( myxabove <> 0, 1, 0);

plot1 := High;
plot2 := Low;
plot3 := clRed;
```

Cross Below (xbelow)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'xbelow', ['N', 'N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "xbelow", Array("N", "N"),
Array("")
```

Formula Language

```
xbelow(DataN, DataN);
```

Formula Column

Not supported.

Definition

Returns 1 when Link 1 crosses below Link 2, 0 otherwise.

Examples

Delphi Script

```
function test : double;
var myxbelow : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mymov1', 'average', ['1'],
[param1.str]);
   ItSelf.MakeIndicator ('mymov2', 'average', ['1'],
[param2.str]);
   ItSelf.MakeIndicator ('myxbelow', 'xbelow', ['mymov1',
'mymov2'], ['']);
   myxbelow := ItSelf.Indicator ('myxbelow');
   if not myxbelow.Valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
   end
   else if myxbelow.Value [0] <> 0 then
   begin
     ItSelf.Plot [1] := data1.high [0];
     ItSelf.Plot [2] := data1.low  [0];
     ItSelf.Plot [3] := clRed;
   end
   else
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
   end;

end;
```

VBScript

```
function testvb()
dim myxbelow

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mymov1", "average", Array("1"),
Array(param1.str)
   ItSelf.MakeIndicator "mymov2", "average", Array("1"),
Array(param2.str)
   ItSelf.MakeIndicator "myxbelow", "xbelow", _
                        Array("mymov1", "mymov2"), Array("")
   myxbelow = ItSelf.Indicator ("myxbelow")

   if not myxbelow.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   elseif myxbelow.Value (0) <> 0 then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clGreen
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if
end function
```

Formula Language

```
mymov1   := average(data1, param1);
mymov2   := average(data1, param2);
myxbelow := xbelow (mymov1, mymov2);


success1 := if( myxbelow <> 0, 1, 0);
success2 := if( myxbelow <> 0, 1, 0);
success3 := if( myxbelow <> 0, 1, 0);

plot1 := High;
plot2 := Low;
plot3 := clRed;
```

Current Day High (currDHigh)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'currDHigh', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "currDHigh", Array("N"),
Array("")
```

Formula Language

```
currDHigh(DataN);
```

Formula Column

```
currDHigh(Tn)
```

Definition

Given a daily or intraday chart, this indicator draws a horizontal line marking the current day high price level and automatically adjust itself when the day has changed. This is a support/resistance indicators. Current Day High requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mycurrDHigh : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mycurrDHigh', 'currDHigh', ['1'],
['']);
   mycurrDHigh := ItSelf.Indicator ('mycurrDHigh');

   if not mycurrDHigh.Valid [0] then
   begin
```

```
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
    end
    else if  Data1.High [0] >= mycurrDHigh.Value [0] then
    begin
      tq_playsound ('c:\windows\media\chimes.wav');
      ItSelf.Plot [1] := data1.high [0];
      ItSelf.Plot [2] := data1.low  [0];
      ItSelf.Plot [3] := clGreen;
    end
    else
    begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
    end;

end;
```

VBScript

```
function testvb()
dim mycurrDhigh

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mycurrdhigh", "currDHigh", Array("1"),
Array("")
   mycurrDhigh = ItSelf.Indicator ("mycurrdhigh")

   if not mycurrDhigh.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   elseif Data1.High (0) >= mycurrDhigh.Value (0)  then
      tq_playsound ("c:\windows\media\chimes.wav")
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clGreen
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
myCurrDHigh := currDHigh(data1);
```

```
success1 := if( h >= myCurrDHigh, 1, 0);
success2 := if( h >= myCurrDHigh, 1, 0);
success3 := if( h >= myCurrDHigh, 1, 0);

plot1 := High;
plot2 := Low;
plot3 := clRed;
```

Formula Column

```
currDHigh (m5)
```

Current Day Low (currDLow)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'currDLow', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "currDLow", Array("N"),
Array("")
```

Formula Language

```
currDLow(DataN);
```

Formula Column

```
currDLow(Tn)
```

Definition

Given a daily or intraday chart, this indicator draws a horizontal line marking the current day low price level and automatically adjust itself when the day has changed. This is a support/resistance indicators. Current Day Low requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mycurrDLow : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mycurrDLow', 'currDLow', ['1'],
['']);
   mycurrDLow := ItSelf.Indicator ('mycurrDLow');

   if not mycurrDLow.Valid [0] then
   begin
```

```
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
    end
    else if Data1.Low [0] <= mycurrDLow.Value [0] then
    begin
      tq_playsound ('c:\windows\media\chimes.wav');
      ItSelf.Plot [1] := data1.high [0];
      ItSelf.Plot [2] := data1.low  [0];
      ItSelf.Plot [3] := clGreen;
    end
    else
    begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
    end;

end;
```

VBScript

```
function testvb()
dim mycurrDLow

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mycurrdlow", "currDLow", Array("1"),
Array("")
   mycurrDhigh = ItSelf.Indicator ("mycurrdlow")

   if not mycurrDlow.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   elseif Data1.Low (0) <= mycurrDlow.Value (0) then
      tq_playsound ("c:\windows\media\chimes.wav")
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clGreen
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
myCurrDLow := currDLow(data1);
```

```
success1 := if( l <= myCurrDLow, 1, 0);
success2 := if( l <= myCurrDLow, 1, 0);
success3 := if( l <= myCurrDLow, 1, 0);

plot1 := High;
plot2 := Low;
plot3 := clRed;
```

Formula Column

```
currDLow(m)
```

Current Day Open (currDOpen)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'currDOpen', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "currDOpen", Array("N"),
Array("")
```

Formula Language

```
currDOpen(dataN)
```

Formula Column

```
curDOpen(Tn)
```

Definition

Given a daily or intraday chart, this indicator draws a horizontal line marking the current day open price level and automatically adjust itself when the day has changed. This is a support/resistance indicators. Current Day Open requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mycurrDOpen : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mycurrDOpen', 'currDOpen', ['1'],
['']);
   mycurrDOpen := ItSelf.Indicator ('mycurrDOpen');

   if not mycurrDOpen.Valid [0] then
   begin
```

```
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
   end
   else
   begin

      if tq_abs(Data1.High [0] - mycurrDOpen.Value [0]) > 0.5
then
      begin
         ItSelf.Plot [1] := data1.high [0];
         ItSelf.Plot [2] := data1.low  [0];
         ItSelf.Plot [3] := clGreen;
      end
      else if tq_abs(Data1.Low [0] - mycurrDOpen.Value [0]) >
0.5 then
      begin
         ItSelf.Plot [1] := data1.high [0];
         ItSelf.Plot [2] := data1.low  [0];
         ItSelf.Plot [3] := clRed;
      end
      else
      begin
        ItSelf.SuccessEx [1] := false;
        ItSelf.SuccessEx [2] := false;
        ItSelf.SuccessEx [3] := false;
      end;

   end;

end;
```

VBScript

```
function testvb()
dim mycurrDopen

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mycurrdopen", "currDOpen", Array("1"),
Array("")
   mycurrDopen = ItSelf.Indicator ("mycurrdopen")

   if not mycurrDOpen.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   elseif tq_abs(Data1.High(0) - mycurrDOpen.Value(0)) > 0.5
then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clGreen
   elseif tq_abs(Data1.Low(0) - mycurrDOpen.Value(0)) > 0.5
then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clRed
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
myhighDiff := absvalue(high - currDOpen(data1));
mylowDiff  := absvalue(low - currDOpen(data1));

success1 := if( myhighDiff > 0.5 or mylowDiff > 0.5, 1, 0);
success2 := if( myhighDiff > 0.5 or mylowDiff > 0.5, 1, 0);
success3 := if( myhighDiff > 0.5 or mylowDiff > 0.5, 1, 0);

plot1 := High;
plot2 := Low;
plot3 := if(myhighDiff > 0.5, clGreen, if(mylowDiff > 0.5,
clRed, 0));
```

Formula Column

```
currDOpen(m10)
```

Current Month Open (currMOpen)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'currMOpen', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "currMOpen", Array("N"),
Array("")
```

Formula Language

```
currMOpen (DataN);
```

Formula Column

```
currMOpen(Tn)
```

Definition

Current Month Open draws a series of horizontal dots marking the opening price level of the current month. It works on series with time frame higher than monthly. It requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mycurrMOpen : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mycurrMOpen', 'currMOpen', ['1'],
['']);
   mycurrMOpen := ItSelf.Indicator ('mycurrMOpen');

   if not mycurrMOpen.Valid [0] then
   begin
```

```
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
   end
   else
   begin

      if tq_abs(Data1.High [0] - mycurrMOpen.Value [0]) > 0.5
then
      begin
         ItSelf.Plot [1] := data1.high [0];
         ItSelf.Plot [2] := data1.low  [0];
         ItSelf.Plot [3] := clGreen;
      end
      else if tq_abs(Data1.Low [0] - mycurrMOpen.Value [0]) >
0.5 then
      begin
         ItSelf.Plot [1] := data1.high [0];
         ItSelf.Plot [2] := data1.low  [0];
         ItSelf.Plot [3] := clRed;
      end
      else
      begin
        ItSelf.SuccessEx [1] := false;
        ItSelf.SuccessEx [2] := false;
        ItSelf.SuccessEx [3] := false;
      end;

   end;

end;
```

VBScript

```
function testvb()
dim mycurrMopen

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mycurrMopen", "currMOpen", Array("1"),
Array("")
   mycurrMOpen = ItSelf.Indicator ("mycurrMOpen")

   if not mycurrMOpen.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   elseif tq_abs(Data1.High(0) - mycurrMOpen.Value(0)) > 0.5
then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clGreen
   elseif tq_abs(Data1.Low(0) - mycurrMOpen.Value(0)) > 0.5
then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clRed
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
myhighDiff := absvalue(high - currMOpen(data1));
mylowDiff  := absvalue(low - currMOpen(data1));

success1 := if( myhighDiff > 0.5 or mylowDiff > 0.5, 1, 0);
success2 := if( myhighDiff > 0.5 or mylowDiff > 0.5, 1, 0);
success3 := if( myhighDiff > 0.5 or mylowDiff > 0.5, 1, 0);

plot1 := High;
plot2 := Low;
plot3 := if(myhighDiff > 0.5, clGreen, if(mylowDiff > 0.5,
clRed, 0));
```

Formula Column

```
currMOpen (m15)
```

Current Week Open (currWOpen)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'currWOpen', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "currWOpen", Array"N"),
Array("")
```

Formula Language

```
currWOpen(DataN);
```

Formula Column

```
currWOpen(Tn)
```

Definition

Current Week Open draws a series of horizontal dots marking the opening price level of the current week. It works on series with time frame higher than weekly. It requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mycurrWOpen : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mycurrWOpen', 'currWOpen', ['1'],
['']);
   mycurrWOpen := ItSelf.Indicator ('mycurrWOpen');

   if not mycurrWOpen.Valid [0] then
   begin
```

```
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
   end
   else
   begin

      if tq_abs(Data1.High [0] - mycurrWOpen.Value [0]) > 0.5
then
      begin
         ItSelf.Plot [1] := data1.high [0];
         ItSelf.Plot [2] := data1.low  [0];
         ItSelf.Plot [3] := clGreen;
      end
      else if tq_abs(Data1.Low [0] - mycurrWOpen.Value [0]) >
0.5 then
      begin
         ItSelf.Plot [1] := data1.high [0];
         ItSelf.Plot [2] := data1.low  [0];
         ItSelf.Plot [3] := clRed;
      end
      else
      begin
        ItSelf.SuccessEx [1] := false;
        ItSelf.SuccessEx [2] := false;
        ItSelf.SuccessEx [3] := false;
      end;

   end;

end;
```

VBScript

```
function testvb()
dim mycurrWopen

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mycurrWopen", "currWOpen", Array("1"),
Array("")
   mycurrWOpen = ItSelf.Indicator ("mycurrWopen")

   if not mycurrWOpen.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   elseif tq_abs(Data1.High(0) - mycurrWOpen.Value(0)) > 0.5
then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clGreen
   elseif tq_abs(Data1.Low(0) - mycurrWOpen.Value(0)) > 0.5
then
      ItSelf.Plot (1) = Data1.High (0)
      ItSelf.Plot (2) = Data1.Low  (0)
      ItSelf.Plot (3) = clRed
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
myhighDiff := absvalue(high - currWOpen(data1));
mylowDiff  := absvalue(low - currWOpen(data1));

success1 := if( myhighDiff > 0.5 or mylowDiff > 0.5, 1, 0);
success2 := if( myhighDiff > 0.5 or mylowDiff > 0.5, 1, 0);
success3 := if( myhighDiff > 0.5 or mylowDiff > 0.5, 1, 0);

plot1 := High;
plot2 := Low;
plot3 := if(myhighDiff > 0.5, clGreen, if(mylowDiff > 0.5,
clRed, 0));
```

Formula Column

```
currWOpen (M10)
```

Custom 1 Delphi (custom)

Custom 1 Delphi is a legacy indicator.

Custom 2 Delphi (custom2)

Custom 2 Delphi is a legacy indicator.

Data Breadth (databreadth)

See Also

- ***Data Breadth*** (on page 1301)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'databreadth', ['N'], ['n',
'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "databreadth", Array("N"),
Array("n", "n")
```

Formula Language

```
databreadth(DataN, n, n);
```

Formula Column

Not supported*

Definition

Data Breadth returns 4 basic real time market breadth data (Advance Issues, Decline Issues, Advance Volume, Decline Volume) based on the data series from the First Series to the Last Series.

Examples

Delphi Script

```
function test : double;
var mybreadth : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
   end;
   ItSelf.MakeIndicator ('mybreadth', 'databreadth', ['1'],
          [params.items [1].str, params.items [2].str]);
   mybreadth := ItSelf.Indicator ('mybreadth');

   if not mybreadth.Valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
   end
   else
   begin

     if mybreadth.ValueEx [1, 0] > mybreadth.ValueEx [2, 0]
then
     begin
       ItSelf.Plot [1] := params.items [3].real;
       ItSelf.Plot [2] := params.items [4].real;
       ItSelf.Plot [3] := tq_str2Color(params.items [5].str);
     end
     else if mybreadth.ValueEx [1, 0] < mybreadth.Valueex [2,
0] then
     begin
       ItSelf.Plot [1] := params.items [3].real;
       ItSelf.Plot [2] := params.items [4].real;
       ItSelf.Plot [3] := tq_str2Color(params.items [6].str);
     end
     else
     begin
       ItSelf.Plot [1] := params.items [3].real;
       ItSelf.Plot [2] := params.items [4].real;
       ItSelf.Plot [3] := tq_str2Color(params.items [7].str);
     end;

   end;

end;
```

VBScript

```
function testvb()
dim mybreadth

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mybreadth", "databreadth", Array("1"),
_
          Array(params(1).str, params(2).str )
   mybreadth = ItSelf.Indicator ("mybreadth")

   if not mybreadth.Valid (0) then
```

```
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   else
      if mybreadth.ValueEx (1, 0) > mybreadth.ValueEx (2, 0)
then
        ItSelf.Plot (1) = Params(3).real
        ItSelf.Plot (2) = Params(4).real
        ItSelf.Plot (3) = tq_str2Color(Params(5).str)
      elseif mybreadth.ValueEx (1, 0) < mybreadth.ValueEx (2,
0) then
        ItSelf.Plot (1) = Params(3).real
        ItSelf.Plot (2) = Params(4).real
        ItSelf.Plot (3) = tq_str2Color(Params(6).str)
      else
        ItSelf.Plot (1) = Params(3).real
        ItSelf.Plot (2) = Params(4).real
        ItSelf.Plot (3) = tq_str2Color(params(7).str)
      end if
   end if

end function
```

Formula Language

```
AdvanceIssue := DataBreadth.plot1 (Data1, param1, param2);
DeclineIssue := DataBreadth.plot2 (Data1, param1, param2);

plot1 := param3;
plot2 := param4;
plot3 := if (AdvanceIssue > DeclineIssue, clGreen,
             if (AdvanceIssue < DeclineIssue, clRed, clBlack));
```

* While you can call data breadth from within a quote window, the calculation is meaningless as there is only one data series present: the symbol itself.

Day Percent (DayPercent)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'DayPercent', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "DayPercent", Array("N"),
Array("")
```

Formula Language

```
DayPercent(DataN);
```

Formula Column

```
DayPercent(Tn)
```

Definition

Day Percent returns in percentage (0 to 100) of the current bar time relative to the trading time range of the hosting function window.

Examples

Delphi Script

```
function test : double
var mydaypct : variant;
begin

   if not data1.valid [0] then
     ItSelf.Success := false;

   ItSelf.MakeIndicator ('mydaypct', 'DayPercent', ['1'],
['']);
   mydaypct := ItSelf.Indicator ('mydaypct');

   if not mydaypct.Valid [0] then
     ItSelf.Success := false
   else
   begin

     if mydaypct.value [0] <> 0 then
        result := Data1.Volume [0] +
                  Data1.Volume [0]*((100-mydaypct.value
[0])/mydaypct.value [0])
     else
        ItSelf.Success := false;

   end;

end;
```

VBScript

```
function testvb()
dim mydaypct

   if not Data1.Valid (0) then
      ItSelf.Success = false
   end if

   ItSelf.MakeIndicator "mydaypct", "DayPercent", Array("1"),
Array("")
   mydaypct = ItSelf.Indicator ("mydaypct")

   if not mydaypct.Valid (0) then
      ItSelf.Success = false
   else
      if mydaypct.Value (0) > 0 then
         testvb = Data1.Volume(0)* _
                  (1+(100-mydaypct.value(0))/mydaypct.value(0))
      else
        ItSelf.Success = false
      end if
   end if

end function
```

Formula Language

```
myDayPct := DayPercent (Data1);
success1 := if (myDayPct > 0, 1, 0);
plot1 := Data1.Volume*(1+(100-MyDayPct)/MyDayPct);
```

Formula Column

```
DayPercent(m5)
```

Days Since (dayssince)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'dayssince', ['N'], ['Date']);
```

VBScript

```
ItSelf.MakeIndicator "string", "dayssince", Array("N"),
Array("Date")
```

Formula Language

```
DaysSince(Data1, Date);
```

Formula Column

```
DaySince(Tn, Date)
```

Definition

Days Since returns the number of days since the given date. This indicator must be applied on a series that has a time frame that is 1 day or lower. Days Since requires one specified series (Link 1).

Examples

Delphi Script

```
function test : double;
var mydayssince : variant;
    BarsAgo : integer;
    HighDate : string;
begin

   // This indicator returns the number of days since this
   // symbol makes its highest high

   if not data1.valid [0] then
   begin
     ItSelf.Success := false;
     exit;
   end;

   if Heap.Size = 0 then
   begin
     Heap.Allocate (2);
```

```
      Heap.Value [0] := Data1.High [0];
      Heap.Value [1] := Data1.Barsnum [0];
      ItSelf.Success := false;
      exit;
    end;

    if Data1.High [0] > Heap.Value [0] then
    begin
      Heap.Value [0] := Data1.High [0];
      Heap.Value [1] := data1.Barsnum [0];
    end;

    BarsAgo := Data1.Barsnum [0] - Heap.Value [1];
    HighDate := tq_date2str(Data1.Date [BarsAgo]);

    ItSelf.MakeIndicator ('mydayssince', 'DaysSince', ['1'],
 [HighDate]);
    mydayssince := ItSelf.Indicator ('mydayssince');

    if not mydayssince.valid [0] then
       ItSelf.Success := false
    else
       result := mydayssince.value [0];

 end;
```

VBScript

```
function testvb()
dim mydayssince
dim BarsAgo
dim HighDate

   if not Data1.Valid (0) then
      ItSelf.Success = false
      exit function
   end if

   if Heap.Size = 0 then
      Heap.Allocate (2)
      Heap.Value (0) = Data1.High (0)
      Heap.Value (1) = Data1.Barsnum (0)
      exit function
   end if

   if Data1.high (0) > Heap.Value (0) then
      Heap.Value (0) = Data1.High (0)
      Heap.Value (1) = Data1.Barsnum (0)
   end if

   BarsAgo  = Data1.Barsnum (0) - Heap.Value (1)
   HighDate = tq_date2str(Data1.Date (BarsAgo))

   ItSelf.MakeIndicator "mydayssince", "DaysSince", Array("1"),
_
                        Array(HighDate)
   mydayssince = ItSelf.Indicator ("mydayssince")

   if not mydayssince.Valid (0) then
      ItSelf.Success = false
   else
      testvb = mydayssince.Value (0)
   end if

end function
```

Formula Language

```
plot1 := DaysSince(data1, "7/1/2002");
```

Formula Column

```
DaysSince (M5, "7/1/2002")
```

Directional Movement Index (DMI)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'DMI', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "DMI", Array("N"), Array("n")
```

Formula Language

```
DMI(DataN, n);
```

Formula Column

```
DMI(Tn, n)
```

Definition

Directional Movement Index is a trend indicator. DMI will rise when the trend is developing and will stay at higher value if the trend is intact. DMI is calculated based on DMI Plus and DMI Minus. DMI takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var mydivergence : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.Success := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myDMI', 'DMI', ['1'], [param1.str]);
   ItSelf.MakeIndicator ('myADX', 'ADX', ['1'], [param2.str]);
   ItSelf.MakeIndicator ('mydivergence', 'diff', ['myDMI',
'myADX'], ['']);
   mydivergence := ItSelf.Indicator ('mydivergence');

   if not mydivergence.valid [0] then
      ItSelf.Success := false
   else
      result := mydivergence.value [0];

end;
```

VBScript

```
function testvb()
dim mydivergenece

   if not Data1.Valid (0) then
      ItSelf.Success = false
      exit function
   end if

   ItSelf.MakeIndicator "myDMI", "DMI", Array("1"),
Array(Param1.str)
   ItSelf.MakeIndicator "myADX", "ADX", Array("1"),
Array(param2.str)
   ItSelf.MakeIndicator "mydivergence", "diff", Array("myDMI",
"myADX"), Array("")
   mydivergence = ItSelf.Indicator ("mydivergence")

   if not mydivergence.Valid (0) then
      ItSelf.Success = false
   else
      testvb = mydivergence.Value (0)
   end if

end function
```

Formula Language

```
plot1 := DMI(Data1, param1) - ADX(data1, param2);
```

Formula Column

```
DMI (M5, 14) - ADX(M5, 14)
```

Directional Movement Index Minus (DMIminus)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'DMIminus', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "DMIminus", Array("N"),
Array("n")
```

Formula Language

```
DMIminus(DataN, n);
```

Formula Column

```
DMIminus(Tn, n)
```

Definition

It is the down component of the DMI. DMI Minus takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var myxabove, myxbelow : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEX [1] := false;
     ItSelf.SuccessEX [2] := false;
     ItSelf.SuccessEX [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myDMIplus',  'DMIplus',  ['1'],
[param1.str]);
   ItSelf.MakeIndicator ('myDMIminus', 'DMIminus', ['1'],
[param2.str]);

   ItSelf.MakeIndicator ('myxabove', 'xabove',
                          ['myDMIplus', 'myDMIminus'], ['']);
```

```
      myxabove := ItSelf.Indicator ('myxabove');

      ItSelf.MakeIndicator ('myxbelow', 'xbelow',
                            ['myDMIplus', 'myDMIminus'], ['']);
      myxbelow := ItSelf.Indicator ('myxbelow');

      if myxabove.value [0] <> 0 then
      begin
         ItSelf.Plot [1] := Data1.High [0];
         ItSelf.Plot [2] := data1.Low  [0];
         ItSelf.Plot [3] := clGreen;
         exit;
      end;

      if myxbelow.value [0] <> 0 then
      begin
         ItSelf.Plot [1] := Data1.High [0];
         ItSelf.Plot [2] := data1.Low  [0];
         ItSelf.Plot [3] := clRed;
         exit;
      end;

      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;

   end;
```

VBScript

```
   function testvb()
   dim myxabove, myxbelow

      if not Data1.Valid (0) then
         ItSelf.SuccessEx (1) = false
         ItSelf.SuccessEx (2) = false
         ItSelf.SuccessEx (3) = false
         exit function
      end if

      ItSelf.MakeIndicator "myDMIplus",  "DMIplus",  Array("1"),
   Array(Param1.str)
      ItSelf.MakeIndicator "myDMIminus", "DMIminus", Array("1"),
   Array(param2.str)

      ItSelf.MakeIndicator "myxabove", "xabove", _
                           Array("myDMIplus", "myDMIminus"),
   Array("")
      myxabove = ItSelf.Indicator ("myxabove")

      ItSelf.MakeIndicator "myxbelow", "xbelow", _
                           Array("myDMIplus", "myDMIminus"),
   Array("")
      myxbelow = ItSelf.Indicator ("myxbelow")

      if myxabove.Value (0) <> 0 then
         ItSelf.Plot (1) = Data1.High (0)
         ItSelf.Plot (2) = Data1.Low  (0)
         ItSelf.Plot (3) = clGreen
         exit function
      end if

      if myxbelow.Value (0) <> 0 then
         ItSelf.Plot (1) = Data1.High (0)
         ItSelf.Plot (2) = Data1.Low  (0)
         ItSelf.Plot (3) = clRed
         exit function
      end if

      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false

   end function
```

Formula Language

```
myxabove := xabove (DMIplus (Data1, param1), DMIminus(Data1,
param2));
myxbelow := xbelow (DMIplus (Data1, param1), DMIminus(Data1,
param2));

success1 := if (myxabove <> 0 or myxbelow <> 0, 1, 0);
success2 := if (myxabove <> 0 or myxbelow <> 0, 1, 0);
success3 := if (myxabove <> 0 or myxbelow <> 0, 1, 0);

plot1 := High;
plot2 := low;
plot3 := if (myxabove <> 0, clGreen, if(myxbelow <> 0, clRed,
0));
```

Formula Column

```
DMIplus (m5, 14)
```

Directional Movement Index Plus (DMIplus)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'DMIplus', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "DMIplus", Array("N"),
Array("n")
```

Formula Language

```
DMIplus(DataN, n);
```

Formula Column

```
DMIplus(Tn, n)
```

Definition

It is the up component of the DMI. DMI Plus takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var myDMIplus, myDMIminus : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEX [1] := false;
     ItSelf.SuccessEX [2] := false;
     ItSelf.SuccessEX [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myDMIplus',  'DMIplus',  ['1'],
[param1.str]);
   myDMIplus  := ItSelf.Indicator ('myDMIplus');

   ItSelf.MakeIndicator ('myDMIminus', 'DMIminus', ['1'],
[param2.str]);
   myDMIminus := ItSelf.Indicator ('myDMIminus');
   if not (myDMIplus.Valid [0] and myDMIminus.Valid [0]) then
   begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
      exit;
   end;

   if myDMIplus.value [0] > myDMIminus.value [0] then
   begin
      ItSelf.Plot [1] := 1;
      ItSelf.Plot [2] := 0;
      ItSelf.Plot [3] := clGreen;
   end
   else if myDMIplus.value [0] < myDMIminus.Value [0] then
   begin
      ItSelf.Plot [1] := 1;
      ItSelf.Plot [2] := 0;
      ItSelf.Plot [3] := clRed;
   end
   else
   begin
      ItSelf.Plot [1] := 1;
      ItSelf.Plot [2] := 0;
      ItSelf.Plot [3] := clWhite;
   end;

end;
```

VBScript

```
function testvb()
dim myDMIplus, myDMIminus

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "myDMIplus",  "DMIplus",  Array("1"),
Array(Param1.str)
   myDMIplus = ItSelf.Indicator ("myDMIplus")

   ItSelf.MakeIndicator "myDMIminus", "DMIminus", Array("1"),
Array(param2.str)
   myDMIminus = ItSelf.Indicator ("myDMIminus")


   if not (myDMIplus.Valid (0) and myDMIminus.Valid (0)) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   if myDMIplus.Value (0) > myDMIminus.Value (0) then
      ItSelf.Plot (1) = 1
      ItSelf.Plot (2) = 0
      ItSelf.Plot (3) = clGreen
   elseif myDMIplus.Value (0) < myDMIminus.Value (0) then
      ItSelf.Plot (1) = 1
      ItSelf.Plot (2) = 0
      ItSelf.Plot (3) = clRed
   else
      ItSelf.Plot (1) = 1
      ItSelf.Plot (2) = 0
      ItSelf.Plot (3) = clWhite
   end if

end function
```

Formula Language

```
plot1 := 1;
plot2 := 0;
plot3 := if (DMIplus (Data1, param1) > DMIminus(Data1, param2),
clGreen,
         if (DMIplus (Data1, param1) < DMIminus(Data1, param2),
clRed, clWhite));
```

Formula Column

```
DMIplus (M5, 14)
```

Division (div)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'div', ['N', 'N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "div", Array("N", "N"),
Array("")
```

Formula Language

```
div(DataN, DataN);
```

Formula Column

Not supported.

Definition

Divides one series from another (Link 1 / Link 2).

Examples

Delphi Script

```
function test : double;
var myrangepct : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.Success := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myrange', 'range', ['1'], ['']);
   ItSelf.MakeIndicator ('mydiff',  'diff',  ['1', '1.1'],
['']);
   ItSelf.MakeIndicator ('myrangepct', 'div', ['mydiff',
'myrange'], ['']);
   myrangepct := ItSelf.Indicator ('myrangepct');

   if myrangepct.Valid [0] then
      result := myrangepct.Value [0]
   else
      ItSelf.Success := false;

end;
```

VBScript

```
function testvb()
dim myrangepct

   if not Data1.Valid (0) then
      ItSelf.Success = false
      exit function
   end if

   ItSelf.MakeIndicator "myrange",  "range",  Array("1"),
Array("")
   ItSelf.MakeIndicator "mydiff",  "diff",  Array("1", "1.l"),
Array("")
   ItSelf.MakeIndicator "myrangepct", "div", _
                        Array("mydiff", "myrange"), Array("")
   myrangepct = ItSelf.Indicator ("myrangepct")

   if myrangepct.Valid (0) then
      testvb = myrangepct.value (0)
   else
      itself.Success = false
   end if

end function
```

Formula Language

```
plot1 := div(diff(Data1, low), range(data1));
```

Double Exponential Moving Average (dema)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'dema', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "dema", Array("N"), Array("n")
```

Formula Language

```
dema(DataN, n);
```

Formula Column

```
dema(Tn, n)
```

Definition

Returns DEMA as described in Technical Analysis Stocks and Commodities magazine, V12.

Examples

Delphi Script

```
function test : double;
var myxabove, myxbelow : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('demafast', 'dema', ['1'],
[param1.str]);
   ItSelf.MakeIndicator ('demaslow', 'dema', ['1'],
[param2.str]);

   ItSelf.MakeIndicator ('myxabove', 'xabove', ['demafast',
'demaslow'], ['']);
   myxabove := ItSelf.Indicator ('myxabove');
   ItSelf.MakeIndicator ('myxbelow', 'xbelow', ['demafast',
'demaslow'], ['']);
   myxbelow := ItSelf.Indicator ('myxbelow');

   if myxabove.Value [0] <> 0 then
   begin
     ItSelf.Plot [1] := Data1.High [0];
     ItSelf.Plot [2] := Data1.Low  [0];
     ItSelf.plot [3] := clGreen;
     exit;
   end;

   if myxbelow.Value [0] <> 0 then
   begin
     ItSelf.Plot [1] := Data1.High [0];
     ItSelf.Plot [2] := Data1.Low  [0];
     ItSelf.plot [3] := clRed;
     exit;
   end;

   ItSelf.SuccessEx [1] := false;
   ItSelf.SuccessEx [2] := false;
   ItSelf.SuccessEx [3] := false;

end;
```

VBScript

```
function testvb()
dim myxabove, myxbelow

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "demafast", "dema", Array("1"),
Array(param1.str)
   ItSelf.MakeIndicator "demaslow", "dema", Array("1"),
Array(param2.str)

   ItSelf.MakeIndicator "myxabove", "xabove", _
                        Array("demafast", "demaslow"),
Array("")
   myxabove = ItSelf.Indicator ("myxabove")

   ItSelf.MakeIndicator "myxbelow", "xbelow", _
                        Array("demafast", "demaslow"),
Array("")
   myxbelow = ItSelf.Indicator ("myxbelow")

   if myxabove.Value (0) <> 0 then
      ItSelf.Plot (1) = Data1.High (0)
```

```
         ItSelf.Plot (2) = Data1.Low  (0)
         ItSelf.Plot (3) = clGreen
         exit function
      end if

      if myxbelow.Value (0) <> 0 then
         ItSelf.Plot (1) = Data1.High (0)
         ItSelf.Plot (2) = Data1.Low  (0)
         ItSelf.Plot (3) = clRed
         exit function
      end if

      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false

   end function
```

Formula Language

```
myxabove := xabove(dema(Data1, param1), dema(Data1, param2));
myxbelow := xbelow(dema(Data1, param1), dema(Data1, param2));

success1 := if (myxabove <> 0 or myxbelow <> 0, 1, 0);
success2 := if (myxabove <> 0 or myxbelow <> 0, 1, 0);
success3 := if (myxabove <> 0 or myxbelow <> 0, 1, 0);

plot1 := High;
plot2 := Low;
plot3 := if(myxabove, clGreen, if(myxbelow, clRed, 0));
```

Ease of Movement (ease)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'ease', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "ease", Array("N"), Array("")
```

Formula Language

```
ease(DataN);
```

Formula Column

```
ease(Tn)
```

Definition

Ease of Movement indicator indicates possible strength when it crosses from below zero to above zero and vice versa. Other usage includes finding specific levels of exhaustion and using that as a hint to identify turning points. Ease of Movement requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var myease, smoothease, myxabove, myxbelow : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myease', 'ease', ['1'], ['']);
   myease := ItSelf.Indicator ('myease');

   ItSelf.MakeIndicator ('smoothease', 'average', ['myease'],
[param1.str]);
```

```
   smoothease := ItSelf.Indicator ('smoothease');

   ItSelf.MakeIndicator ('zeroline', 'const', ['1'],
[param2.str]);

   ItSelf.MakeIndicator ('myxabove', 'xabove',
                          ['smoothease', 'zeroline'], ['']);
   myxabove := ItSelf.Indicator ('myxabove');

   ItSelf.MakeIndicator ('myxbelow', 'xbelow',
                          ['smoothease', 'zeroline'], ['']);
   myxbelow := ItSelf.Indicator ('myxbelow');

   if not myease.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     exit;
   end;

   ItSelf.Plot [1] := myease.Value [0];
   ItSelf.Plot [2] := smoothease.Value [0];

   if myxabove.Value [0] <> 0 then
      ItSelf.Plot [3] := Data1.Value [0]
   else
      ItSelf.SuccessEx [3] := false;

   if myxbelow.Value [0] <> 0 then
      ItSelf.Plot [4] := Data1.Value [0]
   else
      ItSelf.SuccessEx [4] := false;

end;
```

VBScript

```
function testvb()
dim myease, smoothease, myxabove, myxbelow

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      exit function
   end if

   ItSelf.MakeIndicator "myease", "ease", Array("1"), Array("")
   myease = ItSelf.Indicator ("myease")

   ItSelf.MakeIndicator "smoothease", "average", _
                        Array("myease"), Array(param1.str)
   smoothease = ItSelf.Indicator ("smoothease")
   ItSelf.MakeIndicator "zeroline", "const", Array("1"),
Array(param2.str)

   ItSelf.MakeIndicator "myxabove", "xabove", _
                        Array("smoothease", "zeroline"),
Array("")
   myxabove = ItSelf.Indicator ("myxabove")

   ItSelf.MakeIndicator "myxbelow", "xbelow", _
                        Array("smoothease", "zeroline"),
Array("")
   myxbelow = ItSelf.Indicator ("myxbelow")

    if not myease.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      exit function
   end if

   ItSelf.Plot (1) = myease.Value (0)
   ItSelf.Plot (2) = smoothease.Value  (0)

   if myxabove.Value (0) <> 0 then
      ItSelf.Plot (3) = Data1.Value (0)
   else
      ItSelf.SuccessEx (3) = false
   end if

   if myxbelow.Value (0) <> 0 then
      ItSelf.Plot (4) = Data1.Value (0)
   else
      ItSelf.SuccessEx (4) = false
   end if

end function
```

Formula Language

```
myease     := ease(Data1);
smoothease := average (myease, param1);
myxabove := xabove(smoothease, 0);
myxbelow := xbelow(smoothease, 0);

success3 := if (myxabove <> 0, 1, 0);
success4 := if (myxbelow <> 0, 1, 0);

plot1 := myease;
plot2 := smoothease;
plot3 := Data1;
plot4 := Data1;
```

Formula Column

```
ease (M5)
```

Exponential Average Range (xavgrange)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'xavgrange', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "xavgrange", Array("N"),
Array("")
```

Formula Language

```
xavgrange(DataN, n);
```

Formula Column

```
xavgrange(Tn, n)
```

Definition

Returns exponential moving average of the range of the data series. Exponential Average Range requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var myxavg, myxavglow, myxavgrange : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.Success := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myxavgrange', 'xavgrange', ['1'],
[param1.str]);
   myxavgrange := ItSelf.Indicator ('myxavgrange');

   ItSelf.MakeIndicator ('myxavg', 'xaverage', ['1'],
[param2.str]);
   myxavg := ItSelf.Indicator ('myxavg');

   ItSelf.MakeIndicator ('myxavglow', 'xaverage', ['1.l'],
[param3.str]);
   myxavglow := ItSelf.Indicator ('myxavglow');

   if not (myxavgrange.Valid [0] and myxavglow.Valid [0] and
           myxavg.Valid [0]) then
   begin
     ItSelf.Success := false;
     exit;
   end;

   if myxavgrange.Value [0] > 0 then
      result := (myxavg.Value [0] - myxavglow.Value
[0])/myxavgrange.Value [0]
   else
      ItSelf.Success := false;

end;
```

VBScript

```
function testvb()
dim xavgfast, xavgslow, myxabove, myxbelow

   if not Data1.Valid (0) then
      ItSelf.Success = false
      exit function
   end if

   ItSelf.MakeIndicator "myxavg", "xaverage", Array("1"),
Array(param1.str)
   myxavg = ItSelf.Indicator ("myxavg")

   ItSelf.MakeIndicator "myxavglow", "xaverage", Array("1.l"),
Array(param2.str)
   myxavglow = ItSelf.Indicator ("myxavglow")

   ItSelf.MakeIndicator "myxavgrange", "xavgrange", Array("1"),
Array(param3.str)
   myxavgrange = ItSelf.Indicator ("myxavgrange")

   if not (myxavg.Valid (0) and myxavgrange.Valid (0) and _
           myxavglow.Valid (0)) then
      ItSelf.Success = false
      exit function
   end if

   if myxavgrange.Value (0) > 0 then
      testvb = (myxavg.Value (0) - myxavglow.Value
(0))/myxavgrange.Value (0)
   else
      ItSelf.Success = false
   end if

end function
```

Formula Language

```
success1 := if (xavgrange(Data1, param1) > 0, 1, 0);

plot1 := (xaverage(Data1, param2) - xaverage(low, param3))/
          xavgrange(data1, param1);
```

Formula Column

```
xavgrange(m15, 20)
```

Exponential Moving Average (xaverage)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'xaverage', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "xaverage", Array("N"),
Array("n")
```

Formula Language

```
xaverage(DtatN, n);
```

Formula Column

```
xaverage(Tn, n)
```

Definition

Exponential Moving Average is effectively the mean of the complete data series from the beginning up to the point of calculation. Exponential Moving Average takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var xavgfast, xavgslow, myxabove, myxbelow : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('xavgfast', 'xaverage', ['1'],
[param1.str]);
   xavgfast := ItSelf.Indicator ('xavgfast');

   ItSelf.MakeIndicator ('xavgslow', 'xaverage', ['1'],
[param2.str]);
   xavgslow := ItSelf.Indicator ('xavgslow');

   ItSelf.MakeIndicator ('myxabove', 'xabove',
                        ['xavgfast', 'xavgslow'], ['']);
   myxabove := ItSelf.Indicator ('myxabove');

   ItSelf.MakeIndicator ('myxbelow', 'xbelow',
                        ['xavgfast', 'xavgslow'], ['']);
   myxbelow := ItSelf.Indicator ('myxbelow');

   if not (xavgfast.Valid [0] and xavgslow.Valid [0]) then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     exit;
   end;

   ItSelf.Plot [1] := xavgfast.Value [0];
   ItSelf.Plot [2] := xavgslow.Value [0];

   if myxabove.Value [0] <> 0 then
      ItSelf.Plot [3] := Data1.Value [0]
   else
      ItSelf.SuccessEx [3] := false;

   if myxbelow.Value [0] <> 0 then
      ItSelf.Plot [4] := Data1.Value [0]
   else
      ItSelf.SuccessEx [4] := false;

end;
```

VBScript

```
function testvb()
dim xavgfast, xavgslow, myxabove, myxbelow

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      exit function
   end if

   ItSelf.MakeIndicator "xavgfast", "xaverage", Array("1"),
Array(param1.str)
   xavgfast = ItSelf.Indicator ("xavgfast")

   ItSelf.MakeIndicator "xavgslow", "xaverage", Array("1"),
Array(param2.str)
   xavgslow = ItSelf.Indicator ("xavgslow")
   ItSelf.MakeIndicator "myxabove", "xabove", _
                        Array("xavgfast", "xavgslow"),
Array("")
   myxabove = ItSelf.Indicator ("myxabove")

   ItSelf.MakeIndicator "myxbelow", "xbelow", _
                        Array("xavgfast", "xavgslow"),
Array("")
   myxbelow = ItSelf.Indicator ("myxbelow")

   if not (xavgfast.Valid (0) and xavgslow.Valid (0)) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      exit function
   end if

   ItSelf.Plot (1) = xavgfast.Value (0)
   ItSelf.Plot (2) = xavgslow.Value (0)

   if myxabove.Value (0) <> 0 then
      ItSelf.Plot (3) = Data1.Value (0)
   else
      ItSelf.SuccessEx (3) = false
   end if

   if myxbelow.Value (0) <> 0 then
      ItSelf.Plot (4) = Data1.Value (0)
   else
      ItSelf.SuccessEx (4) = false
   end if

end function
```

Formula Language

```
xavgfast := xaverage (Data1, param1);
xavgslow := xaverage (Data1, param2);
myxabove := xabove(xavgfast, xavgslow);
myxbelow := xbelow(xavgfast, xavgslow);

success3 := if (myxabove <> 0, 1, 0);
success4 := if (myxbelow <> 0, 1, 0);

plot1 := xavgfast;
plot2 := xavgslow;
plot3 := Data1;
plot4 := Data1;
```

Formula Column

```
xaverage(M10, 3)
```

Formula (fml)

See Also

- ***Tutorial: Quickest Way to Plot Your Custom Expression Using the Formula Indicator*** (on page 362)
- ***Formula Topics and Tutorials*** (on page 255)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'fml', ['N'], ['formula']);
```

VBScript

```
ItSelf.MakeIndicator "string", "fml", Array("N"),
Array("formula")
```

Formula Language

```
fml(DataN, "formula");
```

Formula Column

```
fml(Tn, "formula")
```

Definition

Returns the evaluation of the expression defined in plot1. Formula requires one series (Link 1).

Examples

Delphi Script

```
function test : double;
var myavgprice : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.Success := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myprice', 'fml', ['1'],
['(h+l+c)/3']);
   ItSelf.MakeIndicator ('myavgprice', 'average', ['myprice'],
[param1.str]);
   myavgprice := ItSelf.Indicator ('myavgprice');

   if myavgprice.Valid [0] then
      result := myavgprice.Value [0]
   else
      ItSelf.Success := false;

end;
```

VBScript

```
function testvb()
dim myavgprice

   if not Data1.Valid (0) then
      ItSelf.Success = false
      exit function
   end if

   ItSelf.MakeIndicator "myprice", "fml", Array("1"),
Array("(H+L+C)/3")
   ItSelf.MakeIndicator "myavgprice", "average",
Array("myprice"), _
                        Array(param1.str)
   myavgprice = ItSelf.Indicator ("myavgprice")

   if myavgprice.Valid (0) then
      testvb = myavgprice.Value (0)
   else
      ItSelf.Success = false
   end if

end function
```

Formula Language

```
plot1 := average(fml(Data1, "(H+L+C)/3"), param1);
```

Formula Column

```
fml (m3, "(H+L+C)/3")
```

Formula 2 (fml2)

See Also

- ***Tutorial: Quickest Way to Plot Your Custom Expression Using the Formula Indicator*** (on page 362)
- ***Formula Topics and Tutorials*** (on page 255)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'fml2', ['N', 'N'],
['formula']);
```

VBScript

```
ItSelf.MakeIndicator "string", "fml2", Array("N", "N"),
Array("formula")
```

Formula Language

```
fml2(DataN, DataN, "fml2");
```

Formula Column

Not supported.

Definition

Returns the evaluation of the expression defined in plot1. Formula 2 requires two series.

Examples

Delphi Script

```
function test : double;
var myavgspread : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.Success := false;
     exit;
   end;
   ItSelf.MakeIndicator ('myspread', 'fml2', ['1', '2'],
      ['(data1.high+data1.low+data1.close)/3-
(data2.high+data2.low+data2.close)/3']);
   ItSelf.MakeIndicator ('myavgspread', 'average',
['myspread'], [param1.str]);
   myavgspread := ItSelf.Indicator ('myavgspread');

   if myavgspread.Valid [0] then
      result := myavgspread.Value [0]
   else
      ItSelf.Success := false;

end;
```

VBScript

```
function testvb()
dim myavgspread

   if not Data1.Valid (0) then
      ItSelf.Success = false
      exit function
   end if

   ItSelf.MakeIndicator "myspread", "fml2", Array("1", "2"), _
          Array("(Data1.High+data1.Low+data1.Close)/3-
(data2.high+Data2.low+data2.close)/3")
   ItSelf.MakeIndicator "myavgspread", "average",
Array("myspread"), _
                        Array(param1.str)
   myavgspread = ItSelf.Indicator ("myavgspread")

   if myavgspread.Valid (0) then
      testvb = myavgspread.Value (0)
   else
      ItSelf.Success = false
   end if

end function
```

Formula Language

```
plot1 := fml2 (Data1, Data2,
     "(Data1.High+Data1.Low+data1.close)/3-
            (data2.high+data2.low+data2.close)/3"),param1);
```

Gap Value (gap)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'gap', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndcator "string", "gap", Array("N"), Array("")
```

Formula Language

```
gap(DataN);
```

Formula Column

```
gap(Tn)
```

Definition

This indicator calculates the day gap value of a security. It return positive value when the security is gap up, negative value if the security is gap down, otherwise it returns 0. A gap occurs if yesterday's high is greater than today's open or yesterday's low is less than today's open. This indicator requires one data series (Link 1).

Examples

Delphi Script

```
function test : double;
var mygap : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mygap', 'gap', ['1'], ['']);
   mygap := ItSelf.Indicator ('mygap');

   if mygap.Valid [0] and (Data1.Date [1] <> Data1.Date [0])
then
   begin
```

```
     ItSelf.Plot [1] := data1.high [0];
     ItSelf.Plot [2] := data1.low  [0];
     if mygap.Value [0] > 0 then
        ItSelf.Plot [3] := clGreen
     else
        ItSelf.Plot [3] := clRed;
   end
   else
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
   end;

end;
```

VBScript

```
function testvb()
dim mygap

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mygap", "gap", Array("1"), Array("")
   mygap = ItSelf.Indicator ("mygap")

   if mygap.Valid (0) and (Data1.Date (1) <> Data1.Date (0))
then
      ItSelf.Plot (1) = data1.high (0)
      ItSelf.Plot (2) = data1.low  (0)
      if mygap.Value (0) > 0 then
         ItSelf.Plot (3) = clGreen
      else
         ItSelf.Plot (3) = clRed
      end if
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
Success1 := if ( gap(Data1) <> 0 and
                 dayofweek (DateTime(0)) <> dayofweek
(DateTime(1)), 1, 0);
Success2 := if ( gap(Data1) <> 0 and
                 dayofweek(DateTime(0)) <>
dayofweek(DateTime(1)), 1, 0);
```

```
Success3 := if ( gap(Data1) <> 0 and
                 dayofweek(DateTime(0)) <>
dayofweek(DateTime(1)), 1, 0);

plot1 := High;
plot2 := Low;
plot3 := if (gap(data1) > 0, clGreen, if (gap(data1) < 0,
clRed, clBlack));
```

Formula Column

```
gap(M5)
```

Gap Down (gapdown)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'gapdown', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "gapdown", Array("N"), Array("")
```

Formula Language

```
gapdown(DataN);
```

Formula Column

```
gapdown(Tn)
```

Definition

This indicator returns a +1 when the security's price gap down. Otherwise it returns a 0. A gap down occurs when today's open is lower than yesterday's low. This indicator require one data series (Link 1).

Examples

Delphi Script

```
function test : double;
var mygapdown : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mygapdown', 'gapdown', ['1'], ['']);
   mygapdown := ItSelf.Indicator ('mygapdown');

   if mygapdown.Valid [0] and (Data1.Date [1] <> Data1.Date
[0]) and
      (mygapdown.Value [0] <> 0) then
   begin
```

```
      ItSelf.Plot [1] := data1.high [0];
      ItSelf.Plot [2] := data1.low  [0];
      ItSelf.Plot [3] := clRed;
    end
    else
    begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
    end;

end;
```

VBScript

```
function testvb()
dim mygapdown

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mygapdown", "gapdown", Array("1"),
Array("")
   mygapdown = ItSelf.Indicator ("mygapdown")

   if (mygapdown.Value (0) > 0) and _
      (Data1.Date (1) <> Data1.Date (0)) then
      ItSelf.Plot (1) = data1.high (0)
      ItSelf.Plot (2) = data1.low  (0)
      ItSelf.Plot (3) = clGreen
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
Success1 := if (gapdown(Data1) > 0, 1, 0);
Success2 := if (gapdown(Data1) > 0, 1, 0);
Success3 := if (gapdown(Data1) > 0, 1, 0);

plot1 := high;
plot2 := low;
plot3 := clRed;
```

Formula Column

```
gapdown(M10)
```

Gap Up (gapup)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'gapup', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator "string", "gapup", Array("N"), Array("")
```

Formula Language

```
gapup(DataN);
```

Formula Column

```
gapup(Tn)
```

Definition

This indicator returns a +1 when a security's price gap up. Otherwise it returns a 0. A gap up is when today's open is higher then previous day high. This indicator requires one data series (Link 1).

Examples

Delphi Script

```
function test : double;
var mygapup : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mygapup', 'gapup', ['1'], ['']);
   mygapup := ItSelf.Indicator ('mygapup');

   if (Data1.Date [1] <> Data1.Date [0]) and
      (mygapup.Value [0] > 0) then
   begin
     ItSelf.Plot [1] := data1.high [0];
```

```
      ItSelf.Plot [2] := data1.low  [0];
      ItSelf.Plot [3] := clGreen;
    end
    else
    begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      ItSelf.SuccessEx [3] := false;
    end;

end;
```

VBScript

```
function testvb()
dim mygapup

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      exit function
   end if

   ItSelf.MakeIndicator "mygapup", "gapup", Array("1"),
Array("")
   mygapup = ItSelf.Indicator ("mygapup")

   if (mygapup.Value (0) > 0) and _
      (Data1.Date (1) <> Data1.Date (0)) then
      ItSelf.Plot (1) = data1.high (0)
      ItSelf.Plot (2) = data1.low  (0)
      ItSelf.Plot (3) = clGreen
   else
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
   end if

end function
```

Formula Language

```
Success1 := if (gapup(Data1) > 0, 1, 0);
Success2 := if (gapup(Data1) > 0, 1, 0);
Success3 := if (gapup(Data1) > 0, 1, 0);

plot1 := high;
plot2 := low;
plot3 := clGreen;
```

Formula Column

```
Gapup(M15)
```

Highest (highest)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'highest', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator "string", "highest", Array("N"), Array("")
```

Formula Language

```
highest(DataN);
```

Formula Column

```
highest(Tn)
```

Definition

Calculates the highest value in the data series since the first bar loaded in the chart. This indicator requires one data series (Link 1).

Examples

Delphi Script

```
function dp_highest : double;
var myhighest : variant;
begin

   if not data1.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   myhighest := itself.makeindicator ('myh', 'highest', ['1'],
['']);

   if myhighest.value [1] < data1.high [0] then
      result := Data1.high [0]
   else
      itself.success := false;

end;
```

VBScript

```
function vb_highestex()
dim myhighest

   if not data1.valid (0) then
      itself.success = false
      exit function
   end if

   myhighest = itself.makeindicator ("myh", "highest",
Array("1"), Array(""))

   if myhighest.value (1) < data1.high (0) then
      vb_highestex = Data1.high (0)
   else
      itself.success = false
   end if

end function
```

Formula Language

```
plot1 := if(highest(1, data1) < high, high, 0);
success1 := if(highest(1, data1) < high, 1, 0);
```

Formula Column

```
highest(M1)
```

Highest Bar (highestb)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'highestb', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator "string", "highestb", Array("N"),
Array("")
```

Formula Language

```
highestb(DataN);
```

Formula Column

```
higestb(Tn)
```

Definition

Calculates the number of bars has passed since the data series highest value. This includes all data loaded in the chart. This indicator requires one data series (Link 1).

Examples

Delphi Script

```
function dp_highestbar : double;
var myhighestbar : variant;
begin
   if not data1.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   myhighestbar := ItSelf.MakeIndicator ('myhb', 'highestb',
['1'], ['']);

   if myhighestbar.value [0] = 9 then
      result := Data1.high [0]
   else
      itself.success := false;
end;
```

VBScript

```
function vb_highestbar()
dim myhighestb

   if not data1.valid (0) then
      itself.success = false
      exit function
   end if

   myhighestb = itself.makeindicator ("myhb", "highestb",
Array("1"), Array(""))

   if myhighestb.value (0) = 9 then
      vb_highestbar = Data1.high (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := if(highestb(Data1) = 9, high, 0);
success1 := if(highestb(Data1) = 9, 1, 0);
```

Formula Column

```
highestb(M1)
```

Highest High Bar (hhb)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'hhb', ['N'], ['Period']);
```

VBScript

```
Itself.MakeIndicator ("string", "hhb", Array("N"),
Array("Period"))
```

Formula Language

```
hhb(DataN, Period);
```

Formula Column

```
hhb(Tn, Period)
```

Definition

Returns the Highest High Bar over Period. Highest High Bar requires one series (Link 1).

Examples

Delphi Script

```
function hhb_ex : double;
var BarsAfter : integer;
var HHBPeriod : string;
var myhhb;
begin
   HHBPeriod := param1.str;
   BarsAfter := param2.int;
   myhhb := itself.MakeIndicator ('myhhbz', 'hhb', ['1'],
[HHBPeriod]);
   if not myhhb.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   if myhhb.value [0] = BarsAfter then
      result := Data1.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function vb_hhb()
dim BarsAfter
dim myhhb
   BarsAfter = param2.int
   myhhb = itself.MakeIndicator ("hhbz", "hhb", Array("1"),
Array(param1.str))
   if not myhhb.valid (0) then
      itself.success = false
      exit function
   end if
   if myhhb.value (0) = BarsAfter then
      vb_hhb = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
HHBPeriod := param1;
BarsAfter := param2;
plot1 := if(hhb(data1, HHBPeriod) = BarsAfter, data1, 0);
Success1 := if(hhb(data1, HHBPeriod) = BarsAfter, 1, 0);
```

Formula Column

```
hhb(M3, 10)
```

Highest High Value (hhv)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'hhv', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "hhv", Array("N"), Array(""))
```

Formula Language

```
hhv(DataN);
```

Formula Column

```
hhv(Tn)
```

Definition

Returns the Highest High Value over Period. Highest High Value requires one series (Link 1).

Examples

Delphi Script

```
function ex_hhv_del : double;
var myhhv;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;
   myhhv := itself.Makeindicator ('xhhv', 'hhv', ['1.high'],
[param1.str]);
   if not myhhv.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;
   if myhhv.value [0] > myhhv.value [1] then
   begin
      itself.plot [1] := 1;
      itself.plot [0] := 0;
      itself.plot [3] := clLime;
   end
   else
   begin
      itself.plot [1] := 1;
      itself.plot [2] := 0;
      itself.plot [3] := clRed;
   end;
end;
```

VBScript

```
   function ex_hhv_vb()
   dim myhhv

      if not data1.valid (0) then
         itself.successall = false
         exit function
      end if

      myhhv = Itself.MakeIndicator ("xhhv", "hhv",
   Array("1.high"), Array(Param1.str))

      if not myhhv.valid (0) then
         itself.successall = false
         exit function
      end if

      if myhhv.value (0) > myhhv.value (1) then
         itself.plot (1) = 1
         itself.plot (2) = 0
         itself.plot (3) = clLime
      else
         itself.plot (1) = 1
         itself.plot (2) = 0
         itself.plot (3) = clRed
      end if
   end function
```

Formula Language

```
   plot1 := 1;
   plot2 := 0;
   plot3 := if(hhv(data1.h,param1) > hhv(1,data1.h,param1),
   clLime, clRed);
```

Formula Column

```
   hhv(0, M15, 20)
```

Highlight 1 Delphi (highlight)

Highlight 1 Delphi is a legacy indicator. Use Highlight Formula instead.

Highlight 2 Delphi (highlight2)

Highlight 2 Delphi is a legacy indicator. Use Highlight 2 Formula instead.

Highlight Bar Formula (highlightbarfml)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'highlightbarfml', ['N'],
['fml']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "highlightbarfml", ["N"],
["fml"])
```

Formula Language

```
highlightbarfml (DataN, "fml");
```

Formula Column

```
highlightbarfml (Tn, "fml")
```

Definition

Highlight Bar Formula highlights the complete bar from High to Low if Condition is true. Highlight Bar Formula requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_highlightbarfml : double;
var pricediff : double;
var hlbfml;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;
   hlbfml := Itself.MakeIndicator ('xhlbfml',
'highlightbarfml',
                           ['1'], [param1.str]);

   if not hlbfml.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;
   result :=  hlbfml.valueex [1, 0] - hlbfml.valueex [2, 0];
end;
```

VBScript

```
function ex_vb_highlightbarfml()
dim hlbfml

   if not data1.valid (0) then
      itself.success = false
      exit function
   end if

   hlbfml = ItSelf.MakeIndicator ("xhlbfml", "highlightbarfml",
_
                                  Array("1"),
Array(param1.str))
   if not hlbfml.valid (0) then
      itself.success = false
      exit function
   end if

   ex_vb_highlightbarfml = hlbfml.ValueEx (1, 0) -
hlbfml.ValueEx (2, 0)
end function
```

Formula Language

```
plot1 := highlightbarfml.Plot1(0,data1,param1) -
highlightbarfml.Plot2(0,data1,param1);
```

Formula Column

```
highlightbarfml (M15, "high > high(1) and low > low(1)")
```

Highlight Formula (highlightfml)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'highlightfml', ['N'],
['fml','High']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "highlightfml", Array("N"),
Array("fml", "High"))
```

Formula Language

```
highlightfml(DataN, "fml", "High");
```

Formula Column

```
highlightfml(Tn, "fml", "High")
```

Definition

Highlight Formula draws a marker onto Position High or Low if Condition is true. Highlight Formula requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_highlightfml : double;
var bull_harami, bear_harami : variant;
begin
   bull_harami := itself.makeIndicator ('tiu', 'highlightfml',
['1'],
                  ['(o(1)>c(1)) and (o(1)>h(0)) and (c(1)<l(0))
and (c(0)>o(0))', 'High']);

   bear_harami := itself.makeIndicator ('tid', 'highlightfml',
['1'],
                  ['(o(1)<c(1)) and (c(1)>h(0)) and (o(1)<l(0))
and (c(0)<o(0))', 'Low']);

   if bull_harami.valid [0] then
   begin
     itself.plot [1] := Data1.High [0];
      itself.successex [2] := false;
   end
   else if bear_harami.valid [0] then
   begin
      itself.successex [1] := false;
      itself.plot [2] := data1.low [0];
   end
   else
   begin
      itself.successex [1] := false;
      itself.successex [2] := false;
   end;
end;
```

VBScript

```
function ex_vb_highlightfml()
dim bull_harami, bear_harami

   bull_harami = itself.MakeIndicator ("buh", "highlightfml",
Array("1"), _
                 Array("(o(1)>c(1)) and (o(1)>h(0)) and
(c(1)<l(0)) and (c(0)>o(0))", "High"))

   bear_harami = itself.MakeIndicator ("beh", "highlightfml",
Array("1"), _
                 Array("(o(1)<c(1)) and (c(1)>h(0)) and
(o(1)<l(0)) and (c(0)<o(0))", "Low"))


   if bull_harami.valid (0) then
      itself.plot (1) = data1.high (0)
      itself.successex (2) = false
   elseif bear_harami.valid (0) then
      itself.successex (1) = false
      itself.plot (2) = data1.low (0)
   else
      itself.successex (1) = false
      itself.successex (2) = false
   end if

end function
```

Formula Language

```
bull_harami := highlightfml(data1,
              "(o(1)>c(1)) and (o(1)>h(0)) and
         (c(1)<l(0)) and (c(0)>o(0))", "High");

bear_harami := highlightfml(data1,
              "(o(1)<c(1)) and (c(1)>h(0)) and
         (o(1)<l(0)) and (c(0)<o(0))", "Low");

plot1 := bull_harami;
plot2 := bear_harami;
```

Formula Column

```
Highlightfml(M5, "(o(1)>c(1)) and (o(1)>h(0)) and
                    (c(1)<l(0)) and (c(0)>o(0))", "High")
```

Historical Volatility (Connor) (historical_vola)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'historical_vola', ['N'],
['Period', 'N']);
```

VBScript

```
Itself.MakeIndicator ("string", "historical_vola", Array("N"),
Array("Period", "N"))
```

Formula Language

```
historical_vola(DataN, Period, N);
```

Formula Column

```
historical_vola(Tn, Period, N)
```

Definition

Historical Volatility as explained in the book Street Smart by Laurence A. Connors and Linda Bradford Raschke.

Examples

Delphi Script

```
function ex_del_historical_vola : double;
var histvola1, histvola2 : variant;
var histvolacompare : double;
begin

   histvola1 := ItSelf.MakeIndicator ('hv1', 'historical_vola',
                                      ['1'], [param1.str,
'1']);
   histvola2 := ItSelf.MakeIndicator ('hv2', 'historical_vola',
                                      ['1'], [param2.str,
'1']);

   if not histvola1.valid [0] or not histvola2.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;
   if histvola2.value [0] <> 0 then
      histvolacompare := histvola1.value [0]/ histvola2.value
[0];

   itself.plot [1] := histvolacompare;
   itself.plot [2] := histvola1.value [0];
   itself.plot [3] := histvola2.value [0];

   if histvolacompare < 0.5 then
      itself.plot [4] := histvolacompare
   else
      itself.successex [4] := false;

end;
```

VBScript

```
function ex_vbs_historical_vola()
dim histvola1, histvola2, histvolacompare

   histvola1 = itself.MakeIndicator ("hv1", "historical_vola",
_
                                            Array("1"),
Array(param1.str, "1"))
   histvola2 = itself.MakeIndicator ("hv2", "historical_vola",
_
                                            Array("1"),
Array(param2.str, "1"))

   if not histvola1.valid (0) or not histvola2.valid (0) then
      itself.successall = false
      exit function
   end if

   if histvola2.value (0) <> 0 then
      histvolacompare = histvola1.value (0)/ histvola2.value
(0)
   end if

   itself.plot (1) = histvolacompare
   itself.plot (2) = histvola1.value (0)
   itself.plot (3) = histvola2.value (0)

   if histvolacompare < 0.5 then
      itself.plot (4) = histvolacompare
   else
      itself.successex (4) = false
   end if
end function
```

Formula Language

```
plot2 := historical_vola(data1, param1, 1);
plot3 := historical_vola(data1, param2, 1);
plot1 := if (plot3 <> 0, plot2/plot3, 0);
plot4 := if (plot1 < 0.5, plot1, 0);
```

```
success4 := if (plot1 < 0.5, 1, 0);
```

Formula Column

```
historical_vola (0, M15, 6, 1)/historical_vola (0, M15, 100, 1)
```

Historical Volatility Ratio (histvol_ratio)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'histvol_ratio', ['N'], ['N',
'N', 'N']);
```

VBScript

```
Itself.MakeIndicator ("string", "histvol_ratio", Array("N"),
Array("N", "N", "N"))
```

Formula Language

```
histvol_ratio(dataN, N, N, N);
```

Formula Column

```
histvol_ratio(Tn, N, N, N)
```

Definition

Historical Volatility Ratio as explained in the book Street Smart by Laurence A. Connors and Linda Bradford Raschke.

Examples

Delphi Script

```
function ex_del_histvol_ratio : double;
var histvolratio, IDNR4, dayseries : varint;
begin
   histvolratio := itself.makeindicator ('hvr1',
'histvol_ratio',
                                          ['1'], [param1.str,
param2.str, '1']);

   Itself.CompressSeries ('dsday', '1', ppDaily, 1);

   IDNR4 := Itself.Makeindicator ('xIDNR4', 'fml', ['$dsday'],
            ['(h(3)-l(3))>(h(0)-l(0)) and (h(2)-l(2))>(h(0)-
l(0)) and (h(1)-l(1))>(h(0)-l(0)) and (h(1)>h(0)) and
(l(0)>l(1))']);

   if histvolratio.valid [0] then
      itself.plot [1] := histvolratio.value [0]
   else
   begin
      itself.successall := false;
      exit;
   end;

   if (IDNR4.value [0] > 0) and (histvolratio.value [0] < 0.5)
then
      itself.plot [2] := histvolratio.value [0]
   else
      itself.successex [2] := false;
end;
```

VBScript

```
function ex_vbs_histvolratio()
dim histvolratio, IDNR4
   histvolratio = itself.MakeIndicator ("hvr", "histvol_ratio",
Array("1"), _
                         Array(param1.str, param2.str, "1"))
   itself.CompressSeries "dsday", "1", ppDaily, 1
   IDNR4 = itself.MakeIndicator ("xIDNR4", "fml",
Array("$dsday"), _
           Array("(h(3)-l(3))>(h(0)-l(0)) and (h(2)-
l(2))>(h(0)-l(0)) and (h(1)-l(1))>(h(0)-l(0)) and (h(1)>h(0))
and (l(0)>l(1))"))

   if histvolratio.valid (0) then
      itself.plot (1) = histvolratio.value (0)
   else
      itself.successall = false
      exit function
   end if

   if (IDNR4.value(0)>0) and (histvolratio.value(0)<0.5) then
      itself.plot (2) = histvolratio.value (0)
   else
      itself.successex (2) = false
   end if
end function
```

Formula Language

```
plot1 := histvol_ratio (data1, param1, param2, 1);
IDNR4 := fml(data1, "(h(3)-l(3))>(h(0)-l(0)) and (h(2)-
l(2))>(h(0)-l(0)) and (h(1)-l(1))>(h(0)-l(0)) and (h(1)>h(0))
and (l(0)>l(1))");

plot2 := if ((IDNR4>0) and (plot1<0.5), plot1, 0);
success2 := if (IDNR4 > 0, 1, 0);
```

Formula Column

```
histvol_ratio (M3, 6, 100, 1)
```

Inside Bar (insidebar)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'insidebar', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator ("string", "insidebar", Array("N"),
Array(""))
```

Formula Language

```
insidebar(DataN)
```

Formula Column

```
insidebar(Tn)
```

Definition

Inside Bar returns 1 when the current bar is an inside bar. Inside Bar requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_insidebar : double;
var ib, nr4 : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   ib  := itself.MakeIndicator ('myib', 'insidebar', ['1'],
['']);
   nr4 := itself.MakeIndicator ('mynr4', 'fml', ['1'],
          ['((h(3)-l(3))>(h(0)-l(0))) and ((h(2)-l(2))>(h(0)-
l(0))) and ((h(1)-l(1))>(h(0)-l(0)))']);

   if (ib.value [0] > 0) and (nr4.value [0] > 0) then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low  [0];
      itself.plot [3] := clLime;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function ex_vbs_insidebar()
dim ib, nr4
   if not data1.valid (0) then
      itself.successall = false
      exit function
   end if

   ib  = itself.MakeIndicator ("myib", "insidebar", Array("1"),
Array(""))
   nr4 = itself.MakeIndicator ("mynr4", "fml", Array("1"), _
         Array("((h(3)-l(3))>(h(0)-l(0))) and ((h(2)-
l(2))>(h(0)-l(0))) and ((h(1)-l(1))>(h(0)-l(0)))"))

   if (ib.value(0)>0) and (nr4.value(0)>0) then
       itself.plot (1) = data1.high (0)
       itself.plot (2) = data1.low  (0)
       itself.plot (3) = clLime
   else
       itself.successall = false
   end if
end function
```

Formula Language

```
ib := insidebar(data1);
nr4 := fml(data1, "((h(3)-l(3))>(h(0)-l(0))) and ((h(2)-
l(2))>(h(0)-l(0))) and ((h(1)-l(1))>(h(0)-l(0)))");

plot1 := if(ib > 0 and nr4 > 0, h, 0);
plot2 := if(ib > 0 and nr4 > 0, l, 0);
plot3 := clLime;

success1 := if (ib > 0 and nr4 > 0, 1, 0);
success2 := if (ib > 0 and nr4 > 0, 1, 0);
success3 := if (ib > 0 and nr4 > 0, 1, 0);
```

Formula Column

```
insidebar(M15)
```

Keltner Channel (keltner)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'keltner', ['N'], ['N','N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "keltner", Array("N"),
Array("N", "N"))
```

Formula Language

```
keltner (dataN, N, N);
```

Formula Column

```
keltner (Tn, N, N)
```

Definition

Returns Keltner Channel. Keltner Channel is a channel indicator based on exponential moving average and true range of the underlying price series. Keltner Channel requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_keltner : double;
var keltnerslow, keltnerfast, iscross : variant;
begin
   keltnerfast := ItSelf.MakeIndicator ('kcfast', 'keltner',
['1'], [param1.str,'2']);
   keltnerslow := ItSelf.MakeIndicator ('kcslow', 'keltner',
['1'], [param2.str,'2']);
   iscross := ItSelf.MakeIndicator ('myx', 'xabove', ['kcfast',
'kcslow'], ['']);
   if iscross.value [0] > 0 then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low  [0];
      itself.plot [3] := clLime;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function ex_vbs_keltner()
dim keltnerfast, keltnerslow, iscross

    keltnerfast = itself.MakeIndicator ("kcf", "keltner",
Array("1"), Array(param1.str,"2"))
    keltnerslow = itself.MakeIndicator ("kcs", "keltner",
Array("1"), Array(param2.str,"2"))
    iscross = itself.MakeIndicator ("myx", "xabove",
Array("kcf", "kcs"), Array(""))
    if iscross.value (0) > 0 then
       itself.plot (1) = data1.high (0)
       itself.plot (2) = data1.low (0)
       itself.plot (3) = clLime
    else
       itself.successall = false
    end if
end function
```

Formula Language

```
myx := xabove(keltner(data1, param1, 2), keltner(data1, param2,
2));
plot1 := h;
plot2 := l;
plot3 := clLime;
success1 := if (myx > 0, 1, 0);
success2 := if (myx > 0, 1, 0);
success3 := if (myx > 0, 1, 0);
```

Formula Column

```
keltner(M13, 20, 2)
```

Keltner Channel 3 Lines (keltner3)

Syntax

Delphi Script

```
Itself.MakeIndicator ('string', 'keltner3', ['N'], ['N', 'N']);
```

VBScript

```
Itself.MakeIndicator ("string", "keltner3", Array("N"),
Array("N", "N"))
```

Formula Language

```
keltner3 (dataN, N, N)
```

Formula Column

```
keltner3 (Tn, N, N)
```

Definition

Keltner Channels 3 Lines is a channel indicator based on exponential moving average and true range of the underlying price series. Keltner Channel 3 returns 3 plot series. This indicator requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_keltner3 : double;
var mykeltner3 : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;
   mykeltner3 := itself.MakeIndicator ('myk3', 'keltner3',
['1'], [param1.str, param2.str]);
   if data1.high [0] > mykeltner3.valueEx [3, 0] then
   begin
      itself.plot [1] := data1.high [0];
      itself.SuccessEx [2] := false;
   end
   else if data1.low [0] < mykeltner3.valueEx [2, 0] then
   begin
      itself.SuccessEx [1] := false;
      itself.plot [2] := data1.low [0];
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function ex_vbs_keltner3()
dim mykeltner3
    if not data1.valid (0) then
       itself.successall = false
       exit function
    end if
    mykeltner3 = itself.MakeIndicator ("myk", "keltner3",
Array("1"), _
                                             Array(param1.str,
param2.str))
    if data1.high (0) > mykeltner3.ValueEx (3, 0) then
       itself.plot (1) = data1.high (0)
       itself.successex (2) = false
    elseif data1.low (0) < mykeltner3.valueEx (2, 0) then
       itself.successex (1) = false
       itself.plot (2) = data1.low (0)
    else
       itself.successall = false
    end if
end function
```

Formula Language

```
plot1 := if(h > keltner3.Plot3(data1,20,2), h, 0);
plot2 := if(l < keltner3.Plot2(data1,20,2), l, 0);
success1 := if(h > keltner3.Plot3(data1,20,2), 1, 0);
success2 := if(l < keltner3.Plot2(data1,20,2), 1, 0);
```

Formula Column

```
if(h(M5)>keltner3.plot3(M5, 20, 2), h,
if(l(M5)<keltner.plot2(0,M5,20,2), l, 0))
```

Latch (latch)

Latch indicator is a memory effect indicator. Latch's plot initially is set to the initial value parameter. When condition 1 parameter becomes true, latch's plot is set to the value of value 1 parameter. When condition 2 parameter becomes true, latch's plot is set to the value of value 2 parameter. In any case, latch will retains its plot value even when the conditions become false, i.e. Latch's memory effect. Latch is useful in quote window formula where you cannot generate indicator series within the formula.

Linear Regression Channel (linregchannel)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'linregchannel', ['N'],
['N','N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "linregchannel", Array("N"),
Array("N","N"))
```

Formula Language

```
linregchannel(dataN, N, N);
```

Formula Column

```
linregchannel(Tn, N, N)
```

Definition

Returns the value of SD standard deviation away from the specified linear regression projection. Linear Regression Channel requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_linregchannel : double;
var drawid, OPoint : integer;
    linregval : variant;
begin
   if heap.size = 0 then
   begin
     DrawingObjects.Deleteall;
     heap.allocate (1);
   end;

   linregval := itself.MakeIndicator ('mylr',
               'linregchannel', ['1'], ['20','1']);

   if data1.islastbar then
   begin
     OPoint := param1.int-1;
     drawid := DrawingObjects.Add (cotTrendline);
     DrawingObjects.Setpoint (drawid, 0,
         Data1.Datetime [OPoint], Data1.close [OPoint]);
     DrawingOBjects.Setpoint (drawid, 1,
         Data1.Datetime [0], linregval.value [0]);
     drawingobjects.penwidth [drawid] := 3;
     drawingobjects.visible [drawid] := true;
   end;
   result := Data1.Value [0];
end;
```

VBScript

```
function ex_vbs_linregchannel()
dim drawid, OPoint
dim linregval
   if heap.size = 0 then
      heap.allocate (0)
      DrawingObjects.DeleteAll
   end if
   linregval = ItSelf.MakeIndicator ("mylr", "linregchannel", _
                                     Array("1"),
Array("20","1"))

   if data1.islastbar then
      OPoint = param1.int-1
      drawid = DrawingObjects.Add(cotTrendline)
      DrawingOBjects.SetPoint drawid, 0,
Data1.Datetime(OPoint), _
                                         Data1.Close(OPoint)
      DrawingObjects.SetPoint drawid, 1, Data1.Datetime(0), _
                                         linregval.value (0)
      DrawingObjects.PenWidth(drawid) = 3
      DrawingObjects.Visible(drawid) = true
   end if
   ex_vbs_linregchannel = Data1.Value (0)
end function
```

Formula Language

```
plot1 := if(linregchannel(data1, param1, param2) > c, high, 0);
plot2 := if(linregchannel(data1, param1, param2) < c, low, 0);
success1 := if(linregchannel(data1, param1, param2) > c, 1, 0);
success2 := if(linregchannel(data1, param1, param2) < c, 1, 0);
```

Formula Column

```
linregchannel(M2, 20, 1)
```

Linear Regression Channel 3 Lines (linregchannel3)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'linregchannel3', ['N'],
['N','N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "linregchannel3", Array("N"),
Array("N","N"))
```

Formula Language

```
linregchannel3(DataN, N, N);
```

Formula Column

```
linregchannel3(Tn, N, N)
```

Definition

Returns the complete channel of SD standard deviation away from the specified linear regression projection. Linear Regression Channel 3 Lines requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_linregchannel3 : double;
var lrc : variant;
    tight_band, wide_band : double;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   lrc := itself.MakeIndicator ('mylrc', 'linregchannel3',
['1'], ['21','2']);
   tight_band := param1.real;
   wide_band  := param1.real;

   if (lrc.valueEx[3,0]-lrc.valueEx[2,0]) < tight_band then
   begin
```

```
         itself.plot [1] := data1.high [0];
         itself.plot [2] := data1.low [0];
         itself.plot [3] := clRed;
      end
      else if (lrc.ValueEx[3,0]-lrc.ValueEx[2,0]) > wide_band then
      begin
         itself.plot [1] := data1.high [0];
         itself.plot [2] := data1.low [0];
         itself.plot [3] := clLime;
      end
      else
         itself.successall := false;
   end;
```

VBScript

```
function ex_vbs_linregchannel3()
dim lrc
   if not data1.valid (0) then
      itself.successall = false
      exit function
   end if
   lrc = itself.MakeIndicator ("mylrc", "linregchannel3", _
                     Array("1"), Array("20","2"))
   tight_band = param1.real
   wide_band  = param2.real
   if (lrc.ValueEx(3,0)-lrc.ValueEx(2,0)) < tight_band then
      itself.plot(1) = data1.high (0)
      itself.plot(2) = data1.low (0)
      itself.plot(3) = clRed
   elseif (lrc.ValueEx(3,0)-lrc.ValueEx(2,0)) > wide_band then
      itself.plot(1) = data1.high (0)
      itself.plot(2) = data1.low (0)
      itself.plot(3) = clLime
   else
      itself.successall = false
   end if
end function
```

Formula Language

```
bandrange := linregchannel3.plot3(data1,20,2)            -
linregchannel3.plot2(data1,20,2);

plot1 := if((bandrange<param1)or(bandrange>param2),high,0);
plot2 := if((bandrange<param1)or(bandrange>param2),low,0);
plot3 :=
if(bandrange<param1,clRed,if(bandrange>param,clLime,0));

success1 := if((bandrange<param1)or(bandrange>param2),1,0);
success2 := if((bandrange<param1)or(bandrange>param2),1,0);
success3 := if((bandrange<param1)or(bandrange>param2),1,0);
```

Formula Column

```
linregchannel3.plot3(M3,20,2)-linregchannel3.plot2(M3,20,2)
```

Linear Regression Constant (linconst)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'linconst', ['N'], ['N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "linconst", Array("N"),
Array("N"))
```

Formula Language

```
linconst(DataN, N);
```

Formula Column

```
linconst(Tn, N)
```

Definition

Similar to linear regression value, it returns the constant of the approximated linear equation. Linear Regression Constant takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_linconst : double;
var slowlc, fastlc, myxabove : variant;
begin

   fastlc := itself.MakeIndicator ('flc', 'linconst', ['1'],
[param1.str]);
   slowlc := itself.MakeIndicator ('slc', 'linconst', ['1'],
[param2.str]);
   myxabove := itself.MakeIndicator ('xa', 'xabove',
['flc','slc'], ['']);

   if myxabove.value [0] > 0 then
      result := Data1.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_linconst()
dim slowlc, fastlc, myxabove

   fastlc = ItSelf.MakeIndicator("flc", "linconst", Array("1"),
Array(param1.str))
   slowlc = ItSelf.MakeIndicator("slc", "linconst", Array("1"),
Array(param2.str))
   myxabove = ItSelf.MakeIndicator("xa", "xabove",
Array("flc","slc"), Array(""))

   if myxabove.value(0) > 0 then
      ex_vbs_linconst = Data1.Value (0)
   else
      itself.success = false
   end if
end function
Formula Language
plot1 :=
if(xabove(linconst(data1,param1),linconst(data1,param2)) > 0,
c, 0);
success1 :=
if(xabove(linconst(data1,param1),linconst(data1,param2)) > 0,
1, 0);
```

Formula Column

```
fml(M3,"xabove(linconst(data1,20),linconst(data1,40))")
```

Linear Regression Slope (linslope)

Syntax

Delphi Script

```
ItSelf.MakIndicator ('string', 'linslope', ['N'], ['N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "linslope", Array("N"),
Array("N"))
```

Formula Language

```
linslope(DataN, N);
```

Formula Clumn

```
linslope(Tn, N)
```

Definition

Similar to linear regression value, it returns the slope of the approximation. Linear Regression Slope takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_linslope : double;
var mylslope : variant;
    ls0, ls1 : double;
begin

   mylslope := itself.MakeIndicator ('myls', 'linslope', ['1'],
[param1.str]);

   if not mylslope.valid [0] or not mylslope.valid [1] then
   begin
      itself.success := false;
      exit;
   end;

   ls0 := mylslope.value [0];
   ls1 := mylslope.value [1];
   if (ls0 > 0) and (ls1 < 0) then //cross above zero
      result := 1
   else if (ls0 < 0) and (ls1 > 0) then
      result := -1
   else
      result := 0
end;
```

VBScript

```
function ex_vbs_linslope()
dim mylslope
   mylslope = ItSelf.MakeIndicator ("myls", "linslope", _
                                    Array("1"),
Array(param1.str))

   if not (mylslope.valid(0) and mylslope.valid(1)) then
      itself.success = false
      exit function
   end if

   ls0 = mylslope.value (0)
   ls1 = mylslope.value (1)

   if (ls0>0) and (ls1<0) then
       ex_vbs_linslope = 1
   elseif (ls0<0) and (ls1>0) then
       ex_vbs_linslope = -1
   else
       ex_vbs_linslope = 0
   end if
end function
```

Formula Language

```
ls := linslope(data1,param1);
plot1 := if((ls>0)and(ls(1)<0),1,if((ls<0)and(ls(1)>0),-1,0));
```

Formula Column

```
if((linslope(M12,20)>0) and (linslope(1,M12,20)<0),1,
    if((linslope(M12,20)<0) and (linslope(1,M12,20)>0),-1,0))
```

Linear Regression Value (linreg)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'linreg', ['N'], ['N','N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "linreg", Array("N"),
Array("N","N"))
```

Formula Language

```
linreg(DataN, N, N);
```

Formula Column

```
linreg(Tn, N, N)
```

Definition

Given Period of bars and number of bars to project into the future (Projection), linear regression value returns the projected value based on the least-square approximation method. Linear Regression Value requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_linreg : double;
var fast_linreg, slow_linreg : variant;
begin

   fast_linreg := Itself.MakeIndicator ('flr', 'linreg', ['1'],
[param1.str,'0']);
   slow_linreg := ItSelf.MakeIndicator ('slr', 'linreg', ['1'],
[param2.str,'0']);

   if not (fast_linreg.valid [0] and slow_linreg.valid [0])
then
   begin
      itself.success := false;
      exit;
   end;

   result := slow_linreg.value[0] - fast_linreg.value[0];
end;
```

VBScript

```
function ex_vbs_linreg()
dim slow_linreg, fast_linreg
   fast_linreg = ItSelf.MakeIndicator ("flr", "linreg", _
                     Array("1"), Array(param1.str,"0"))
   slow_linreg = ItSelf.MakeIndicator ("slr", "linreg", _
                     Array("1"), Array(param2.str,"0"))
   if not (fast_linreg.valid(0) and slow_linreg.valid(0)) then
      itself.success = false
      exit function
   end if
   ex_vbs_linreg = slow_linreg.value(0)-fast_linreg.value(0)
end function
```

Formula Language

```
plot1 := linreg(data1,param2,0)-linreg(data1,param1,0);
```

Formula Column

```
linreg(M3,20,0)-linreg(M3,10,0)
```

Lowest (lowest)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'lowest', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "lowest", Array("N"),
Array(""))
```

Formula Language

```
lowest(DataN);
```

Formula Column

```
lowest(Tn)
```

Definition

Returns the lowest value of a data series since the first bar loaded in the chart. This indicator require one data series (Link 1).

Examples

Delphi Script

```
function ex_del_lowest : double;
var mylow : variant;
begin

   if heap.size = 0 then
   begin
      heap.allocate (1);
      heap.value [0] := 0;
   end;

   mylow := ItSelf.MakeIndicator ('myl', 'lowest', ['1.1'],
['']);

   if not (mylow.valid [0] and mylow.valid [1]) then
   begin
      itself.success := false;
      exit;
   end;
```

```
      if (mylow.value [0] <> mylow.value [1]) then
      begin
         result := mylow.value[1]-mylow.value[0];
         heap.value [0] := result;
      end
      else
        result := heap.value [0];
   end;
```

VBScript

```
function ex_vbs_lowest()
dim mylow

   if heap.size = 0 then
      heap.allocate(1)
      heap.value(0) = 0
   end if

   mylow = itself.Makeindicator ("myl", "lowest", Array("1.l"),
Array(""))

   if not (mylow.valid(0) and mylow.valid(1)) then
      itself.success = false
      exit function
   end if

   if mylow.value(0) <> mylow.value(1) then
      ex_vbs_lowest = mylow.value(1)-mylow.value(0)
      heap.value(0) = ex_vbs_lowest
   else
      ex_vbs_lowest = heap.value(0)
   end if
end function
```

Formula Language

```
mylow  := lowest(data1.l);
mylow1 := lowest(1, data1.l);
mydiff := if(mylow<>mylow1, mylow1-mylow, mydiff(1));
plot1 := mydiff;
```

Formula Column

```
lowest(M3.L)-lowest(1,M3.L)
```

Lowest Bar (lowestb)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'lowestb', ['N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "lowestb", Array("n"),
Array(""))
```

Formula Language

```
lowest(DataN);
```

Formula Column

```
lowest(Tn)
```

Definition

Calculates the number of bars that have passed form the lowest value of a data series. This indicator require one data series (Link 1)

Examples

Delphi Script

```
function ex_del_lowestb : double;
var mylowb : variant;
begin
   if not data1.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   mylowb := itself.MakeIndicator ('mylb', 'lowestb', ['1.l'],
['']);

   if mylowb.value [0] = param1.int then
      result := Data1.low [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_lowestb()
dim mylowb

   if not data1.valid(0) then
      itself.success = false
      exit function
   end if

   mylowb = ItSelf.MakeIndicator ("mylb", "lowestb",
Array("1.l"), Array(""))

   if mylowb.value(0) = param1.int then
      ex_vbs_lowestb = Data1.low (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := if(lowestb(data1.l)=param1, Low, 0);
success1 := if(lowestb(data1.l)=param1, 1, 0);
```

Formula Column

```
if(lowestb(M3.L)=15, low, 0);
```

Lowest Low Bar (llb)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'llb', ['N'], ['N']);
```

VBScript

```
ItSelf.MakeIndicator ("string","llb", Array("N"), Array("N"))
```

Formula Language

```
llb(DataN, N);
```

Formula Column

```
llb(Tn, N)
```

Definition

Returns the Lowest Low Bar over Period. Lowest Low Bar requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_llb : double;
var myllb : variant;
    mymessage : string;
begin

   if not data1.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   myllb := ItSelf.MakeIndicator ('mllb', 'llb', ['1.l'],
[param1.str]);

   if myllb.value [1] = param2.int then
   begin
      result := Data1.low [0];
      mymessage := 'Low of ' + param2.str + 'th lowest low bar
with value ' +
                    NTlib.double2str(data1.low [0]);
      itself.alert('High',mymessage,clLime,clBlack,false);
   end
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_llb()
dim myllb
   if not data1.valid(0) then
      itself.success = false
      exit function
   end if
   myllb = ItSelf.MakeIndicator ("xllb", "llb", Array("1.l"),
Array(param1.str))

   if myllb.value(1) = param2.int then
      ex_vbs_llb = Data1.low (0)
      mymessage = "Low of " + param2.str + "th lowest low bar
with value " + _
                    NTlib.double2str(data1.low(0))
      itself.alert "high", mymessage, clLime, clBlack, false
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := if(llb(1, data1.l, param1) = param2, low, 0);
success := if (llb(1, data1.l, param1) = param2, 1, 0);
```

Formula Column

```
if(llb(M3.l, 14)=5, M3.l, 0)
```

Lowest Low Value (llv)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'llv', ['N'], ['N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "llv", Array("N"), Array("N"))
```

Formula Language

```
llv(DataN, N);
```

Formula Column

```
llv(Tn, N)
```

Definition

Returns the Lowest Low Value over Period. Lowest Low Value requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_llv : double;
var myllv : variant;
begin

   if not data1.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   myllv := itself.MakeIndicator ('xllv', 'llv', ['1.l'],
[param1.str]);

   if data1.low [0] < myllv.value [1] then
   begin
      result := Data1.low [0];
      itself.alert('high', 'Breakout Low', clLime, clBlack,
false);
   end
   else
```

```
      itself.success := false;
end;
```

VBscript

```
function ex_vbs_llv()
dim myllv

   if not data1.valid(0) then
      itself.success = false
      exit function
   end if

   myllv = ItSelf.MakeIndicator ("xllv", "llv", _
                                 Array("1.l"),
Array(param1.str))

   if data1.low(0) < myllv.value(1) then
      ex_vbs_llv = Data1.low (0)
      itself.alert "high", "Breakout Low", clLime, clBlack,
false
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := if(low < llv(1, data1.l, param1), low, 0);
success1 := if(low < llv(1, data1.l, param1), 1, 0);
```

Formula Column

```
low(M30)<llv(1, M30, 14)
```

Moving Average Convergence Divergence (MACD)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'MACD', ['N'], ['type', 'N',
'type', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "MACD", Array("N"),
Array("type", "N", "type", "N"))
```

Formula Language

```
macd(DataN, "type", N, "type", N);
```

Formula Column

```
macd(Tn, "type", N, "type", N)
```

Definition

MACD is a price momentum indicator. It is based on the difference between two moving averages. Usually a moving average is applied to the MACD as a signal line. MACD takes four parameters. MA1_type is the type of the first moving average. MA1_period is the period of the first moving average. MA2_type is the type of the second moving average. MA2_period is the period of the second moving average. MACD requires one series (Link 1).

Examples

Delpih Script

```
function ex_del_MACD : double;
var my_macd, signal_line : variant;
begin
   my_macd := Itself.MakeIndicator('macd1', 'MACD', ['1'],
                     ['Exponential', param1.str, 'Exponential',
param2.str]);
   signal_line := Itself.Makeindicator('avg1', 'xaverage',
                                        ['macd1'],
[param3.str]);
   if not(my_macd.valid[1] and signal_line.valid[0]) then
   begin
      itself.successall := false;
      exit;
   end;

   Itself.plot[1] := my_macd.value[0];
   ItSelf.plot[2] := signal_line.value[0];
   Itself.plot[3] := Itself.plot[1]-itself.plot[2];
end;
```

VBScript

```
function ex_vbs_macd()
dim my_macd, signal_line

   my_macd = Itself.MakeIndicator("macd1", "MACD", Array("1"),
_
                    Array("Exponential", param1.str,
"Exponential", param2.str))
   signal_line = Itself.MakeIndicator("signal1", "xaverage", _
                                       Array("macd1"),
Array(param3.str))
   if not (my_macd.valid(0) and signal_line.valid(0)) then
      itself.successall = false
      exit function
   end if

   itself.plot(1) = my_macd.value(0)
   itself.plot(2) = signal_line.value(0)
   itself.plot(3) = itself.plot(1)-itself.plot(2)
end function
```

Formula Language

```
my_macd := macd(data1, "Exponential", param1, "Exponential",
param2);
signal_line := xaverage(my_macd, param3);
plot1 := my_macd;
plot2 := signal_line;
plot3 := plot1-plot2;
success1 := if(valid(my_macd, 0) > 0 and valid(signal_line, 0)
> 0, 1, 0);
success2 := if(valid(my_macd, 0) > 0 and valid(signal_line, 0)
> 0, 1, 0);
success3 := if(valid(my_macd, 0) > 0 and valid(signal_line, 0)
> 0, 1, 0);
```

Formula Column

```
macd(D1, "Exponential", 12, "Exponential", 26)
```

Mass Index (mass)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'mass', ['N'], ['N','N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "mass", Array("N"), Array("N",
"N"))
```

Formula Language

```
mass(dataN, N, N);
```

Formula Column

```
mass(Tn, N, N)
```

Definition

Returns the Mass Index. Period1 is the exponential moving average period of the daily ranges. Period2 is the smoothing factor period. Mass Index requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_mass : double;
var mymass : variant;
var revers_level : double;
begin
   mymass := itself.MakeIndicator ('mym', 'mass', ['1'],
                                   [param1.str, param2.str]);

   revers_level := param3.real;
   if (mymass.value [1] > revers_level) and
      (mymass.value [0] < revers_level) then
      result := Data1.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_mass()
dim mymass
```

```
   mymass = itself.MakeIndicator ("mym", "mass", Array("1"), _
                                  Array(param1.str,
param2.str))

   revers_level = param3.real
   if (mymass.value(1) > revers_level) and _
      (mymass.value(0) < revers_level) then
      ex_vbs_mass = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
mymass := mass(data1, param1, param2);
revers_level := param3;
plot1 := if((mymass(1)>revers_level) and
             (mymass<revers_level), data1, 0);
success1 := if((mymass(1)>revers_level) and
             (mymass<revers_level), 1, 0);
```

Formula Column

```
(mass(1, M13, 10, 20)>26.5) and (mass(0, M13, 10, 20)<26.5)
```

Maximum (max)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'max', ['N','N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator ("strgin", "max", Array("N", "N"),
Array(""))
```

Formula Language

```
max(DataN, DataN);
```

Formula Column

N/A

Definition

Returns the largest of the two series.

Examples

Delphi Script

```
function ex_del_max : double;
var fast_ma, slow_ma, mymax : variant;
begin
   fast_ma := itself.MakeIndicator ('fma', 'mov', ['1'],
                                    ['Simple', param1.str]);
   slow_ma := itself.MakeIndicator ('sma', 'mov', ['1'],
                                    ['Simple', param2.str]);
   mymax := itself.MakeIndicator ('mmax', 'max', ['fma',
'sma'], ['']);

   result := mymax.Value [0];
end;
```

VBScript

```
function ex_vbs_max()
dim fast_ma, slow_ma, mymax

   fast_ma = itself.MakeIndicator ("fma", "mov", Array("1"), _
                                   Array("Simple", param1.str))
   slow_ma = itself.MakeIndicator ("sma", "mov", Array("1"), _
                                   Array("Simple", param2.str))
   mymax = itself.MakeIndicator ("mmax", "max", Array("fma",
"sma"), Array(""))

   ex_vbs_max = mymax.Value (0)
end function
```

Formula Language

```
plot1 := max(mov(data1,"Simple",param1),
mov(data1,"Simple",param2));
```

Mean Deviation (meandev)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'meandev', ['N'], ['N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "meandev", Array("N"),
Array("N"))
```

Formula Language

```
meandev(DataN, N);
```

Formula Column

```
meandev(Tn, N)
```

Definition

Returns the statistics mean deviation over the specified Period. Mean Deviation requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_meandev : double;
var band_value : variant;
begin
   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   band_value := itself.MakeIndicator ('bv', 'meandev', ['1'],
[param1.str]);

   if band_value.valid [0] then
   begin
      itself.plot [1] := Data1.Value [0]+ band_value.value[0];
      itself.plot [2] := Data1.Value [0]- band_value.value[0];
   end
   else
      itself.successall := false;
```

```
end;
```

VBScript

```
function ex_vbs_meandev()
dim band_val

   if not data1.valid (0) then
      itself.successall = false
      exit function
   end if

   band_val = Itself.MakeIndicator ("bv", "meandev", _
                                    Array("1"),
Array(param1.str))

   if band_val.valid (0) then
      itself.plot (1) = Data1.Value (0) + band_val.value (0)
      itself.plot (2) = Data1.Value (0) - band_val.value (0)
   else
      itself.successall = false
   end if
end function
```

Formula Language

```
plot1 := c+meandev(data1, param1);
plot2 := c-meandev(data1, param1);
```

Formula Column

```
meandev(M3, 20)
```

Median (median)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'median', ['N'], ['N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "median", Array("N"),
Array("N"))
```

Formula Language

```
median(dataN, N);
```

Formula Column

```
median(Tn, N)
```

Definition

Median returns the middle value of the data series sorted over Period of bars. Median requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_median : double;
var median_val : variant;
begin
   if not data1.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   median_val := ItSelf.MakeIndicator ('mv', 'median', ['1'],
[param1.str]);

   if median_val.valid [0] and (median_val.value [0] =
data1.value [0]) then
      result := Data1.Value [0]
   else
      itself.success := false;

end;
```

VBScript

```
function ex_vbs_median()
dim median_val

   if not Data1.Valid (0) then
      itself.success = false
      exit function
   end if

   median_val = ItSelf.MakeIndicator ("mv", "median", _
                                      Array("1"),
Array(param1.str))
   if median_val.valid (0) and (median_val.value(0) =
data1.value(0)) then
      ex_vbs_median = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := c;
success1 := if(c=median(data1,param1), 1, 0);
```

Formula Column

```
median(M13, 20)
```

MidPoint(midpoint)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'midpoint', ['N'], ['N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "midpoint", Array("N"),
Array("N"))
```

Formula Language

```
midpoint(dataN, N);
```

Formula Column

```
midpoint(Tn, N)
```

Definition

Midpoint returns the middle point of the price range from the lowest low to the highest high over the specified Period of bars. Midpoint requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_midpoint : double;
var xo, xu : variant;
begin

   Itself.MakeIndicator('mp1', 'midpoint', ['1'],
[param1.str]);
   Itself.MakeIndicator('ma1', 'average', ['1'], [param1.str]);

   xo := Itself.MakeIndicator('xo1', 'xabove', ['mp1', 'ma1'],
['']);
   xu := Itself.MakeIndicator('xu1', 'xbelow', ['mp1', 'ma1'],
['']);

   if xo.value[0] > 0 then
   begin
      itself.plot[1] := data1.high[0];
      itself.plot[2] := data1.low[0];
      itself.plot[3] := clgreen;
   end
   else if xu.value[0] > 0 then
```

```
      begin
         itself.plot[1] := data1.high[0];
         itself.plot[2] := data1.low[0];
         itself.plot[3] := clRed;
      end
      else
         itself.successall := false;
   end;
```

VBScript

```
function ex_vbs_min()
dim min_o, min_h, min_l, min_c

   min_o = itself.MakeIndicator ("min1", "min", Array("1.o",
"2.o"), Array(""))
   min_h = itself.MakeIndicator ("min2", "min", Array("1.h",
"2.h"), Array(""))
   min_l = itself.MakeIndicator ("min3", "min", Array("1.l",
"2.l"), Array(""))
   min_c = itself.MakeIndicator ("min4", "min", Array("1.c",
"2.c"), Array(""))

   if not data1.valid(0) then
      itself.successall = false
      exit function
   end if

   itself.plot(1) = min_o.value(0)
   itself.plot(2) = min_h.value(0)
   itself.plot(3) = min_l.value(0)
   itself.plot(4) = min_c.value(0)

end function
```

Formula Language

```
$xo := xabove(midpoint(data1, param1), average(data1, param1));
$xu := xbelow(midpoint(data1, param1), average(data1, param1));

plot1 := if(($xo > 0) or ($xu > 0), h, 0);
plot2 := if(($xo > 0) or ($xu > 0), l, 0);
plot3 := if($xo > 0, clgreen, if ($xu > 0, clred, 0));

Success1 := if(($xo > 0) or ($xu > 0), 1, 0);
Success2 := if(($xo > 0) or ($xu > 0), 1, 0);
Success3 := if(($xo > 0) or ($xu > 0), 1, 0);
```

Formula Column

```
midpoint(M1, 10)
```

Minimum (min)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'min', ['N','N'], ['']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "min", Array("N", "N"),
Array(""))
```

Formula Language

```
min(dataN, dataN);
```

Formula Column

N/A

Definition

Returns the smallest of the two data series.

Examples

Delphi Script

```
function ex_del_min : double;
var min_o, min_h, min_l, min_c : variant;
begin

   min_o := itself.Makeindicator ('min1', 'min', ['1.o',
'2.o'], ['']);
   min_h := itself.Makeindicator ('min2', 'min', ['1.h',
'2.h'], ['']);
   min_l := itself.Makeindicator ('min3', 'min', ['1.l',
'1.l'], ['']);
   min_c := itself.Makeindicator ('min4', 'min', ['1.c',
'2.c'], ['']);

   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;
   itself.plot[1] := min_o.Value[0];
   itself.plot[2] := min_h.value[0];
   itself.plot[3] := min_l.value[0];
   itself.plot[4] := min_c.value[0];
end;
```

VBScript

```
function ex_vbs_min()
dim min_o, min_h, min_l, min_c

   min_o = itself.MakeIndicator ("min1", "min", Array("1.o",
"2.o"), Array(""))
   min_h = itself.MakeIndicator ("min2", "min", Array("1.h",
"2.h"), Array(""))
   min_l = itself.MakeIndicator ("min3", "min", Array("1.l",
"2.l"), Array(""))
   min_c = itself.MakeIndicator ("min4", "min", Array("1.c",
"2.c"), Array(""))

   if not data1.valid(0) then
      itself.successall = false
      exit function
   end if

   itself.plot(1) = min_o.value(0)
   itself.plot(2) = min_h.value(0)
   itself.plot(3) = min_l.value(0)
   itself.plot(4) = min_c.value(0)

end function
```

Formula Language

```
plot1 := min(data1.o, data2.o);
plot2 := min(data1.h, data2.h);
plot3 := min(data1.l, data2.l);
plot4 := min(data1.c, data2.c);
```

Momentum(mo)

Syntax

Delphi Script

ItSelf.MakeIndicator ('string', 'mo', ['N'], ['N']);

VBScript

ItSelf.MakeIndicator ("string", "mo", Array("N"), Array("N"))

Formula Language

mo(dataN, N);

Formula Column

mo(Tn, N)

Definition

Momentum is the raw price difference between the current value of the data series and the value of the data series Period ago. It is similar to the Rate of Change indicator. Momentum requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_mo : double;
var mo_top, mo_bottom : variant;
begin

   Itself.MakeIndicator('mo1', 'mo', ['1'], [param1.str]);

   mo_top := Itself.MakeIndicator('top1', 'hhb', ['mo1'],
[param2.str]);
   mo_bottom := Itself.MakeIndicator('bot1', 'llb', ['mo1'],
[param2.str]);

   // Within 5 bars of the momentum top
   if mo_top.value[0] < param3.int then
   begin
```

```
         itself.plot[1] := data1.high[0];
         itself.plot[2] := data1.low[0];
         itself.plot[3] := clgreen;
      end
      else if mo_bottom.value[0] < param3.int then
      begin
         itself.plot[1] := data1.high[0];
         itself.plot[2] := data1.low[0];
         itself.plot[3] := clred;
      end
      else
         itself.successall := false;
   end;
```

VBScript

```
function ex_vbs_mo()
dim top_mo, bottom_mo

   Itself.MakeIndicator "mo1", "mo", Array("1"),
Array(param1.str)

   top_mo = Itself.Makeindicator("top1", "hhb", _
                                 Array("mo1"),
Array(param2.str))
   bottom_mo = ItSelf.MakeIndicator("bot1", "llb", _
                                 Array("mo1"),
Array(param2.str))

   if top_mo.value(0) < param3.int then
      itself.plot(1) = data1.high(0)
      itself.plot(2) = data1.low(0)
      itself.plot(3) = clgreen
   elseif bottom_mo.value(0) < param3.int then
      itself.plot(1) = data1.high(0)
      itself.plot(2) = data1.low(0)
      itself.plot(3) = clred
   else
      itself.successall = false
   end if

end function
```

Formula Language

```
$top_mo := hhb(mo(data1, param1), param2);
$bottom_mo := llb(mo(data1, param1), param2);
plot1 := if(($top_mo < param3) or ($bottom_mo < param3), h, 0);
plot2 := if(($top_mo < param3) or ($bottom_mo < param3), l, 0);
plot3 := if($top_mo < param3, clgreen, if($bottom_mo < param3,
clred, 0));
Success1 := if(($top_mo < param3) or ($bottom_mo < param3), 1,
0);
Success2 := if(($top_mo < param3) or ($bottom_mo < param3), 1,
0);
Success3 := if(($top_mo < param3) or ($bottom_mo < param3), 1,
0);
```

Formula Column

```
mo(D1, 10)
```

Money Flow Index (moneyflow)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'moneyflow', ['N'], ['N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "moneyflow", Array("N"),
Array("N"))
```

Formula Language

```
moneyflow(dataN, N);
```

Formula Column

```
moneyflow(Tn, N)
```

Definition

Money Flow Index is similar to RSI with volume taken into consideration. It can be interpreted in a way similar to the RSI. Money Flow Index requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_moneyflow : double;
var mf : variant;
begin

   mf := Itself.MakeIndicator ('mf1', 'moneyflow', ['1'],
[param1.str]);

   if not data1.valid[0] then
   begin
      itself.successall := false;
      exit;
   end;

   if mf.value[0] > 80 then
   begin
      itself.plot[1] := data1.high[0];
      itself.plot[2] := data1.low[0];
      itself.plot[3] := clgreen;
   end
   else if mf.value[0] < 20 then
   begin
      itself.plot[1] := data1.high[0];
      itself.plot[2] := data1.low[0];
      itself.plot[3] := clred;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function ex_vbs_moneyflow()
dim mf
    mf = ItSelf.MakeIndicator("mf1", "moneyflow", _
                              Array("1"), Array(param1.str))

    if mf.value(0) > 80 then
       itself.plot(1) = data1.high(0)
       itself.plot(2) = data1.low(0)
       itself.plot(3) = clgreen
    elseif mf.value(0) < 20 then
       itself.plot(1) = data1.high(0)
       itself.plot(2) = data1.low(0)
       itself.plot(3) = clred
    else
       itself.successall = false
    end if
end function
```

Formula Language

```
$mf := moneyflow(data1, param1);
plot1 := if(($mf > 80) or ($mf < 20), h, 0);
plot2 := if(($mf > 80) or ($mf < 20), l, 0);
plot3 := if($mf > 80, clgreen, if($mf < 20, clred, 0));
success1 := if(($mf > 80) or ($mf < 20), 1, 0);
success2 := if(($mf > 80) or ($mf < 20), 1, 0);
success3 := if(($mf > 80) or ($mf < 20), 1, 0);
```

Formula Column

```
moneyflow(M1, 10)
```

Moving Average (mov)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'mov', ['N'], ['type', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "mov", Array("N"), Array("type",
"N"))
```

Formula Language

```
mov(DataN, "type", N);
```

Formula Column

```
mov(Tn, "type", N)
```

Definition

All Moving Average Type of a specified period. Moving Average requires one specified series (Link 1).

Examples

Delphi Script

```
function ex_del_mov : double;
var double_mov : variant;
begin
   Itself.MakeIndicator('mov1', 'mov', ['1'], ['Exponential',
param1.str]);
   double_mov := Itself.MakeIndicator('mov2', 'mov',
                          ['mov1'], ['Exponential', param1.str]);
   if double_mov.valid [0] then
      result := double_mov.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_mov()
dim double_mov
   ItSelf.MakeIndicator "mov1", "mov", Array("1"), _
                        Array("Exponential", param1.str)
   double_mov = Itself.MakeIndicator("mov2", "mov",
Array("mov1"), _
                        Array("Exponential", param1.str))
   if double_mov.valid(0) then
      ex_vbs_mov = double_mov.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := mov(mov(data1, "Exponential", param1), "Exponential",
param1);
```

Formula Column

```
mov(mov(D1, "Exponential", 10), "Exponential", 10)
```

Moving Average 2 Lines (mov2)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'mov2', ['N'],
['type','N','N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "mov2", Array("N"),
Array("type","N","N"))
```

Formula Language

```
mov2(DataN,"type",N,N);
```

Formula Column

```
mov2(Tn,"type",N,N)
```

Definition

Moving averages for 2 different periods. Moving Average 2 Lines requires one specified series (Link 1).

Examples

Delphi Script

```
function ex_del_mov2 : double;
var ma2line : variant;
begin
   ma2line := ItSelf.MakeIndicator('ma2', 'mov2', ['1'],
                      ['Exponential', param1.str, param2.str]);
   if ma2line.valid[0] then
      result := ma2line.ValueEx[2, 0]-ma2line.ValueEx[1, 0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_mov2()
dim ma2line
   ma2line = Itself.MakeIndicator ("ma2", "mov2", Array("1"), _
                     Array("Exponential", param1.str,
param2.str))
   if ma2line.valid(0) then
```

```
      ex_vbs_mov2 = ma2line.ValueEx(2, 0)-ma2line.ValueEx(1, 0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := mov2.plot2(data1, "Exponential", param1, param2)-
         mov2.plot1(data1, "Exponential", param1, param2);
```

Formula Column

```
mov2.Plot2(0,M1,"Simple",10,20)
```

Moving Average 3 Lines (mov3)

Delphi Script and VBScript examples here use Trade object, which is not available to EOD version. For Delphi Script and VBScript examples for moving average, adapt the scripts from Moving Average 2.

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'mov3', ['N'],
['type','N','N','N']);
```

VBScript

```
ItSelf.MakeIndicator ("string", "mov3", Array("N"),
Array("type","N","N","N"))
```

Formula Language

```
mov3(DataN,"type",N,N,N);
```

Formula Column

```
mov3(Tn,"type",N,N,N)
```

Definition

Moving averages for 3 different periods. Moving Average 3 Lines requires one specified series (Link 1).

Examples

Delphi Script

```
function ex_del_mov3 : double;
Const Isxabove = 0;
      Isxbelow = 1;
var ma3line : variant;
    ff1, ff0, f1, f0, s1, s0 : integer;
begin

   ma3line := ItSelf.MakeIndicator('ma3', 'mov3', ['1'],
                        ['Exponential', param1.str, param2.str,
param3.str]);

   if heap.size = 0 then
   begin
```

```
      heap.allocate(2);
      heap.fill (0, 1, 0);
   end;

   if not (ma3line.validex[1, 0] and ma3line.validex[1, 1] and
           ma3line.validex[2, 0] and ma3line.validex[2, 1] and
           ma3line.validex[3, 0] and ma3line.validex[3, 1])
then
   begin
      itself.success := false;
      exit;
   end;

   ff1 := ma3line.valueex[1, 1];
   f1  := ma3line.valueex[2, 1];
   s1  := ma3line.valueex[3, 1];

   ff0 := ma3line.valueex[1, 0];
   f0  := ma3line.valueex[2, 0];
   s0  := ma3line.valueex[3, 0];

   if (ff1 < s1) and (ff0 > s0) then
   begin
      heap.value[Isxabove] := 1;
      heap.value[Isxbelow] := 0;
   end;

   if (ff1 > s1) and (ff0 < s0) then
   begin
      heap.value[Isxabove] := 0;
      heap.value[Isxbelow] := 1;
   end;

   if trade.openpositionflat then
   begin
      if (f1 < s1) and (f0 > s1) and
         (heap.value[Isxabove] > 0) then
      begin
         trade.longatmarket(100, 'Enter Long');
         heap.value[Isxabove] := 0;
      end;

      if (f1 > s1) and (f0 < s0) and
         (heap.value[Isxbelow] > 0) then
      begin
         trade.shortatmarket(100, 'Enter Short');
         heap.value[Isxbelow] := 0;
      end;

   end
   else
   begin
      if (ff1 > f1) and (ff0 < f0) and
         trade.openpositionlong then
          trade.longexitatmarket(100, 'Exit Long');
      if (ff1 < f1) and (ff0 > f0) and
          trade.openpositionshort then
          trade.shortexitatmarket(100, 'Exit Short');
   end;

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_mov3()
Const Isxabove = 0
      Isxbelow = 1
dim ma3line

   ma3line = Itself.MakeIndicator("ma3", "mov3", Array("1"), _
                    Array("Exponential", param1.str,
param2.str, param3.str))

   if heap.size = 0 then
      heap.allocate(2)
      heap.fill 0, 1, 0
   end if

   if not (ma3line.ValidEx(1, 0) and ma3line.ValidEx(1, 1) and
_
           ma3line.ValidEx(2, 0) and ma3line.ValidEx(2, 1) and
_
           ma3line.ValidEx(3, 0) and ma3line.ValidEx(3, 1))
then
      itself.success = false
      exit function
   end if

   ff1 = ma3line.ValueEx(1, 1)
   f1  = ma3line.ValueEx(2, 1)
   s1  = ma3line.ValueEx(3, 1)

   ff0 = ma3line.ValueEx(1, 0)
   f0  = ma3line.ValueEx(2, 0)
   s0  = ma3line.ValueEx(3, 0)

   if (ff1 < s1) and (ff0 > s0) then
      heap.value(Isxabove) = 1
      heap.value(Isxbelow) = 0
   end if

   if (ff1 > s1) and (ff0 < s0) then
      heap.value(Isxabove) = 0
      heap.Value(Isxbelow) = 1
   end if

   if trade.openpositionflat then
      if (f1 < s1) and (f0 > s0) and _
         (heap.value(Isxabove) > 0) then
         heap.value(Isxabove) = 0
```

```
            trade.longatmarket 100, "Enter Long"
         end if
         if (f1 > s1) and (f0 < s0) and  _
            (heap.value(Isxbelow) > 0) then
            heap.value(Isxbelow) = 0
            trade.shortatmarket 100, "Enter short"
         end if
      else
         if (ff1 > f1) and (ff0 < f0) and _
             trade.openpositionlong then
             trade.longexitatmarket 100, "Exit Long"
         end if
         if (ff1 < f1) and (ff0 > f0) and _
            trade.openpositionshort then
            trade.shortexitatmarket 100, "Exit Short"
         end if
      end if

      ex_vbs_mov3 = trade.currentequity

   end function
```

Formula Language

```
ds_ff := mov3.plot1(1, data1, "Exponential", param1, param2,
param3);
ds_f  := mov3.plot2(1, data1, "Exponential", param1, param2,
param3);
ds_s  := mov3.plot3(1, data1, "Exponential", param1, param2,
param3);
$myCond1 := xabove(ds_ff,ds_s);
$myCond2 := xbelow(ds_ff,ds_s);
$isxabove := if($myCond1 > 0, 1, if($myCond2 > 0, 0,
$isxabove));
$isxbelow := if($myCond1 > 0, 0, if($myCond2 > 0, 1,
$isxbelow));
$myCond3 := xabove(ds_f,ds_s);
$myCond4 := xbelow(ds_f,ds_s);
longatmarket($myCond3 > 0 and $isxabove > 0 and
              openpositionflat > 0, 100, "Enter Long");
shortatmarket($myCond4 > 0 and $isxbelow > 0 and
              openpositionflat > 0, 100, "Enter Short");
longexitatmarket(xbelow(ds_ff, ds_f) > 0 and
                          openpositionlong > 0, 100, "Exit
Long");
shortexitatmarket(xabove(ds_ff, ds_f) > 0 and
                          openpositionshort > 0, 100, "Exit
Short");
plot1 := currentequity;
```

Formula Column

```
mov3.plot3(M3, "Exponential",5,15,30)
```

Moving Avg. Crossover (mov_crossover)

This is a script-based example. You can use script editor to view the script.

Multiplication (mult)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'mult', ['N','N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "mult", Array("N","N"),
Array(""))
```

Formula Language

```
mult(DataN, DataN);
```

Formula Column

N/A

Definition

Multiplication of two series (Link 1 * Link 2).

Examples

Delphi Script

```
function ex_del_mult : double;
var multiple_avg : variant;
begin
   Itself.MakeIndicator('m1', 'mult', ['1', '2'], ['']);
   multiple_avg := Itself.MakeIndicator('mavg', 'average',
                                          ['m1'], [param1.str]);

   if multiple_avg.valid[0] then
      result := multiple_avg.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_mult()
dim multiple_avg
   Itself.MakeIndicator "m1", "mult", Array("1", "2"),
Array("")
```

```
   multiple_avg = Itself.MakeIndicator("mavg", "average", _
                                       Array("m1"),
Array(param1.str))

   if multiple_avg.valid (0) then
      ex_vbs_mult = multiple_avg.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := average(mult(data1, data2), param1);
```

Formula Column

N/A

Narrow Range (nr)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'nr', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "nr", Array("N"), Array("N"))
```

Formula Language

```
nr(dataN, N);
```

Formula Column

```
nr(Tn, N)
```

Definition

Returns 1 for narrowest range within the specified Period and 0 otherwise. Narrow Range requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_nr : double;
Const TradeDays = 0;
var insideday, nr4 : variant;
    myOrderSize : integer;
    TrailStopPt, StopLossPt : double;
    Position2Day : boolean;
begin
   myOrderSize := param1.int;
   TrailStopPt := param2.int*trade.MinTickSize;

   if (itself.TimeFrametype <> ppMin) and
      (itself.TImeFrametype <> ppHour) then
   begin
      Itself.success := false;
      if data1.islastbar then
         ItSelf.alert('High', 'Invalid time frame type',
                              clRed, clBlack, false);
      exit;
   end;
   if heap.size < 1 then
   begin
      Heap.allocate(1);
      Heap.Value[TradeDays] := 0;
   end;

   Itself.CompressSeries ('ds1', '1', ppDaily, 1);
   insideday := Itself.MakeIndicator('isd', 'insidebar',
['$ds1'], ['']);
   nr4 := Itself.MakeIndicator('nrx', 'nr', ['$ds1'], ['4']);

   if (insideday.value[1]>0) and (nr4.value[1]>0) and
      (trade.OpenOrderCount<1) and trade.OpenPositionflat then
   begin
      trade.cancelallopenorders;
      Heap.value[TradeDays] := 0;

      StopLossPt :=
Itself.series('ds1').high[1]+trade.MinTickSize-
                    param3.int*trade.MinTickSize;

trade.LongStop(Itself.series('ds1').high[1]+trade.MinTickSize,
            myOrderSize, otfDay, 'Long previous day high +1');
      trade.LongExitStop(StopLossPt, myOrderSize,
            otfGoodtilCancel, 'Long exit stop loss');

      StopLossPt := Itself.series('ds1').low[1]-
trade.MinTickSize+
                    param3.int*trade.MinTickSize;
      trade.Shortstop(itself.series('ds1').low[1]-
trade.MinTickSize,
            myOrderSize, otfDay, 'Short previous day low -1');
      trade.ShortExitStop(StopLossPt, myOrderSize,
            otfGoodtilCancel, 'Short exit stop loss');
   end;

   if trade.openpositionlong and (trade.OpenPositionPL > 0)
then
      trade.LongExitStop(Data1.close[0]-TrailStopPt,
            myOrderSize, otfFillorKill, 'Long exit take
profit');

   if trade.openpositionshort and (trade.OpenPositionPL > 0)
then
      trade.ShortExitStop(Data1.close[0]+TrailStopPt,
           myOrderSize, otfFillorKill, 'Short exit take
profit');

   if not trade.openpositionflat and
      (ntlib.day(trade.openpositionentrydatetime) <>
       ntlib.day(data1.datetime[0])) and
      (ntlib.day(data1.datetime[1]) <>
ntlib.day(data1.datetime[0])) then
      Heap.inc(TradeDays);

   if not trade.openpositionflat and
```

```
      (NTLib.day(itself.futuredatetime(1)) <>
       NTLib.day(itself.futuredatetime(2))) and
      (Heap.value[TradeDays] > 0) then
   begin
      trade.cancelallopenorders;
      trade.LongExitNextClose(myOrderSize, 'Long exit end of
day 2');
      trade.ShortExitNextClose(myOrderSize, 'Short exit end of
day 2');
   end;

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_nr()
Const TradeDays = 0
dim insideday, nr4
dim myorder

   myOrderSize = param1.int
   TrailStopPt = param2.int*trade.MinTickSize

   if Heap.size < 1 then
      heap.allocate(1)
      heap.value(TradeDays) = 0
   end if

   if (Itself.TimeFrameType <> ppMin) and _
      (Itself.TimeFrameType <> ppHour) then
      ItSelf.success = false
      if data1.islastbar then
         ItSelf.Alert "High", "Invalid time frame type", _
                              clRed, clBlack, false
      end if
      exit function
   end if

   Itself.CompressSeries "ds1", "1", ppDaily, 1
   insideday = Itself.MakeIndicator("isd", "insidebar", _
                     Array("$ds1"), Array(""))
   nr4 = Itself.MakeIndicator("nrx", "nr", Array("$ds1"),
Array("4"))

   if (insideday.value(1)>0) and (nr4.value(1)>0) and _
      (trade.OpenOrderCount<1) and trade.OpenPositionflat then

      Heap.value(TradeDays) = 0
      trade.CancelAllOpenOrders

      StopLossPt =
ItSelf.series("ds1").high(1)+trade.MinTickSize- _
                     param3.int*trade.MinTickSize
      trade.LongStop
ItSelf.series("ds1").high(1)+trade.MinTickSize, _
```

```
            myOrderSize, otfday, "Long previous day high +1"
      trade.LongExitStop StopLossPt, myOrderSize, _
            otfGoodtilCancel, "Long exit stop loss"


      StopLossPt = ItSelf.series("ds1").low(1)-
trade.MinTickSize+ _
                   param3.int*trade.MinTickSize
      trade.ShortStop Itself.Series("ds1").low(1)-
trade.MinTickSize, _
            myOrderSize, otfday, "Short previous day low -1"
      trade.ShortExitStop StopLossPt, myOrderSize, _
            otfGoodtilCancel, "Short exit stop loss"
   end if

   if trade.openpositionlong and (trade.OpenPositionPL>0) then
      trade.LongExitStop Data1.close(0)-TrailStopPt, _
            myOrderSize, otfFillorKill, "Long exit take profit"
   end if

   if trade.openpositionshort and (trade.OpenPositionPL>0) then
      trade.ShortExitStop Data1.close(0)+TrailStopPt, _
            myOrderSize, otfFillorKill, "Short exit take
profit"
   end if

   if not trade.openpositionflat and _
      (NTLib.day(trade.openpositionentrydatetime) <> _
       NTlib.day(data1.datetime(0))) and _
      (NTlib.day(data1.datetime(1)) <>
NTlib.day(data1.datetime(0))) then
      Heap.inc(TradeDays)
   end if

   if not trade.openpositionflat and _
      (ntlib.day(itself.futuredatetime(1)) <> _
       ntlib.day(itself.futuredatetime(2))) and _
       Heap.value(TradeDays) > 0 then
      trade.cancelallopenorders

      trade.longexitnextclose myOrderSize, "Long exit end of
day 2"
      trade.shortexitnextclose myOrderSize, "Short exit end of
day 2"
   end if

   ex_vbs_nr = trade.currentequity
end function
```

Formula Language

```
$myOrderSize      := param1;
$myTrialStopPoint := param2;
$myStoplostpoint  := param3;
$myMinTickSize    := param4;
```

```
'calculate entry prices
$mylongentryprice := prevDHigh(data1)+$myMinTickSize;
$myshortentryprice := prevDLow(data1)-$myMinTickSize;

'calculate stop lost
$mylongstoplost := OpenPositionAverageEntryPrice-
                   ($myStoplostpoint*$myMinTickSize);
$myshortstoplost := OpenPositionAverageEntryPrice+
                   ($myStoplostpoint*$myMinTickSize);

'calculate trial stop
$mylongtrialstop  := OpenPositionAverageEntryPrice+
                    ($myTrialStopPoint*$myMinTickSize);
$myshorttrialstop := OpenPositionAverageEntryPrice-
                    ($myTrialStopPoint*$myMinTickSize);

compressseries(xseries, data1, ppDaily, 1);
$insideday := insidebar(1, xseries);
$nr4 := nr(1, xseries, 4);

longstop(($insideday>0) and ($nr4>0) and (openpositionflat>0),
                               $mylongentryprice,
$myOrderSize);
shortstop(($insideday>0) and ($nr4>0) and (openpositionflat>0),
                               $myshortentryprice,
$myOrderSize);

longexitstop((openpositionlong>0) and
(c>openpositionaverageentryprice),
                               $mylongtrialstop,
openpositionabssize);
shortexitstop((openpositionshort>0) and
(c<openpositionaverageentryprice),
                               $myshorttrialstop,
openpositionabssize);

longexitatmarket((openpositionlong>0) and (c<$mylongstoplost),
                                  openpositionabssize);
shortexitatmarket((openpositionshort>0)
and(c>$myshortstoplost),
                                  openpositionabssize);

plot1 := currentequity;
```

Formula Column

```
nr(m1, 4)
```

Narrow Range N Bar (nrn)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'nrn', ['N'], ['N', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "nrn", Array("N"), Array("N",
"N"))
```

Formula Language

```
nrn(dataN, N, N);
```

Formula Column

```
nrn(Tn, N, N)
```

Definition

Narrow Range returns 1 when the current N bars is the narrowest range over the Period specified by the parameter. Narrow Range requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_nrn : double;
var ex_nrn : variant;
begin
   ex_nrn := itself.makeindicator ('mynrn', 'nrn',
                                   ['1'], [param1.str,
param2.str]);

   if not data1.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   itself.plot[1] := Data1.Volume [0];
   itself.plot[2] := 0;
   if ex_nrn.value [0] > 0 then
     itself.plot[3] := param3.color
   else
     itself.plot[3] := param4.color;
```

```
end;
```

VBScript

```
function ex_vb_nrn()
dim ex_nrn

   ex_nrn = itself.makeindicator ("mynrn", "nrn", _
                                  Array("1"), Array(param1.str,
param2.str))

   if data1.valid(0) = false then
      itself.successall = false
      exit function
   end if

   itself.plot(1) = data1.volume(0)
   itself.plot(2) = 0
   if ex_nrn.value(0) > 0 then
      itself.plot(3) = param3.color
   else
      itself.plot(3) = param4.color
   end if
end function
```

Formula Language

```
plot1 := v;
plot2 := 0;
plot3 := if(nrn(data1, param1, param2) > 0, param3, param4);
```

Formula Column

```
nrn(m1, 7, 20)
```

Now Percent (NowPercent)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'NowPercent', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "NowPercent", Array("N"),
Array(""))
```

Formula Language

```
NOwPercent(dataN);
```

Formula Column

```
NowPercent(Tn)
```

Definition

NowPercent returns in percentage (0 to 100) of the time relative to the trading time range of the hosting function window.

Examples

Delphi Script

```
function ex_del_NowPercent : double;
var ind_np, ind_xover, ind_xunder as variant;
begin

   ind_np := itself.makeindicator('np', 'NowPercent', ['1'],
['']);

   itself.makeindicator('longavg', 'average', ['1'], ['10']);
   itself.makeindicator('shortavg', 'average', ['1'], ['30']);
   ind_xover := itself.makeindicator('xover', 'xabove',
                                        ['shortavg', 'longavg'],
['']);
   ind_xunder := itself.makeindicator('xunder', 'xbelow',
                                        ['shortavg', 'longavg'],
['']);

   if (ind_np.value[0] < 50) and (ind_xover.value[0] > 0) and
      not trade.openpositionlong then
   begin
```

```
      if trade.openpositionshort then
         trade.shortexitatmarket(trade.openpositionabssize,
'Reverse Long');

      trade.longatmarket(param1.int, 'Long entry');
   end;

   if (ind_np.value[0] < 50) and (ind_xunder.value[0] > 0) and
       not trade.openpositionshort then
   begin
      if trade.openpositionlong then
         trade.longexitatmarket(trade.openpositionsize, 'Revers
Short');

      trade.shortatmarket(param1.int, 'Short entry');
   end;

   if (ind_np.value[0] > 50) and not trade.openpositionflat
then
      trade.exitcurrentposition('after half day exit');

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vb_nowpercent()
dim ind_np, ind_xover, ind_xunder

   ind_np = itself.makeindicator("np", "NowPercent",
Array("1"), Array(""))
   itself.makeindicator "shortavg", "average", Array("1"),
Array("10")
   itself.makeindicator "longavg", "average", Array("1"),
Array("30")
   ind_xover = itself.makeindicator("xover", "xabove", _
                                     Array("shortavg",
"longavg"), Array(""))
   ind_xunder = itself.makeindicator("xunder", "xbelow", _
                                     Array("shortavg",
"longavg"), Array(""))

   if (ind_np.value(0) < 50) and (ind_xover.value(0) > 0) and _
             trade.openpositionlong = false then
      if trade.openpositionshort = true then
         trade.shortexitatmarket trade.openpositionabssize,
"Reverse long"
      end if

      trade.longatmarket param1.int, "Long Entry"
   end if

   if (ind_np.value(0) < 50) and (ind_xunder.value(0) > 0) and
_
             trade.openpositionshort = false then
      if trade.openpositionlong = true then
```

```
         trade.longexitatmarket trade.openpositionabssize,
"Reverse short"
      end if

      trade.shortatmarket param1.int, "Short Entry"
   end if

   if (ind_np.value(0) > 50) and trade.openpositionflat = false
then
       trade.exitcurrentposition("Exit positions after half
day")
   end if

   ex_vb_nowpercent = trade.currentequity
end function
```

Formula Language

```
$is_long := (NowPercent(data1) < 50) and
            (xabove(average(data1, 10), average(data1, 30)) >
0);
$is_short := (NowPercent(data1) < 50) and
             (xbelow(average(data1, 10), average(data1,30)) >
0);
shortatmarket($is_long > 0 and openpositionshort > 0, "Reverse
Long");
longatmarket($is_long > 0, "Long entry");
longatmarket($is_short > 0 and openpositionlong > 0, "Reverse
Short");
shortatmarket($is_short > 0, "Short entry");
plot1 := currentequity;
```

Formula Column

```
NowPercent(m1)
```

On Balance Volume (obv)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'obv', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "obv", Array("N"), Array(""))
```

Formula Language

```
obv(dataN);
```

Formula Column

```
obv(Tn)
```

Definition

On Balance Volume is a volume trend indicator. Its main usage is for detecting non-confirmation of price moves. On Balance Volume requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_obv : double;
Const ARRAYSIZE  = 2
      LASTPEAK   = 0
      LASTTROUGH = 1
var ind_obv, ind_hhb, ind_llb : variant;
    i : inetger;
begin

   if heap.size < 1 then
   begin
      heap.allocate(ARRAYSIZE);
      heap.fill(0, ARRAYSIZE-1, 0);
   end;

   ind_obv := itself.makeindicator ('myobv', 'obv', ['1'],
['']);

   ind_hhb := itself.makeindicator ('obv_hhb', 'hhb',
                                    ['myobv'], [param1.str]);
   ind_llb := itself.makeindicator ('obv_llb', 'llb',
                                     ['myobv'], [param1.str]);

   if (ind_hhb.value[1] = 0) and
      (ind_obv.value[2] < ind_obv.value[1]) and
      (ind_obv.value[0] < ind_obv.value[1]) then
   begin
      if (heap.value[LASTPEAK] > 0) then
      begin
         if heap.value[LASTPEAK] < ind_obv.value[1] then
         begin
            i := drawingobjects.add(cotMarker);
            with drawingobjects do
            begin
               AutoRemoveOnUpdateByTick[i] := False;
               Color[i] := param2.color;
               MarkerStyle[i] := msCircle;
               SetPoint(i, 0, data1.DateTime[1],
data1.close[1]);
               Visible[i] := True;
            end;
         end
         else if heap.value[LASTPEAK] > ind_obv.value[1] then
         begin
            i := drawingobjects.add(cotMarker);
            with drawingobjects do
            begin
               AutoRemoveOnUpdateByTick[i] := False;
               Color[i] := param3.color;
               MarkerStyle[i] := msCircle;
               SetPoint(i, 0, data1.DateTime[1],
data1.close[1]);
               Visible[i] := True;
            end;
         end
         else
         begin
            i := drawingobjects.add(cotMarker);
            with drawingobjects do
            begin
               AutoRemoveOnUpdateByTick[i] := False;
               Color[i] := param4.color;
               MarkerStyle[i] := msCircle;
               SetPoint(i, 0, data1.DateTime[1],
data1.close[1]);
               Visible[i] := True;
            end;
         end;
      end;
      heap.value[LASTPEAK] := ind_obv.value[1];
   end;

   if (ind_llb.value[1] = 0) and
      (ind_obv.value[2] > ind_obv.value[1]) and
      (ind_obv.value[0] > ind_obv.value[1]) then
   begin
      if (heap.value[LASTTROUGH] > 0) then
      begin
         if heap.value[LASTTROUGH] < ind_obv.value[1] then
         begin
            i := drawingobjects.add(cotMarker);
            with drawingobjects do
            begin
               AutoRemoveOnUpdateByTick[i] := False;
               Color[i] := param2.color;
               MarkerStyle[i] := msCircle;
               SetPoint(i, 0, data1.DateTime[1],
data1.close[1]);
               Visible[i] := True;
            end;
         end
         else if heap.value[LASTTROUGH] > ind_obv.value[1] then
         begin
            i := drawingobjects.add(cotMarker);
            with drawingobjects do
            begin
               AutoRemoveOnUpdateByTick[i] := False;
               Color[i] := param3.color;
               MarkerStyle[i] := msCircle;
               SetPoint(i, 0, data1.DateTime[1],
data1.close[1]);
               Visible[i] := True;
            end;
         end
         else
         begin
            i := drawingobjects.add(cotMarker);
            with drawingobjects do
            begin
               AutoRemoveOnUpdateByTick[i] := False;
               Color[i] := param4.color;
               MarkerStyle[i] := msCircle;
               SetPoint(i, 0, data1.DateTime[1],
data1.close[1]);
               Visible[i] := True;
            end;
         end;
      end;
      heap.value[LASTTROUGH] := ind_obv.value[1];
   end;

   result := data1.value[0];
end;
```

VBScript

```
function ex_vbs_obv()
Const ARRAYSIZE  = 2
      LASTPEAK   = 0
      LASTTROUGH = 1
dim ind_obv, ind_hhb, ind_lb

   if heap.size < 1 then
```

```
      heap.allocate(ARRAYSIZE)
      heap.fill 0, ARRAYSIZE-1, 0
   end if

   ind_obv = itself.makeindicator("myobv", "obv", Array("1"),
Array(""))
   ind_hhb = itself.makeindicator("myhhb", "hhb", _
                                  Array("myobv"),
Array(param1.str))
   ind_llb = itself.makeindicator("myllb", "llb", _
                                  Array("myobv"),
Array(param1.str))

   if ind_hhb.value(1) = 0 and _
      ind_obv.value(2) < ind_obv.value(1) and _
      ind_obv.value(0) < ind_obv.value(1) then
      if heap.value(LASTPEAK) > 0 then
         if heap.value(LASTPEAK) < ind_obv.value(1) then
            i = drawingobjects.add(cotMarker)
            with drawingobjects
               .AutoRemoveOnUpdateByTick(i) = False
               .Color(i) = params.Items(2).Color
               .MarkerStyle(i) = msCircle
               .SetPoint i, 0, data1.DateTime(1),
data1.value(1)
               .Visible(i) = True
            end with
         elseif heap.value(LASTPEAK) > ind_obv.value(1) then
            i = drawingobjects.add(cotMarker)
            with drawingobjects
               .AutoRemoveOnUpdateByTick(i) = False
               .Color(i) = params.Items(3).Color
               .MarkerStyle(i) = msCircle
               .SetPoint i, 0, data1.DateTime(1),
data1.value(1)
               .Visible(i) = True
            end with
         else
            i = drawingobjects.add(cotMarker)
            with drawingobjects
               .AutoRemoveOnUpdateByTick(i) = False
               .Color(i) = params.Items(4).Color
               .MarkerStyle(i) = msCircle
               .SetPoint i, 0, data1.DateTime(1),
data1.value(1)
               .Visible(i) = True
            end with
         end if
      end if
      heap.value(LASTPEAK) = ind_obv.value(1)
   end if

   if ind_llb.value(1) = 0 and _
      ind_obv.value(2) > ind_obv.value(1) and _
      ind_obv.value(0) > ind_obv.value(1) then
      if heap.value(LASTTROUGH) > 0 then
```

```
         if heap.value(LASTROUGH) < ind_obv.value(1) then
            i = drawingobjects.add(cotMarker)
            with drawingobjects
               .AutoRemoveOnUpdateByTick(i) = False
               .Color(i) = params.Items(2).Color
               .MarkerStyle(i) = msCircle
               .SetPoint i, 0, data1.DateTime(1),
data1.value(1)
               .Visible(i) = True
            end with
         elseif heap.value(LASTTROUGH) > ind_obv.value(1) then
            i = drawingobjects.add(cotMarker)
            with drawingobjects
               .AutoRemoveOnUpdateByTick(i) = False
               .Color(i) = params.Items(3).Color
               .MarkerStyle(i) = msCircle
               .SetPoint i, 0, data1.DateTime(1),
data1.value(1)
               .Visible(i) = True
            end with
         else
            i = drawingobjects.add(cotMarker)
            with drawingobjects
               .AutoRemoveOnUpdateByTick(i) = False
               .Color(i) = params.Items(4).Color
               .MarkerStyle(i) = msCircle
               .SetPoint i, 0, data1.DateTime(1),
data1.value(1)
               .Visible(i) = True
            end with
         end if
      end if
      heap.value(LASTTROUGH) = ind_obv.value(1)
   end if

   ex_vbs_obv = Data1.Value (0)
end function
```

Formula Language

```
$ind_hhb1 := hhb(1, obv(data1), param1);
$ind_llb1 := llb(1, obv(data1), param1);

$ispeak := $ind_hhb1 = 0 and obv(2,data1) < obv(1,data1) and
                             obv(data1) < obv(1,data1);
$istrough := $ind_llb1 = 0 and obv(2,data1) > obv(1,data1) and
                               obv(data1) > obv(1,data1);

$risingpeak    := ($ispeak > 0) and ($lastpeak > 0) and
                                (obv(1,data1) > $lastpeak);
$fallingpeak   := ($ispeak > 0) and ($lastpeak > 0) and
                              (obv(1,data1) < $lastpeak);
$risingtrough  := ($istrough > 0) and ($lasttrough > 0) and
                              (obv(1,data1) > $lasttrough);
$fallingtrough := ($istrough > 0) and ($lasttrough > 0) and
                              (obv(1,data1) < $lasttrough);
```

```
$lastpeak := if($ind_hhb1 = 0, obv(1, data1), $lastpeak);
$lasttrough := if($ind_llb1 = 0, obv(1, data1), $lasttrough);

plot1 := h;
plot2 := l;
plot3 := if($risingpeak > 0 or $risingtrough > 0, param2,
             if($fallingpeak > 0 or $fallingtrough > 0, param3,
param4));

success1 := if($ispeak > 0 or istrough > 0, 1, 0);
success2 := if($ispeak > 0 or istrough > 0, 1, 0);
success3 := if($ispeak > 0 or istrough > 0, 1, 0);
```

Formula Column

```
obv(m1)
```

Open Interest (openint)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'openint', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "openint", Array("N"),
Array(""))
```

Formula Language

```
openint(dataN);
```

Formula Column

```
openint(Tn)
```

Definition

Returns the open interest of Link 1. The value is undefined if Link 1 does not contain any open interest information, e.g. Link 1 is a stock.

Examples

Delphi Script

```
function ex_del_openint : double;
var ind_oi, ind_roc_oi, ind_avg_roc_oi : variant;
begin
   ind_oi := itself.makeindicator ('myoi', 'openint', ['1'],
['']);
   ind_roc_oi := itself.makeindicator('myroc', 'roc', ['myoi'],
['1']);
   ind_avg_roc_oi := itself.makeindicator('myavg', 'average',
                                         ['myroc'],
[param1.str]);


   if not (ind_oi.valid[0] and ind_roc_oi.valid[0]) then
   begin
      itself.successall := false;
      exit;
   end;

   itself.plot[1] := ind_oi.value[0];
```

```
      itself.plot[2] := 0;

      if ind_roc_oi.value[0] > ind_avg_roc_oi.value[0] then
         itself.plot[3] := params.items[2].color
      else
         itself.plot[3] := params.items[3].color;

   end;
```

VBScript

```
   function ex_vbs_openint()
   dim ind_oi, ind_roc_oi, ind_avg_roc_oi

      ind_oi = itself.makeindicator("myoi", "openint", Array("1"),
   Array(""))
      ind_roc_oi = itself.makeindicator("myroc", "roc", _
                                        Array("myoi"),
   Array("1"))
      ind_avg_roc_oi = itself.makeindicator("myavg", "average", _
                                        Array("myroc"),
   Array(param1.str))

      if ind_oi.valid(0) = false or ind_roc_oi.valid(0) = false
   then
         itself.successall = false
         exit function
      end if

      itself.plot(1) = ind_oi.value(0)
      itself.plot(2) = 0

      if ind_roc_oi.value(0) > ind_avg_roc_oi.value(0) then
         itself.plot(3) = params.items(2).color
      else
         itself.plot(3) = params.items(3).color
      end if

   end function
```

Formula Language

```
   plot1 := openint(data1);
   plot2 := 0;
   plot3 := if(roc(openint(data1),1) >
            average(roc(openint(data1),1),param1), param2,
   param3);
```

Formula Column

```
   openint(m5)
```

Optimization Demo (optdemo)

The indicator is part of the tutorial on optimization. It basically provides two moving averages as signals.

Optimization Demo System (optdemosys)

The indicator is part of the tutorial on optimization. It takes signals from another indicator, performs a crossover check, and sends out long/short orders.

Outside Bar (outsidebar)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'outsidebar', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "outsidebar", Array("N"),
Array(""))
```

Formula Language

```
outsidebar(dataN);
```

Formula Column

```
outsidebar(Tn)
```

Definition

Outside Bar returns 1 when the current bar is an outside bar. Outside Bar requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_outsidebar : double;
var ind_osb, ind_avg_vol : variant;
begin

   ind_osb := itself.makeindicator('myosb', 'outsidebar',
['1'], ['']);
   ind_avg_vol := itself.makeindicator('myvol', 'average',
                                       ['1.v'], [param1.str]);

   if ind_osb.value [0] < 1 then
   begin
      itself.successall := false;
      exit;
   end;

   if (ind_osb.value [0] > 0) and
      (data1.volume[1] > ind_avg_vol.value[1]) and
      (data1.volume[0] < ind_avg_vol.value[0]) then
   begin
```

```
      itself.plot[1] := Data1.high [0];
      itself.plot[2] := data1.low [0];
      itself.plot[3] := param2.color;
   end
   else
     itself.successall := false;
end;
```

VBScript

```
function ex_vbs_outsidebar ()
dim ind_osb, ind_avg_vol

   ind_osb = itself.makeindicator("myosb", "outsidebar",
Array("1"), Array(""))
   ind_avg_vol = itself.makeindicator("myvol", "average", _
                                      Array("1.v"),
Array(param1.str))

   if ind_osb.value(0) < 1 then
      itself.successall = false
      exit function
   end if

   if ind_osb.value(0) > 0 and _
      data1.volume(1) > ind_avg_vol.value(1) and _
      data1.volume(0) < ind_avg_vol.value(0) then
      itself.plot(1) = Data1.high (0)
      itself.plot(2) = data1.low(0)
      itself.plot(3) = param2.color
   else
      itself.successall = false
   end if
end function
```

Formula Language

```
$isdowntrend := average(1, v, param1) < v(1) and
               average(v, param1) > v and outsidebar(data1) >
0;

plot1 := h;
plot2 := l;
plot3 := if($isdowntrend>0, param2, 0);

success1 := if($isdowntrend<1 or outsidebar(data1)>0, 1, 0);
success2 := if($isdowntrend<1 or outsidebar(data1)>0, 1, 0);
success3 := if($isdowntrend<1 or outsidebar(data1)>0, 1, 0);
```

Formula Column

```
outsidebar(m5)
```

Parabolic SAR (parabolic)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'parabolic', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "parabolic", Array("N"),
Array("N"))
```

Formula Language

```
parabolic(dataN, N);
```

Formula Column

```
parabolic(Tn, N)
```

Definition

Should always plotted directly onto the prices. Parabolic SAR is a stop and reverse (SAR) model based on the concept that in the beginning of a trend the fluctuation tends to be high and requiring larger tolerance, while in a mature trend, tighter stop is preferred because reversal may happen any time. Parabolic SAR takes one parameter AFactor and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_parabolic : double;
var ind_xover, ind_xunder, ind_sar : variant;
begin

   itself.makeindicator('slow_avg', 'average', ['1'], ['30']);
   itself.makeindicator('fast_avg', 'average', ['1'], ['10']);

   ind_xover := itself.makeindicator('xover', 'xabove',
                                       ['fast_avg',
'slow_avg'], ['']);
   ind_xunder := itself.makeindicator('xunder', 'xbelow',
                                       ['fast_avg',
'slow_avg'], ['']);
   ind_sar := itself.makeindicator('sar', 'parabolic', ['1'],
['0.02']);
```

```
   if trade.openpositionflat then
   begin
      if (ind_xover.value[0] > 0) and (ind_sar.value[0] <
data1.low[0]) then
         trade.longatmarket(param1.int, 'xover long');

      if (ind_xunder.value[0] > 0) and (ind_sar.value[0] >
data1.high[0]) then
         trade.shortatmarket(param1.int, 'xunder short');
   end
   else if trade.openpositionlong then
   begin
      if (ind_sar.value[0] > data1.high[0]) then
         trade.longexitatmarket(trade.openpositionabssize, 'SAR
exit long');
   end
   else if trade.openpositionshort then
   begin
      if (ind_sar.value[0] < data1.low[0]) then
         trade.shortexitatmarket(trade.openpositionabssize,
'SAR exit shart');
   end;

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_parabolic()
dim ind_xover, in_xunder, ind_sar

   itself.makeindicator "slow_avg", "average", Array("1"),
Array("30")
   itself.makeindicator "fast_avg", "average", Array("1"),
Array("10")

   ind_xover = itself.makeindicator("xover", "xabove", _
                   Array("fast_avg", "slow_avg"), Array(""))
   ind_xunder = itself.makeindicator("xunder", "xbelow", _
                   Array("fast_avg", "slow_avg"), Array(""))
   ind_sar = itself.makeindicator("sar", "parabolic", _
                   Array("1"), Array("0.02"))

   if trade.openpositionflat = true then
      if (ind_xover.value(0)>0) and _
         (ind_sar.value(0)<data1.low(0)) then
         trade.longatmarket param1.int, "xover long"
      end if

      if (ind_xunder.value(0)>0) and _
         (ind_sar.value(0)>data1.high(0)) then
         trade.shortatmarket param1.int, "xunder short"
      end if
   elseif trade.openpositionlong = true then
      if ind_sar.value(0)>data1.high(0) then
```

```
         trade.longexitatmarket trade.openpositionabssize, "SAR
exit long"
      end if
   elseif trade.openpositionshort = true then
      if ind_sar.value(0)<data1.low(0) then
         trade.shortexitatmarket trade.openpositionabssize,
"SAR exit short"
      end if
   end if

   ex_vbs_parabolic = trade.currentequity
end function
```

Formula Language

```
$ind_xover := xabove(average(data1, 10), average(data1, 30));
$ind_xunder := xbelow(average(data1, 10), average(data1, 30));

longatmarket((openpositionflat>0) and ($ind_xover>0) and
             (parabolic(data1, 0.02)<low(0)), param1, "xover
long");
shortatmarket((openpositionflat>0) and ($ind_xunder>0) and
               (parabolic(data1, 0.02)>high(0)), param1,
"xunder short");
longexitatmarket((openpositionlong>0) and
                 (parabolic(data1, 0.02)>high(0)), param1, "SAR
long exit");
shortexitatmarket((openpositionshort>0) and
                  (parabolic(data1, 0.02)<low(0)), param1, "SAR
short exit");

plot1 := currentequity;
```

Formula Column

```
parabolic(m5, 0.02)
```

Parabolic SAR EX (parabolicex)

Definition

Should always plotted directly onto the prices. Parabolic SAR EX is a stop and reverse (SAR) model based on the concept that in the beginning of a trend the fluctuation tends to be high and requiring larger tolerance, while in a mature trend, tighter stop is preferred because reversal may happen any time. Parabolic SAR EX takes two parameters Factor and Max and requires one series (Link 1).

Parabolic SAR EX is the extended version of ***Parabolic SAR (parabolic)*** (on page 1997). The EX version lets you adjust the Max parameter. If Max parameter is set to the classic value of 0.2, Parabolic SAR EX returns the same value as Parablic SAR.

See ***Parabolic SAR (parabolic)*** (on page 1997) for usage examples.

Percentage Bands 3 Lines (percentageband3)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'percentageband3', ['N'],
['string','N','N']);
```

VBScript

```
Itself.MakeIndicator("string", "percentageband3", Array("N"),
Array("string","N","N"))
```

Formula Language

```
percentageband3(dataN,"string",N,N);
```

Formula Column

```
percnetageband3(Tn,"String",N,N)
```

Definition

Percentage Bands 3 Lines. The specific moving average and the percentage offsets are plotted. This indicator requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_percentageband3 : double;
var pb3 : variant;
begin

   pb3 := itself.makeindicator('pb3', 'percentageband3', ['1'],
                               ['Exponential', '20', '15']);

   if not pb3.valid[0] then
   begin
      itself.success := false;
      exit;
   end;

   if (data1.close[0] > pb3.valueex[1, 0]) and
      (data1.close[0] < pb3.valueex[3, 0]) then
      result := pb3.valueex[3, 0]
   else if (data1.close[0] < pb3.valueex[1, 0]) and
```

```
           (data1.close[0] > pb3.valueex[2, 0]) then
      result := pb3.valueex[2, 0]
   else
      result := pb3.valueex[1, 0];

end;
```

VBScript

```
function ex_vbs_percentageband3()
dim pb3

   pb3 = itself.makeindicator("pb3", "percentageband3",
Array("1"), _
                                     Array("exponential", "20",
"15"))

   if pb3.valid(0) = false then
      itself.success = false
      exit function
   end if

   if (data1.close(0)>pb3.valueex(1, 0)) and _
      (data1.close(0)<pb3.valueex(3, 0)) then
      ex_vbs_percentageband3 = pb3.Valueex (3, 0)
   elseif (data1.close(0)<pb3.valueex(1, 0)) and _
          (data1.close(0)>pb3.valueex(2, 0)) then
      ex_vbs_percentageband3 = pb3.Valueex (2, 0)
   else
      ex_vbs_percentageband3 = pb3.Valueex (1, 0)
   end if
end function
```

Formula Language

```
$center := percentageband3.p1(data1, "exponential", "20",
"15");
$below  := percentageband3.p2(data1, "exponential", "20",
"15");
$above  := percentageband3.p3(data1, "exponential", "20",
"15");

plot1 := if((c>$center)and(c<$above),$above,
         if((c<$center)and(c>$below),$below,$center));
```

Formula Column

```
percentageband3.p2(m5, "Exponential", 20, 1.4)
```

Plot Count (plotcount)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'plotcount', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "plotcount", Array("N"),
Array(""))
```

Formula Language

N/A

Formula Column

N/A

Definition

Plot Count returns the number of plots in the input. Plot Count requires one link (Link 1).

Examples

Delphi Script

```
function ex_del_plotcount : double;
var ind_bbands3, pc : variant;
begin

   ind_bbands3 := itself.makeindicator('mybb3', 'bbands3',
                                                   ['1'],
['10','1']);
   pc := itself.makeindicator('pc', 'plotcount', ['mybb3'],
['']);

   result := ind_bbands3.valueex[pc.value[0], 0];
end;
```

VBScript

```
function ex_vbs_plotcount()
dim pc
dim ind_bbands3
```

```
   ind_bbands3 = itself.makeindicator("mybb3", "bbands3", 
Array("1"), _
                                    Array("10", "1"))
   pc = itself.makeindicator("pc", "plotcount", Array("mybb3"), 
Array(""))

   if ind_bbands3.valid(0) = false then
      itself.success = false
      exit function
   end if

   ex_vbs_plotcount = ind_bbands3.valueex(pc.value(0), 0)
end function
```

Formula Language

N/A

Formula Column

N/A

Plot Value (plotvalue)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'plotvalue', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "plotvalue", Array("N"),
Array("N"))
```

Formula Language

N/A

Formula Column

N/A

Definition

Given the plot number. Plot Value returns the value of a specific plot. Plot Value requires one link (Link 1).

Examples

Delphi Script

```
function ex_del_plotvalue : double;
var ind_pv : variant;
begin
   itself.makeindicator('mymacd', 'qc_macd', ['1'],
                                  ['12', '26', 'No', '9']);
   ind_pv := itself.makeindicator('pv', 'plotvalue',
['mymacd'], ['3']);

   if not (ind_pv.valid[0] and data1.valid[0]) then
   begin
      itself.successall := false;
      exit;
   end;

   itself.plot[1] := ind_pv.value[0];
   itself.plot[2] := 0;
```

```
      if ind_pv.value[0] > 0 then
         itself.plot[3] := clgreen
      else
         itself.plot[3] := clred;
   end;
```

VBScript

```
   function ex_vbs_plotvalue()
   dim ind_pv

      itself.makeindicator "mymacd", "qc_macd", Array("1"), _
                                     Array("12", "26", "No", "9")
      ind_pv = itself.makeindicator("pv", "plotvalue", _
                                     Array("mymacd"), Array("3"))

      if data1.valid(0) = false or ind_pv.valid(0) = false then
         itself.successall = false
         exit function
      end if

      itself.plot(1) = ind_pv.value(0)
      itself.plot(2) = 0

      if ind_pv.value(0) > 0 then
         itself.plot(3) = clgreen
      else
         itself.plot(3) = clred
      end if
   end function
```

Formula Language

N/A

Formula Column

N/A

Previous Day Average (prevDAverage)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevDAverage', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevDAverage", Array("N"),
Array(""))
```

Formula Language

```
prevDAverage(dataN)
```

Formula Column

```
prevDAverage(Tn)
```

Definition

Previous Day Average is a day trading support resistance indicator that marks the average of high, low and close of previous trading day with horizontal dots. Previous Day Average works on series with minute or tick time frame. It requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevdaverage : double;
var pva : variant;
begin

   if heap.size = 0 then
   begin
      heap.allocate(1);
      heap.value[0] := 0;
   end;

   pva := itself.makeindicator('pva', 'prevDAverage', ['1'],
['']);

   if data1.close[0] > pva.value[0] then
      heap.inc(0);

   result := heap.value(0);
end;
```

VBScript

```
function ex_vbs_prevdaverage()
dim pva

   if heap.size = 0 then
      heap.allocate(1)
      heap.value(0) = 0
   end if

   pva = itself.makeindicator("pva", "PrevDAverage",
Array("1"), Array(""))

   if data1.close(0) > pva.value(0) then
      heap.inc(0)
   end if

   ex_vbs_prevdaverage = heap.value(0)
end function
```

Formula Language

```
$hcounter := if(c>prevDaverage(data1), $hcounter+1, $hcounter);
plot1 := $hcounter;
```

Formula Column

```
prevDAverage(m1)
```

Previous Day Close (prevDClose)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevDClose', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevDClose", Array("N"),
Array(""))
```

Formula Language

```
prevDClose(dataN)
```

Formula Column

```
prevDClose(Tn)
```

Definition

Previous Day Close draws a series of horizontal dots marking the closing price level of the previous trading day. It works on series with minute or tick time frame. It requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevdclose : double;
var pvc : variant;
begin

   if heap.size = 0 then
   begin
      heap.allocate(1);
      heap.value[0] := 0;
   end;

   pvc := itself.makeindicator('pvc', 'prevDClose', ['1'],
['']);

   if data1.close[0] > pvc.value[0] then
      heap.inc(0)
   else if data1.close[0] < pvc.value[0] then
      heap.dec(0);
   result := heap.value[0];
end;
```

VBScript

```
function ex_vbs_prevdclose()
dim pvc

   if heap.size = 0 then
      heap.allocate(1)
      heap.value(0) = 0
   end if

   pvc = itself.makeindicator("pvc", "prevDClose", Array("1"),
Array(""))

   if data1.close(0) > pvc.value(0) then
      heap.inc(0)
   elseif data1.close(0) < pvc.value(0) then
      heap.dec(0)
   end if

   ex_vbs_prevdclose = heap.value(0)
end function
```

Formula Language

```
$tcounter := if(c>prevDClose(data1), $tcounter+1,
                if(c<prevDclose(data1), $tcounter-1,
$tcoutner));
plot1 := $tcounter;
```

Formula Column

```
prevDClose(m1)
```

Previous Day High (prevDHigh)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevDHigh', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevDHigh", Array("N"),
Array(""))
```

Formula Language

```
prevDHigh(dataN)
```

Formula Column

```
prevDHigh(Tn)
```

Definition

Previous Day High draws a series of horizontal dots marking the high price level of the previous trading day. It works on series with minute or tick time frame. It requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevDHigh  : double;
var pvh : variant;
    i, vol_period : integer;
begin

   vol_period := param1.int;

   if heap.size = 0 then
   begin
      heap.allocate(1);

      heap.allocatereallist('vol_array', vol_period);
      for i := 0 to vol_period-1 do
      begin
         heap.reallist['vol_array', i] := 0;
      end;
   end;
   pvh := itself.makeindicator('pvh', 'prevDHigh', ['1'],
['']);

   if data1.high[0] > pvh.value[0] then
      heap.listpush('vol_array', data1.volume[0]);

   if data1.barsnum[0] > vol_period then
      result := heap.listsum('vol_array')
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_prevdhigh()
dim pvh

   vol_period = param1.int

   if heap.size = 0 then
      heap.allocate(1)

      heap.allocatereallist "vol_array", vol_period
      for i = 0 to vol_period-1
         heap.reallist("vol_array", i) = 0
      next
   end if

   pvh = itself.makeindicator("pvh", "prevDHigh", Array("1"),
Array(""))

   if data1.valid(0) = false then
      itself.success = false
      exit function
   end if

   if data1.high(0) > pvh.value(0) then
      heap.listpush "vol_array", data1.volume(0)
   end if

   ex_vbs_prevdhigh = heap.listsum("vol_array")
end function
```

Formula Language

```
$vol_sum := if(h>prevDhigh(data1), $vol_sum+v, $vol_sum);
plot1 := $vol_sum;
```

Formula Column

```
prevDHigh(m1)
```

Previous Day Low (prevDLow)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevDLow', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevDLow", Array("N"),
Array(""))
```

Formula Language

```
prevDLow(dataN)
```

Formula Column

```
prevDLow(Tn)
```

Definition

Previous Day Low draws a series of horizontal dots marking the low price level of the previous trading day. It works on series with minute or tick time frame. It requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevdlow  : double;
var pvl : variant;
    i, vol_period : integer;
begin

   vol_period := param1.int;

   if heap.size = 0 then
   begin
      heap.allocate(1);

      heap.allocatereallist('myvol', vol_period);
      for i := 0 to vol_period-1 do
      begin
         heap.reallist['myvol', i] := 0;
      end;
   end;
```

```
      pvl := itself.makeindicator('pvl', 'prevdlow', ['1'], ['']);

      if not (pvl.valid[0] and data1.valid[0]) then
      begin
         itself.success := false;
         exit;
      end;

      if data1.low[0] < pvl.value[0] then
         heap.listpush('myvol', data1.volume[0]);

      if data1.barsnum[0] > vol_period then
         result := heap.listsum('myvol')
      else
         itself.success := false;
   end;
```

VBScript

```
   function ex_vbs_prevdlow()
   dim pvl
      vol_period = param1.int
      if heap.size = 0 then
         heap.allocate(1)

         heap.allocatereallist "myvol", vol_period
         for i = 0 to vol_period-1
            heap.reallist("myvol", i) = 0
         next
      end if

      pvl = itself.makeindicator("pvl", "prevdlow", Array("1"),
   Array(""))

      if data1.valid(0) = false or pvl.valid(0) = false then
         itself.success = false
         exit function
      end if

      if data1.low(0) < pvl.value(0) then
         heap.listpush "myvol", data1.volume(0)
      end if

      if data1.barsnum(0) > vol_period then
         ex_vbs_prevdlow = heap.listsum("myvol")
      else
         itself.success = false
      end if
   end function
```

Formula Language

```
   $vol_total := $vol_total + if(l<prevDlow(data1), v, 0);
   plot1 := $vol_total;
```

Formula Column

```
prevDLow(m1)
```

Previous Day Open (prevDOpen)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevDOpen', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevDOpen", Array("N"),
Array(""))
```

Formula Language

```
prevDOpen(dataN)
```

Formula Column

```
prevDOpen(Tn)
```

Definition

Given a daily or intraday chart, this indicator draws a horizontal line marking the previous day opening price level and automatically adjust itself when the day has changed. This is a support/resistance indicators. Previous Day Open requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevdopen : double;
var pvo : variant;
    typical_price : double;
    myperiod, i : integer;
begin

   myperiod := param1.int;

   if heap.size = 0 then
   begin
      heap.allocate(1);

      heap.allocatereallist('myprice', myperiod);
      for i := 0 to myperiod-1 do
      begin
         heap.reallist['myprice', i] := 0;
      end;
   end;
   pvo := itself.makeindicator('pvo', 'prevDOpen', ['1'],
['']);

   if not (pvo.valid[0] and data1.valid[0]) then
   begin
      itself.success := false;
      exit;
   end;

   typical_price :=
(data1.close[0]+data1.high[0]+data1.low[0])/3;

   if typical_price > pvo.value[0] then
      heap.listpush('myprice', typical_price);

   if data1.barsnum[0] > myperiod then
      result := heap.listsum('myprice')/myperiod
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_prevdopen()
dim pvo

   myperiod = param1.int

   if heap.size = 0 then
      heap.allocate(1)

      heap.allocatereallist "myprice", myperiod
      for i = 0 to myperiod-1
         heap.reallist("myprice", i) = 0
      next
   end if

   pvo = itself.makeindicator("pvo", "pervDOpen", Array("1"),
Array(""))

   if not data1.valid(0) then
      itself.success = false
      exit function
   end if

   typical_price =
(data1.high(0)+data1.low(0)+data1.close(0))/3

   if typical_price > pvo.value(0) then
      heap.listpush "myprice", typical_price
   end if

   if data1.barsnum(0) > myperiod then
      ex_vbs_prevdopen = heap.listsum("myprice")/myperiod
   else
```

```
      itself.success = false
   end if
end function
```

Formula Language

```
$typical_price := (h+l+c)/3;
$sum_price := $sum_price + if($typical_price>prevDOpen(data1),
$typical_price, 0);
$mycounter := $mycounter + if($typical_price>prevDOpen(data1),
1, 0);
plot1 := if($mycounter>0, $sum_price/$mycounter, 0);
```

Formula Column

```
prevDOpen(m15)
```

Previous Day Resistance (prevDResistance)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevDResistance', ['N'],
['string']);
```

VBScript

```
Itself.MakeIndicator("string", "prevDResistance", Array("N"),
Array("string"))
```

Formula Language

```
prevDResistance(dataN, "string")
```

Formula Column

```
prevDResistance(Tn, "string")
```

Definition

Previous Day Resistance is a day trading technique called support/resistance based on the previous trading day's traded range. The Previous Day Resistance calculates and plots two resistance levels. It is usually used with Previous Day Average and Previous Day Support. Previous Day Resistance works on series with minute or tick time frame. It takes one parameter Type, which is either Upper or Lower. Previous Day Resistance requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevdresistance  : double;
var pvr : variant;
    mydiff : double;
begin

   pvr := itself.makeindicator('pvr', 'prevDResistance', ['1'],
['Upper']);

   if not (pvr.valid[0] and data1.valid[0]) then
   begin
      itself.successall := false;
      exit;
   end;
```

```
    mydiff := pvr.value[0]-data1.value[0];
    itself.plot[1] := mydiff;
    itself.plot[2] := 0;
    if mydiff > 0 then
       itself.plot[3] := clgreen
    else
       itself.plot[3] := clred;
end;
```

VBScript

```
function ex_vbs_prevdresistance()
dim pvr

    pvr = itself.makeindicator("pvr", "prevdresistance", _
                               Array("1"), Array("Upper"))

    if not (pvr.valid (0) and data1.valid (0)) then
       itself.successall = false
       exit function
    end if

    mydiff = pvr.value (0)-Data1.Value (0)
    itself.plot(1) = mydiff
    itself.plot(2) = 0
    if mydiff > 0 then
       itself.plot(3) = clgreen
    else
       itself.plot(3) = clred
    end if
end function
```

Formula Language

```
$mydiff := prevdresistance(data1,"Upper")-data1;
plot1 := $mydiff;
plot2 := 0;
plot3 := if($mydiff>0, clgreen, clred);
```

Formula Column

```
prevDResistance(m15, "Upper")
```

Previous Day Support (prevDSupport)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevDSupport', ['N'],
['string']);
```

VBScript

```
Itself.MakeIndicator("string", "prevDSupport", Array("N"),
Array("string"))
```

Formula Language

```
prevDSupport(dataN, "string")
```

Formula Column

```
prevDSupport(Tn, "string")
```

Definition

Previous Day Support is a day trading technique called support/resistance based on the previous trading day's traded range. The Previous Day Support calculates and plots the two support levels. It is usually used with Previous Day Average and Previous Day Resistance. Previous Day Support works on series with minute or tick time frame. It takes one parameter Type, which is either Upper or Lower. Previous Day Support requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevdsupport : double;
var pvs, pvr : variant;
    mysize : integer;
begin
   mysize := param1.int;

   pvs := itself.makeindicator('pvs', 'prevDSupport', ['1'],
['Upper']);
   pvr := itself.makeindicator('pvr', 'prevDResistance', ['1'],
['Upper']);

   if (data1.value[0]-pvs.value[0]) < 0.05 then
      if trade.openpositionflat and (trade.openordercount = 0)
```

```
then
         trade.longstop((pvs.value[0]+0.1), mysize,
                         otfgoodtilcancel, 'buy need support');

   if (trade.openordercount > 0) and (data1.value[0] <
pvs.value[0]) then
      trade.cancelallbuyorders;

   if (pvr.value[0]-data1.value[0]) < 0.05 then
      trade.longexitatmarket( mysize, 'exit need resistance');

   if ((pvs.value[0]-data1.value[0]) > 0.1) and
trade.openpositionlong then
      trade.longexitstop((pvs.value[0]-0.5), mysize,
                         otffillorkill, 'stop lost support
break');
   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_prevdsupport()
dim pvs, pvr

   mysize = param1.int

   pvs = itself.makeindicator("pvs", "prevDSupport", _
                              Array("1"), Array("Upper"))
   pvr = itself.makeindicator("pvr", "prevDResistance", _
                              Array("1"), Array("Upper"))

   if (data1.value(0)-pvs.value(0)) < 0.05 then
      if trade.openpositionflat = true and trade.openordercount
= 0 then
         trade.longstop (pvs.value(0)+0.1), mysize, _
                         otfgoodtilcancel, "entry need support"
      end if
   end if

   if (trade.openordercount > 0) and (data1.value(0) <
pvs.value(0)) then
      trade.cancelallbuyorders
   end if

   if (pvr.value(0)-data1.value(0)) < 0.05 then
      trade.longexitatmarket mysize, "exit need resistance"
   end if

   if (pvs.value(0)-data1.value(0)) > 0.1 and _
      (trade.openpositionlong = true) then
      trade.longexitstop (pvs.value(0)-0.5), mysize, _
                          otffillorkill, "stop lost support
break"
   end if

   ex_vbs_prevdsupport = trade.currentequity
```

```
end function
```

Formula Language

```
$mysize := param1;
$pvs := prevDSupport(data1, "Upper");
$pvr := prevDResistance(data1, "Upper");
longstop((data1-$pvs)<0.05, $pvs+0.1, $mysize, "entry need
support");
longexitatmarket(($pvr-data1)<0.05, $mysize, "exit need
resistance");
longexitstop(($pvs-data1)>0.1, $pvs-0.5, $mysize, "stop lost
support break");
plot1 := currentequity;
```

Formula Column

```
prevDSupport(m15, "Upper")
```

Previous Month Average (prevMAverage)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevMAverage', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevMAerage", Array("N"),
Array(""))
```

Formula Language

```
prevMAverage(dataN)
```

Formula Column

```
prevMAverage(Tn)
```

Definition

Previous Month Average is a support resistance indicator that marks the average of high, low and close of previous month with a dotted line and can automatically "jump" to the right level when the data switch over to the next month. Previous month average requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevmaverage : double;
var pma, ma : variant;
begin
   pma := itself.makeindicator('pma', 'prevmaverage', ['1']
,['']);
   ma  := itself.makeindicator('ma', 'average', ['1'],
[param1.str]);

   if not (pma.valid[0] and ma.valid[0]) then
   begin
      itself.success := false;
      exit;
   end;

   result := ma.value[0]-pma.value[0];
end;
```

VBScript

```
function ex_vbs_prevmaverage()
dim pma, ma
   pma = itself.makeindicator("pma", "prevmaverage",
Array("1"), Array(""))
   ma  = itself.makeindicator("ma", "average", Array("1"),
Array(param1.str))

   if not (pma.valid(0) and ma.valid(0)) then
      itself.success = false
      exit function
   end if
   ex_vbs_prevmaverage = ma.value(0)-pma.value(0)
end function
```

Formula Language

```
plot1 := average(data1,param1)-prevmaverage(data1);
```

Formula Column

```
prevMAverage(m15)
```

Previous Month Close (prevMClose)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevMClose', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevMClose", Array("N"),
Array(""))
```

Formula Language

```
prevMClose(dataN)
```

Formula Column

```
prevMClose(Tn)
```

Definition

Given a daily or weekly chart, this indicator draws a series of horizontal dots marking the previous month closing price level and automatically adjust itself when the month has changed. One of the best support/resistance indicators available. Usually the complete group of Previous Month indicators are used together on the same chart. Previous Month Close requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevmclose : double;
var pmc : variant;
    price_diff : double;
begin

   pmc := itself.makeindicator('pmc', 'prevMClose', ['1'],
['']);

   price_diff := data1.value[0]-pmc.value[0];

   itself.plot[1] := price_diff;
   itself.plot[2] := 0;

   if price_diff > 0 then
      itself.plot[3] := clgreen
   else
      itself.plot[3] := clred;
end;
```

VBScript

```
function ex_vbs_prevmclose()
dim pmc

   pmc = itself.makeindicator("pmc", "prevMClose", Array("1"),
Array(""))

   if not (data1.valid(0) and pmc.valid(0)) then
      itself.successall = false
      exit function
   end if

   price_diff = data1.value(0)-pmc.value(0)

   itself.plot(1) = price_diff
   itself.plot(2) = 0
   if price_diff>0 then
      itself.plot(3) = clgreen
   else
      itself.plot(3) = clred
   end if
end function
```

Formula Language

```
$price_diff := data1-prevmclose(data1);
plot1 := $price_diff;
plot2 := 0;
plot3 := if($price_diff>0, clgreen, clred);
```

Formula Column

```
prevMClose(m15)
```

Previous Month High (prevMHigh)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevMHigh', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevMHigh", Array("N"),
Array(""))
```

Formula Language

```
prevMHigh(dataN)
```

Formula Column

```
prevMHigh(Tn)
```

Definition

Similar to Previous Month Close, this indicator plots the previous month high in a series of dots on daily and weekly charts. Previous Month High requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevmhigh : double;
var pmh, myhhv : varaint;
begin

   pmh := itself.makeindicator('pmh', 'prevmhigh', ['1'],
['']);
   myhhv := itself.makeindicator('myhhv', 'hhv', ['1.h'],
['10']);

   if not (myhhv.valid[0] and pmh.value[0]) then
   begin
      itself.success := false;
      exit;
   end;

   if pmh.value[0]>myhhv.value[0] then
      result := pmh.value[0]
   else
```

```
      result := myhhv.value[0];
end;
```

VBScript

```
function ex_vbs_prevmhigh()
dim pmh, myhhv

   pmh = itself.makeindicator("pmh", "prevMHigh", Array("1"),
Array(""))
   myhhv = itself.makeindicator("myhhv", "hhv", Array("1.h"),
Array("10"))

   if not (pmh.valid(0) and myhhv.valid(0)) then
      itself.success = false
      exit function
   end if

   if pmh.value(0)>myhhv.value(0) then
      ex_vbs_prevmhigh = pmh.value(0)
   else
      ex_vbs_prevmhigh = myhhv.value(0)
   end if
end function
```

Formula Language

```
plot1 := if(prevmhigh(data1)>hhv(h,30), prevmhigh(data1),
hhv(h,30));
```

Formula Column

```
if(prevMHigh(m15)>hhv(M15,30), prevMhigh(m15), hhv(M15,30))
```

Previous Month Low (prevMLow)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevMLow', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevMLow", Array("N"),
Array(""))
```

Formula Language

```
prevMLow(dataN)
```

Formula Column

```
prevMLow(Tn)
```

Definition

Similar to Previous Month Close, this indicator plots the previous month low in a series of dots on daily and weekly charts. Previous Month Low requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevmlow : double;
var pml, myllv : variant;
begin
   pml := itself.makeindicator('pml', 'prevmlow', ['1'], ['']);
   myllv := itself.makeindicator('myllv', 'llv', ['1.1'],
[param1.str]);

   if not (pml.valid[0] and myllv.valid[0]) then
   begin
      itself.success := false;
      exit;
   end;

   if pml.value[0]>myllv.value[0] then
      result := myllv.value[0]
   else
      result := pml.value[0];
end;
```

VBScript

```
function ex_vbs_prevmlow()
dim pml, myllv
   pml = itself.makeindicator("pml", "prevmlow", Array("1"),
Array(""))
   myllv = itself.makeindicator("myllv", "llv", _
                                Array("1.l"),
Array(param1.str))

   if not (pml.valid(0) and myllv.valid(0)) then
      itself.success = false
      exit function
   end if

   if pml.value(0)>myllv.value(0) then
      ex_vbs_prevmlow = myllv.value(0)
   else
      ex_vbs_prevmlow = pml.value(0)
   end if
end function
```

Formula Language

```
plot1 := if(prevmlow(data1)>llv(l,param1), llv(l,param1),
prevmlow(data1));
```

Formula Column

```
if(prevMLow(m15)>llv(M15,30), prevMlow(m15), llv(M15,30))
```

Previous Month Resistance (prevMResistance)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevMResistance', ['N'],
['string']);
```

VBScript

```
Itself.MakeIndicator("string", "prevMResistance", Array("N"),
Array("string"))
```

Formula Language

```
prevMResistance(dataN, "string")
```

Formula Column

```
prevMResistance(Tn, "string")
```

Definition

Borrowed from a day trading technique called support/resistance based on the previous trading day's traded range, the Previous Month Resistance calculates and plots the two resistance levels based on previous month's traded range. It is usually used with Previous Month Support. Previous Month Resistance takes one parameter Type, which is either Upper or Lower. Previous Month Resistance requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevmresistance  : double;
var pmr, pms : variant;
    mid_line : double;
begin
   pmr := itself.makeindicator('pmr', 'prevMResistance', ['1'],
['Upper']);
   pms := itself.makeindicator('pms', 'prevMSupport', ['1'],
['Upper']);

   if not (pmr.valid[0] and pms.valid[0]) then
   begin
      itself.successall := false;
      exit;
   end;

   mid_line := pms.value[0]+(pmr.value[0]-pms.value[0])/2;

   itself.plot[1] := (data1.value[0]-mid_line)/(mid_line-
pms.value[0]);
   itself.plot[2] := 0;

   if data1.value[0]>mid_line then
      itself.plot[3] := clgreen
   else
      itself.plot[3] := clred;
end;
```

VBScript

```
function ex_vbs_prevmresistance()
dim pmr, pms

   pmr = itself.makeindicator("pmr", "prevMResistance", _
                              Array("1"), Array("Upper"))
   pms = itself.makeindicator("pms", "prevMSupport", _
                              Array("1"), Array("Upper"))

   if not(pmr.valid(0) and pms.valid(0)) then
      itself.successall = false
      exit function
   end if

   mid_line = pms.value(0)+(pmr.value(0)-pms.value(0))/2

   itself.plot(1) = (data1.value(0)-mid_line)/(mid_line-
pms.value(0))
   itself.plot(2) = 0
   if data1.value(0)>mid_line then
      itself.plot(3) = clgreen
   else
      itself.plot(3) = clred
   end if
end function
```

Formula Language

```
$mid_line := prevmsupport(data1,"Upper")+
             (prevmresistance(data1,"Upper")-
prevmsupport(data1,"Upper"))/2;

plot1 := (data1-$mid_line)/($mid_line-
prevmsupport(data1,"Upper"));
plot2 := 0;
plot3 := if(data1>$mid_line, clgreen, clred);
```

Formula Column

```
$mid_line := prevmsupport(d1,"upper")+
            (prevmresistance(d1,"Upper")-
prevmsupport(d1,"Upper"))/2;
```

```
(c(d1)-$mid_line)/($mid_line-prevmsupport(d1,"upper"));
```

Previous Month Support (prevMSupport)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevMSupport', ['N'],
['string']);
```

VBScript

```
Itself.MakeIndicator("string", "prevMSupport", Array("N"),
Array("string"))
```

Formula Language

```
prevMSupport(dataN, "string")
```

Formula Column

```
prevMSupport(Tn, "string")
```

Definition

Borrowed from a day trading technique called support/resistance based on the previous trading day's traded range, the Previous Month Support calculates and plots the two support levels based on previous month's traded range. It is usually used with Previous Month Resistance. Previous Month Support takes one parameter Type, which is either Upper or Lower. Previous Month Support requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevmsupport : double;
var pms, pmr : varaint;
begin
   pms := itself.makeindicator('pms', 'prevmsupport', ['1'],
['Upper']);
   pmr := itself.makeindicator('pmr', 'prevmresistance', ['1'],
['Upper']);

   if not (pms.valid[0] and pmr.valid[0]) then
   begin
      itself.success := false;
      exit;
   end;
   if data1.value[0]>pmr.value[0] then
      result := 1
   else if data1.value[0]<pms.value[0] then
      result := -1
   else
      result := 0;
end;
```

VBScript

```
function ex_vbs_prevmsupport()
dim pms, pmr
   pms = itself.makeindicator("pms", "prevmsupport", _
                              Array("1"), Array("Upper"))
   pmr = itself.makeindicator("pmr", "prevmresistance", _
                              Array("1"), Array("Upper"))

   if not (pms.valid(0) and pmr.valid(0)) then
      itself.success = false
      exit function
   end if

   if data1.value(0)<pms.value(0) then
      ex_vbs_prevmsupport = -1
   elseif data1.value(0)>pmr.value(0) then
      ex_vbs_prevmsupport = 1
   else
      ex_vbs_prevmsupport = 0
   end if
end function
```

Formula Language

```
plot1 := if(data1<prevmsupport(data1,"Upper"),-1,
            if(data1>prevmresistance(data1,"Upper"),1,0));
```

Formula Column

```
if(c(m1)<prevmsupport(d1,"upper"),-
1,if(c(m1)>prevmresistance(d1,"upper"),1,0))
```

Previous Week Average (prevWAverage)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevWAverage', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevWAverage", Array("N"),
Array(""))
```

Formula Language

```
prevWAverage(dataN)
```

Formula Column

```
prevWAverage(Tn)
```

Definition

Previous Week Average is a support resistance indicator that marks the average of high, low and close of previous week with a dotted line and can automatically "jump" to the right level when the data switch over to the next week. Previous week average requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevwaverage : double;
var pwa : variant;
begin

   pwa := itself.makeindicator('pwa', 'PrevWAverage', ['1'],
['']);

   if not pwa.valid[0] then
   begin
      itself.success := false;
      exit;
   end;

   result := Data1.Value [0]-pwa.value[0];
end;
```

VBScript

```
function ex_vbs_prevwaverage()
dim pwa

   pwa = itself.makeindicator("pwa", "prevwaverage", _
                              Array("1"), Array(""))
   if pwa.valid(0) = false then
      itself.success = false
      exit function
   end if

   ex_vbs_prevwaverage = Data1.Value (0)-pwa.value (0)
end function
```

Formula Language

```
plot1 := data1-prevwaverage(data1);
```

Formula Column

```
if(c(m1)<prevwaverage(d1),-1,if(c(m1)>prevwaverage(d1),1,0))
```

Previous Week Close (prevWClose)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevWClose', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevWClose", Array("N"),
Array(""))
```

Formula Language

```
prevWClose(dataN)
```

Formula Column

```
prevWClose(Tn)
```

Definition

Given a daily or intraday chart, this indicator draws a series of horizontal dots marking the previous week closing price level and automatically adjust itself when the week has changed. One of the best support/resistance indicators available. Usually the complete group of Previous Week indicators are used together on the same chart. Previous Week Close requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevwclose : double;
var pwc, pwh : variant;
    myrange, rangeval : double;
begin
   pwc := itself.makeindicator('pwc', 'prevwclose', ['1'],
['']);
   pwh := itself.makeindicator('pwh', 'prevwhigh', ['1'],
['']);

   if not (pwc.valid[0] and pwh.valid[0]) then
   begin
     itself.success := false;
     exit;
   end;
   myrange := pwh.value[0]-pwc.value[0];

   if myrange > 0 then
   begin
      rangeval := data1.value[0]-pwc.value[0];
      result := rangeval/myrange;
   end
   else
      result := data1.value[0]/pwc.value[0];
end;
```

VBScript

```
function ex_vbs_prevwclose()
dim pwc, pwh

   pwc = itself.makeindicator("pwc", "prevwclose", Array("1"),
Array(""))
   pwh = itself.makeindicator("pwh", "prevwhigh", Array("1"),
Array(""))

   if not (pwc.valid(0) and pwh.valid(0)) then
      itself.success = false
      exit function
   end if

   myrange = pwh.value(0)-pwc.value(0)

   if myrange > 0 then
      rangeval = data1.close(0)-pwc.value(0)
      ex_vbs_prevwclose = rangeval/myrange
   else
      ex_vbs_prevwclose = data1.close(0)/pwc.value(0)
   end if
end function
```

Formula Language

```
$myrange := prevwhigh(data1)-prevwclose(data1);
plot1 := if($myrange>0, (data1-prevwclose(data1))/$myrange,
               data1/prevwclose(data1));
```

Formula Column

```
$myrange := prevwhigh(D1)-prevwclose(D1);
if($myrange>0, (close(D1)-prevwclose(D1))/$myrange,
close(D1)/prevwclose(D1))
```

Previous Week High (prevWHigh)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevWhigh', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevWhigh", Array("N"),
Array(""))
```

Formula Language

```
prevWhigh(dataN)
```

Formula Column

```
prevWhigh(Tn)
```

Definition

Similar to Previous Week Close, this indicator plots the previous week high in a series of dots on lower time frame charts. Previous Week High requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevwhigh : double;
var pwc, pwh, pwl : variant;
begin
   pwh := itself.makeindicator('pwh', 'prevwhigh', ['1'],
['']);
   pwc := itself.makeindicator('pwc', 'prevwclose', ['1'],
['']);
   pwl := itself.makeindicator('pwl', 'prevwlow', ['1'], ['']);

   if not (pwh.valid[0] and pwc.valid[0] and pwl.valid[0]) then
   begin
      itself.success := false;
      exit;
   end;

   result := (pwh.value[0]+pwl.value[0]+pwc.value[0])/3;
end;
```

VBScript

```
function ex_vbs_prevwhigh()
dim pwc, pwh, pwl

   pwh = itself.makeindicator("pwh", "prevwhigh", Array("1"),
Array(""))
   pwc = itself.makeindicator("pwc", "prevwclose", Array("1"),
Array(""))
   pwl = itself.makeindicator("pwl", "prevwlow", Array("1"),
Array(""))

   if not (pwh.valid(0) and pwc.valid(0) and pwl.valid(0)) then
      itself.success = false
      exit function
   end if

   ex_vbs_prevwhigh =
(pwh.value(0)+pwc.value(0)+pwl.value(0))/3
end function
```

Formula Language

```
plot1 :=
(prevwhigh(data1)+prevwlow(data1)+prevwclose(data1))/3;
```

Formula Column

```
prevwhigh(M1)
```

Previous Week Low (prevWLow)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevWlow', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "prevWlow", Array("N"),
Array(""))
```

Formula Language

```
prevWlow(dataN)
```

Formula Column

```
prevWlow(Tn)
```

Definition

Similar to Previous Week Close, this indicator plots the previous week low in a series of dots on lower time frame charts. Previous Week Low requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevwlow : double;
var pwl, pwh : variant;
begin

   pwl := itself.makeindicator('pwl', 'prevwlow', ['1'], ['']);
   pwh := itself.makeindicator('pwh', 'prevwhigh', ['1'],
['']);

   if not (pwl.valid[0] and pwh.valid[0]) then
   begin
      itself.success := false;
      exit;
   end;

   if data1.value[0] > pwh.value[0] then
      result := 1
   else if data1.value[0] < pwl.value[0] then
      result := -1
   else
```

```
      result := 0;
end;
```

VBScript

```
function ex_vbs_prevwlow()
dim pwl, pwh

   pwl = itself.makeindicator("pwl", "prevwlow", Array("1"),
Array(""))
   pwh = itself.makeindicator("pwh", "prevwhigh", Array("1"),
Array(""))

   if not (pwl.valid(0) and pwh.valid(0)) then
      itself.success = false
      exit function
   end if

   if data1.value(0) < pwl.value(0) then
      ex_vbs_prevwlow = -1
   elseif data1.value(0) > pwh.value(0) then
      ex_vbs_prevwlow = 1
   else
      ex_vbs_prevwlow = 0
   end if

end function
```

Formula Language

```
plot1 := if(data1<prevwlow(data1), -1,
if(data1>prevwhigh(data1), 1, 0));
```

Formula Column

```
if(last<prevwlow(m1), -1, if(last>prevwhigh(m1), 1, 0))
```

Previous Week Resistance (prevWResistance)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevWresistance', ['N'],
['string']);
```

VBScript

```
Itself.MakeIndicator("string", "prevWresistance", Array("N"),
Array("string"))
```

Formula Language

```
prevWresistance(dataN,"string")
```

Formula Column

```
prevWresistance(Tn,"string")
```

Definition

Borrowed from a day trading technique called support/resistance based on the previous trading day's traded range, the Previous Week Resistance calculates and plots the two resistance levels based on previous week's traded range. Usually used with Previous Week Support. Previous Week Resistance takes one parameter Type, which is either Upper or Lower. Previous Week Resistance requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevwresistance  : double;
var pwr : variant;
begin
   pwr := itself.makeindicator('pwr', 'prevwresistance', ['1'],
['upper']);

   if not pwr.valid[0] then
   begin
      itself.success := false;
      exit;
   end;

   if trade.openpositionflat then
   begin
```

```
      if (data1.value[0]<data1.value[1]) and
          ((pwr.value[0]-data1.value[2])<(data1.value[2]*0.01))
then
         trade.shortatmarket(param1.int, 'short entry');
   end
   else
   begin
      if trade.openordercount = 0 then
         trade.shortexitstop(trade.openpositionentryprice*1.02,
param1.int,
                             otfGoodtilCancel, 'stop 2% lost');

      if  data1.value[0] < (trade.openpositionentryprice*0.95)
then
      begin
         trade.cancelallopenorders;
         trade.shortexitatmarket(param1.int, 'exit 5% gain');
      end;

      if (data1.barsnum[0]-trade.openpositionentrybar) >= 20
then
      begin
         trade.cancelallopenorders;
         trade.shortexitatmarket(param1.int, 'exit 20 bars');
      end;
   end;
   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_prevwresistance()
dim pwr
   pwr = itself.makeindicator("pwr", "prevwresistance",
Array("1"), Array("upper"))

   if pwr.valid(0) = false then
      itself.success = false
      exit function
   end if

   if trade.openpositionflat then
      if (data1.value(0)<data1.value(1)) and _
         (pwr.value(0)-data1.value(2))<data1.value(2)*0.01 then
         trade.shortatmarket param1.int, "short entry"
      end if
   else
      if trade.openordercount = 0 then
         trade.shortexitstop trade.openpositionentryprice*1.02,
param1.int, _
                              otfGoodtilCancel, "stop 2% lost"
      end if

      if data1.value(0)<(trade.openpositionentryprice*0.95)
then
         trade.cancelallopenorders
```

```
            trade.shortexitatmarket param1.int, "exit 5% gain"
         end if

         if (data1.barsnum(0)-trade.openpositionentrybar)>=20 then
             trade.cancelallopenorders
             trade.shortexitatmarket param1.int, "exit 20 bars"
         end if
      end if

      ex_vbs_prevwresistance = trade.currentequity
   end function
```

Formula Language

```
$entrybar := if(openpositionshort>0, $entrybar+1, 0);
shortatmarket(openpositionflat>0 and data1(0)<data1(1) and
              (prevwresistance(data1, "upper")-data1(2))<0.01
and
              prevwresistance(data1, "upper")>0,
              param1, "short entry");
shortexitstop(openpositionshort>0,
openpositionaverageentryprice*1.02,
              param1, "stop 2% lost");
shortexitatmarket(openpositionshort>0 and
                  data1<openpositionaverageentryprice*0.95,
                  param1, "exit 5% gain");
shortexitatmarket(openpositionshort>0 and $entrybar>=20,
param1,
                  "exit 20 bars");
plot1 := currentequity;
```

Formula Column

```
prevwresistance(M1,"upper")
```

Previous Week Support (prevWSupport)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'prevWsupport', ['N'],
['string']);
```

VBScript

```
Itself.MakeIndicator("string", "prevWsupport", Array("N"),
Array("string"))
```

Formula Language

```
prevWsupport(dataN, "string")
```

Formula Column

```
prevWsupport(Tn, "string")
```

Definition

Borrowed from a day trading technique called support/resistance based on the previous trading day's traded range, the Previous Week Support calculates and plots the two support levels based on previous week's traded range. Usually used with Previous Week Resistance. Previous Week Support takes one parameter Type, which is either Upper or Lower. Previous Week Support requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_prevwsupport : double;
var pws : variant;
begin
   pws := itself.makeindicator('pws', 'prevwsupport', ['1'],
['upper']);

   if not pws.valid[0] then
   begin
      itself.success := false;
      exit;
   end;

   if trade.openpositionflat then
   begin
```

```
      if (data1.value[0]>data1.value[1]) and
         ((data1.value[2]-pws.value[0])<(data1.value[0]*0.01))
and
         (data1.value[0]>pws.value[0]) then
         trade.longatmarket(param1.int, 'long entry');
   end
   else
   begin
      if trade.openordercount=0 then
         trade.longexitstop(trade.openpositionentryprice*0.98,
param1.int,
                           otfGoodtilCancel, 'exit 2% lost');

      if data1.value[0]>trade.openpositionentryprice*1.05 then
      begin
         trade.cancelallopenorders;
         trade.longexitatmarket(param1.int, 'exit 5% gain');
      end;

      if (data1.barsnum[0]-trade.openpositionentrybar)>=20 then
      begin
         trade.cancelallopenorders;
         trade.longexitatmarket(param1.int, 'exit 20 bars');
      end;
   end;

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_prevwsupport()
dim pws

   pws = itself.makeindicator("pws", "prevwsupport",
Array("1"), Array("upper"))

   if not pws.valid(0) then
      itself.success = false
      exit function
   end if

   if trade.openpositionflat then
      if (data1.value(0)>data1.value(1)) and _
         ((data1.value(2)-pws.value(0))<(data1.value(0)*0.01))
and _
         (data1.value(0)>pws.value(0)) then
         trade.longatmarket param1.int, "long entry"
      end if
   else

      if (trade.openordercount=0) and _
         (trade.openpositionlong=true) then
         trade.longexitstop trade.openpositionentryprice*0.98,
param1.int, _
                           otfGoodtilCancel, "Exit stop 2%
```

```
lost"
         trade.longexitlimit trade.openpositionentryprice*1.05, 
param1.int, _
                              otfGoodtilCancel, "Exit limit 5% 
gain"
      end if

      if (data1.barsnum(0)-trade.openpositionentrybar)>=20 then
         trade.cancelallopenorders
         trade.longexitatmarket param1.int, "Exit 20 bars"
      end if
    end if

   ex_vbs_prevwsupport = trade.currentequity
end function
```

Formula Language

```
$entrybar := if(openpositionlong>0, $entrybar+1, 0);

longatmarket(prevwsupport(data1,"upper")>0 and data1>data1(1) 
and
             (data1(2)-
prevwsupport(data1,"upper"))<(data1*0.01) and
             data1(0)>prevwsupport(data1,"upper"),
              param1, "long entry");
longexitstop(openpositionlong>0, 
openpositionaverageentryprice*0.98, param1,
             "Exit 2% lost");
longexitlimit(openpositionlong>0, 
openpositionaverageentryprice*1.05, param1,
              "Exit 5% gain");
longexitatmarket($entrybar>20, param1, "exit 20 bars");
plot1 := currentequity;
```

Formula Column

```
prevwsupport(m1, "upper")
```

QC Bollinger Band (qc_bbands)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_bbands', ['N'], ['n',
'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "qc_bbands", Array("N"),
Array("n", "n")
```

Formula Language

```
qc_bbands(DataN, n, n);
```

Formula Column

```
qc_bbands(Tn, n, n)
```

Definition

This indicator emulate the behaviour of the "Bollinger Bands" studys from quote.com's charting application. It will automatic have band plot around a moving average.

Examples

Delphi Script

```
function test : double;
var myqc_bbands : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mybbands', 'qc_bbands', ['1'], ['20',
'2']);
   myqc_bbands := ItSelf.Indicator ('mybbands');

   if Data1.High [0] > myqc_bbands.ValueEx [1, 0] then
   begin
     ItSelf.Plot [1] := data1.high [0];
```

```
      ItSelf.SuccessEx [2] := false;
      exit;
    end;

    if Data1.Low [0] < myqc_bbands.ValueEx [3, 0] then
    begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.Plot [2] := Data1.Low [0];
      exit;
    end;

    ItSelf.SuccessEx [1] := false;
    ItSelf.SuccessEx [2] := false;

end;
```

VBScript

```
function testvb()
dim myqc_bbands

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      exit function
   end if

   ItSelf.MakeIndicator "mybbands", "qc_bbands", Array("1"),
Array("20", "2")
   myqc_bbands = ItSelf.Indicator ("mybbands")

   if Data1.High (0) > myqc_bbands.ValueEx (1, 0) then
      ItSelf.Plot (1) = data1.high (0)
      ItSelf.SuccessEx (2) = false
      exit function
   end if

   if Data1.Low (0) < myqc_bbands.ValueEx (3, 0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.Plot (2) = data1.low  (0)
      exit function
   end if

   ItSelf.SuccessEx (1) = false
   ItSelf.SuccessEx (2) = false

end function
```

Formula Language

```
Success1 := if ( high > qc_bbands.p1(data1, 20, 2), 1, 0);
Success2 := if ( low  < qc_bbands.p3(data1, 20, 2), 1, 0);

plot1 := High;
plot2 := Low;
```

Formula Column

```
qc_bbands.p1 (m5, 20, 2)
```

QC Choppiness (qc_choppiness)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_choppiness', ['N'], ['n',
'n', 'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "qc_choppiness", Array{"N"),
Array("n", "n", "n")
```

Formula Language

```
qc_choppiness(DataN, n, n, n);
```

Formula Column

```
qc_choppiness(Tn, n, n, n)
```

Definition

The Choppines Index measures he ratio of the average price range over a Length of interval to the price range over the entire length.

Examples

Delphi Script

```
function choppiness_ex : double;
var mychoppiness : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('mychoppiness', 'qc_choppiness',
                         ['1'], ['14', '61.8', '38.2']);
   mychoppiness := ItSelf.Indicator ('mychoppiness');

   if mychoppiness.ValueEx [1, 0] > mychoppiness.ValueEx [2, 0]
then
   begin
```

```
      ItSelf.plot [1] := Data1.High [0];
      ItSelf.SuccessEx [2] := false;
    end
    else if mychoppiness.ValueEx [1, 0] < mychoppiness.ValueEx
[3, 0] then
    begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.plot [2] := Data1.low [0];
    end
    else
    begin
      ItSelf.SuccessEx [1] := false;
      ItSelf.SuccessEx [2] := false;
      exit;
    end;

end;
```

VBScript

```
function choppiness_ex()
dim mychoppiness

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      exit function
   end if

   ItSelf.MakeIndicator "mychoppiness", "qc_choppiness",
Array("1"), _
                        Array("14", "61.8", "38.2")
   mychoppiness = ItSelf.Indicator ("mychoppiness")

   if mychoppiness.ValueEx (1, 0) > mychoppiness.ValueEx (2, 0)
then
      ItSelf.Plot (1) = data1.high (0)
      ItSelf.SuccessEx (2) = false
      exit function
   end if

   if mychoppiness.ValueEx (1,0) < mychoppiness.ValueEx (3, 0)
then
      ItSelf.SuccessEx (1) = false
      ItSelf.Plot (2) = data1.low  (0)
      exit function
   end if

   ItSelf.SuccessEx (1) = false
   ItSelf.SuccessEx (2) = false

end function
```

Formula Language

```
Success1 := if ( qc_choppiness.p1 (data1, 14, 61.8, 38.2) >
                   qc_choppiness.p2 (data1, 14, 61.8, 38.2), 1,
0);
Success2 := if ( qc_choppiness.p1 (data1, 14, 61.8, 38.2) <
                   qc_choppiness.p3 (data1, 14, 61.8, 38.2), 1,
0);

plot1 := High;
plot2 := Low;
```

Formula Column

```
qc_choppiness.p1 (M5, 14, 61.8, 38.2)
```

QC Directional Movement (qc_ADX)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_ADX', ['N'], ['n', 'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "qc_ADX", Array("N"), Array("n",
"n")
```

Formula Language

```
qc_ADX(DataN, n, n);
```

Formula Column

```
qc_ADX(Tn, n, n)
```

Definition

In Directional Movement the +DI line shows the average percentage of the range that is upward movement; -DI line shows the percentage that is downward movement. The crossing ot these lines indicates a change in trend. The ADXR line indicates how much the instrument is trending - below 20 indicates a non-trending environment.

Examples

Delphi Script

```
function test : double;
var myADX : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.Success := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myADX', 'qc_ADX', ['1'], ['14',
'14']);
   myADX := ItSelf.Indicator ('myADX');

   if myADX.ValidEx [1, 0] then
     result := myADX.ValueEx [1, 0] - myADX.ValueEx [2, 0]
   else
```

```
      ItSelf.Success := false;

end;
```

VBScript

```
function testvb()
dim myADX

   if not Data1.Valid (0) then
      ItSelf.Success = false
      exit function
   end if

   ItSelf.MakeIndicator "myADX", "qc_ADX", Array("1"),
Array("14", "14")
   myADX = ItSelf.Indicator ("myADX")

   if myADX.ValidEx (1, 0) then
      testvb = myADX.ValueEx (1, 0) - myADX.ValueEx (2, 0)
   else
      ItSelf.Success = false
   end if

end function
```

Formula Language

```
plot1 := qc_ADX.p1 (Data1, 14, 14) - qc_ADX.p2 (Data1, 14, 14);
```

Formula Column

```
qc_ADX(m15, 14, 14)
```

QC Donchian Channel (qc_donchian)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_donchian', ['1'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "qc_donchian", Array ("N"),
Array ("n")
```

Formula Language

```
qc_donchian(DataN, n);
```

Formula Column

```
qc_donchian(Tn, n)
```

Definition

This indicator emulate the "Donchain Channel" study from quote.com charting application. Donchain channel plot the highest high and the lowest low over a specified interval Length.

Examples

Delphi Script

```
function donchian_ex : double;
var donchianfast, donchianslow : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     ItSelf.SuccessEx [5] := false;
     ItSelf.SuccessEx [6] := false;
     ItSelf.SuccessEx [7] := false;
     ItSelf.SuccessEx [8] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('donchianfast', 'qc_donchian', ['1'],
['20']);
   donchianfast := ItSelf.Indicator ('donchianfast');

   ItSelf.MakeIndicator ('donchianslow', 'qc_donchian', ['1'],
['40']);
   donchianslow := ItSelf.Indicator ('donchianslow');

   if not (donchianfast.ValidEx [1, 0] and donchianslow.ValidEx
[1, 0] ) then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     ItSelf.SuccessEx [5] := false;
     ItSelf.SuccessEx [6] := false;
     ItSelf.SuccessEx [7] := false;
     ItSelf.SuccessEx [8] := false;
     exit;
   end;

   ItSelf.Plot [1] := donchianfast.ValueEx [1, 0];
   ItSelf.Plot [2] := donchianfast.ValueEx [2, 0];
   ItSelf.Plot [3] := donchianslow.ValueEx [1, 0];
   ItSelf.Plot [4] := donchianslow.ValueEx [2, 0];

   If donchianfast.ValueEx [1, 0] > donchianfast.ValueEx [1, 1]
then
      ItSelf.Plot [5] := Data1.High [0]
   else
      ItSelf.SuccessEx [5] := false;

   if donchianfast.ValueEx [2, 0] < donchianfast.ValueEx [2, 1]
then
      ItSelf.Plot [6] := Data1.Low [0]
   else
      ItSelf.SuccessEx [6] := false;

   if donchianslow.ValueEx [1, 0] > donchianslow.ValueEx [1, 1]
then
      ItSelf.Plot [7] := Data1.High [0]
   else
      ItSelf.SuccessEx [7] := false;

   if donchianslow.ValueEx [2, 0] < donchianslow.ValueEx [2, 1]
then
      ItSelf.Plot [8] := Data1.Low [0]
   else
      ItSelf.SuccessEx [8] := false;

end;
```

VBScript

```
function donchian_ex ()
dim donchianfast, donchianslow
```

```
   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      ItSelf.SuccessEx (5) = false
      ItSelf.SuccessEx (6) = false
      ItSelf.SuccessEx (7) = false
      ItSelf.SuccessEx (8) = false
      exit function
   end if

   ItSelf.MakeIndicator "donchianfast", "qc_donchian",
Array("1"), Array("20")
   donchianfast = ItSelf.Indicator ("donchianfast")

   ItSelf.MakeIndicator "donchianslow", "qc_donchian",
Array("1"), Array("40")
   donchianslow = ItSelf.Indicator ("donchianslow")

   if not (donchianfast.ValidEx (1, 0) and donchianslow.ValidEx
(1, 0)) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      ItSelf.SuccessEx (5) = false
      ItSelf.SuccessEx (6) = false
      ItSelf.SuccessEx (7) = false
      ItSelf.SuccessEx (8) = false
      exit function
   end if

   ItSelf.Plot (1) = donchianfast.ValueEx (1, 0)
   ItSelf.Plot (2) = donchianfast.ValueEx (2, 0)
   ItSelf.Plot (3) = donchianslow.ValueEx (1, 0)
   ItSelf.Plot (4) = donchianslow.ValueEx (2, 0)

   if donchianfast.ValueEx (1, 0) > donchianfast.ValueEx (1, 1)
then
      ItSelf.Plot (5) = Data1.High (0)
   else
      ItSelf.SuccessEx (5) = false
   end if

   if donchianfast.ValueEx (2, 0) < donchianfast.ValueEx (2, 1)
then
      ItSelf.Plot (6) = Data1.High (0)
   else
      ItSelf.SuccessEx (6) = false
   end if

   if donchianslow.ValueEx (1, 0) > donchianslow.ValueEx (1, 1)
then
      ItSelf.Plot (7) = Data1.Low (0)
   else
      ItSelf.SuccessEx (7) = false
   end if

   if donchianslow.ValueEx (2, 0) < donchianslow.ValueEx (2, 1)
then
      ItSelf.Plot (8) = Data1.High (0)
   else
      ItSelf.SuccessEx (8) = false
   end if

end function
```

Formula Language

```
Success5 := if ( qc_donchian.p1 (0, data1, param1) >
                 qc_donchian.p1 (1, data1, param1), 1, 0);
Success6 := if ( qc_donchian.p2 (0, data1, param1) <
                 qc_donchian.p2 (1, data1, param1), 1, 0);
Success7 := if ( qc_donchian.p1 (0, data1, param2) >
                 qc_donchian.p1 (1, data1, param2), 1, 0);
Success8 := if ( qc_donchian.p2 (0, data1, param2) <
                 qc_donchian.p2 (1, data1, param2), 1, 0);

plot1 := qc_donchian.p1 (data1, param1);
plot2 := qc_donchian.p2 (data1, param1);
plot3 := qc_donchian.p1 (data1, param2);
plot4 := qc_donchian.p2 (data1, param2);
plot5 := High;
plot6 := low;
plot7 := High;
plot8 := low;
```

Formula Column

```
qc_donchian.p1 (M5, 20)
```

QC Envelopes (qc_envelopes)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_envelopes', ['N'], ['n',
'string', 'n', 'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "qc_envelopes", Array("N"),
Array("n", "string", "n", "n")
```

Formula Language

```
qc_envelopes (DataN, n, "string", n, n)
```

Formula Column

```
qc_envelopes (Tn, n, "string", n, n)
```

Definition

Envelopes create bands at a chosen percentage plus and minus a moving average. The default percent of 10 yields bands at 110% and 90% of the moving average values.

Examples

Delphi Script

```
function envelopes_ex : double;
var myenvelopes : variant;
begin

   if not data1.valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     ItSelf.SuccessEx [5] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('myenvelopes', 'qc_envelopes',
                         ['1'], [params.items [1].str,
params.items [2].str,
                                 params.items [3].str,
params.items [4].str]);
   myenvelopes := ItSelf.Indicator ('myenvelopes');

   if not myenvelopes.ValidEx [1, 0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     ItSelf.SuccessEx [5] := false;
     exit;
   end;

   ItSelf.Plot [1] := myenvelopes.ValueEx [1, 0];
   ItSelf.Plot [2] := myenvelopes.ValueEx [2, 0];
   ItSelf.Plot [3] := myenvelopes.ValueEx [3, 0];

   if Data1.High [0] > myenvelopes.ValueEx [2, 0] then
      ItSelf.Plot [4] := Data1.High [0]
   else
      ItSelf.SuccessEx [4] := false;

   if Data1.Low [0] < myenvelopes.ValueEx [3, 0] then
      ItSelf.Plot [5] := Data1.Low [0]
   else
      ItSelf.SuccessEx [5] := false;

end;
```

VBScript

```
function envelopes_ex ()
dim myenvelopes

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      ItSelf.SuccessEx (5) = false
      exit function
   end if

   ItSelf.MakeIndicator "myenvelopes", "qc_envelopes",
Array("1"), _
         Array(params(1).str, params(2).str, params(3).str,
params(4).str)
   myenvelopes = ItSelf.Indicator ("myenvelopes")

   if not myenvelopes.ValidEx (1, 0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      ItSelf.SuccessEx (5) = false
      exit function
   end if

   ItSelf.Plot (1) = myenvelopes.ValueEx (1, 0)
   ItSelf.Plot (2) = myenvelopes.ValueEx (2, 0)
   ItSelf.Plot (3) = myenvelopes.ValueEx (3, 0)

   if Data1.High (0) > myenvelopes.ValueEx (2, 0) then
      ItSelf.Plot (4) = Data1.High (0)
   else
      ItSelf.SuccessEx (4) = false
   end if

   if Data1.Low (0) < myenvelopes.ValueEx (3, 0) then
      ItSelf.Plot (5) = Data1.low (0)
   else
      ItSelf.SuccessEx (5) = false
   end if

end function
```

Formula Language

```
Success4 := if (high >
                qc_envelopes.p2 (data1, param1, param2, param3,
param4), 1, 0);
Success5 := if (low <
                qc_envelopes.p3 (data1, param1, param2, param3,
param4), 1, 0);

plot1 := qc_envelopes.p1 (data1, param1, param2, param3,
param4);
plot2 := qc_envelopes.p2 (data1, param1, param2, param3,
param4);
plot3 := qc_envelopes.p3 (data1, param1, param2, param3,
param4);
plot4 := high;
plot5 := low;
```

Formula Column

```
qc_envelopes.p1 (m3, 20, "off", 4, 4)
```

QC Exponential Moving Average (qc_xaverage)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_xaverage', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "qc_xaverage", Array("N"),
Array("n")
```

Formula Language

```
qc_xaverage(DataN, n);
```

Formula Column

```
qc_xaverage(Tn, n)
```

Definition

Exponential Moving Average is effectively the mean of the complete data series since the beginning up to the point of calculation.

Examples

Delphi Script

```
function qc_xaverage_ex : double;
var fastxavg, slowxavg : variant;
    myxabove, myxbelow : variant;
begin

   if not Data1.Valid [0] then
   begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     exit;
   end;

   ItSelf.MakeIndicator ('fastxavg', 'qc_xaverage', ['1'],
[param1.str]);
   fastxavg := ItSelf.Indicator ('fastxavg');

   ItSelf.MakeIndicator ('slowxavg', 'qc_xaverage', ['1'],
[param2.str]);
   slowxavg := ItSelf.Indicator ('slowxavg');

   ItSelf.MakeIndicator ('myxabove', 'xabove', ['fastxavg',
'slowxavg'], ['']);
   myxabove := ItSelf.Indicator ('myxabove');

   ItSelf.MakeIndicator ('myxbelow', 'xbelow', ['fastxavg',
'slowxavg'], ['']);
   myxbelow := ItSelf.Indicator ('myxbelow');

   if fastxavg.Valid [0] then
      ItSelf.Plot [1] := fastxavg.value [0]
   else
      ItSelf.SuccessEx [1] := false;

   if slowxavg.Valid [0] then
      ItSelf.Plot [2] := slowxavg.Value [0]
   else
      ItSelf.SuccessEx [2] := false;

   if myxabove.Value [0] > 0 then
      ItSelf.Plot [3] := Data1.High [0]
   else
      ItSelf.SuccessEx [3] := false;

   if myxbelow.Value [0] > 0 then
      ItSelf.Plot [4] := Data1.Low [0]
   else
      ItSelf.SuccessEx [4] := false;

end;
```

VBScript

```
function qc_xaverage_vb()
dim fastxavg, slowxavg
dim myxabove, myxbelow

    if not Data1.Valid (0) then
       ItSelf.SuccessEx (1) = false
       ItSelf.SuccessEx (2) = false
       ItSelf.SuccessEx (3) = false
       ItSelf.SuccessEx (4) = false
       exit function
    end if

    ItSelf.MakeIndicator "fastxavg", "qc_xaverage", _
                         Array ("1"), Array (param1.str)
    fastxavg = ItSelf.Indicator ("fastxavg")

    ItSelf.MakeIndicator "slowxavg", "qc_xaverage", _
                         Array ("1"), Array (param2.str)
    slowxavg = ItSelf.Indicator ("slowxavg")

    ItSelf.MakeIndicator "myxabove", "xabove", _
```

```
                          Array ("fastxavg", "slowxavg"), Array
("")
    myxabove = ItSelf.Indicator ("myxabove")

    ItSelf.MakeIndicator "myxbelow", "xbelow", _
                          Array ("fastxavg", "slowxavg"),
Array("")
    myxbelow = Itself.Indicator ("myxbelow")

    if fastxavg.Valid (0) then
       ItSelf.Plot (1) = fastxavg.Value (0)
    else
       ItSelf.SuccessEx (1) = false
    end if

    if slowxavg.Valid (0) then
       ItSelf.Plot (2) = slowxavg.Value (0)
    else
       ItSelf.SuccessEx (2) = false
    end if

    if myxabove.Value (0) > 0 then
       ItSelf.Plot (3) = Data1.high (0)
    else
       ItSelf.SuccessEx (3) = false
    end if

    if myxbelow.Value (0) > 0 then
       ItSelf.Plot (4) = Data1.Low (0)
    else
       Itself.SuccessEx (4) = false
    end if

end function
```

Formula Language

```
fastxavg := qc_xaverage(Data1, param1);
slowxavg := qc_xaverage(Data1, param2);

success3 := if (xabove(fastxavg,slowxavg) > 0, 1, 0);
success4 := if (xbelow(fastxavg,slowxavg) > 0, 1, 0);

plot1 := fastxavg;
plot2 := slowxavg;
plot3 := high;
plot4 := low;
```

Formula Column

```
qc_xaverage (M5, 20)
```

QC MACD (qc_macd)

Syntax

Delphi Script

```
ItSelf.MakeIncidator ('string', 'qc_macd', ['N'], ['n', 'n',
'string', 'n']);
```

VBScript

```
ItSelf.MakeIndicatro "string", "qc_macd", Array("N"),
Array("n", "n", "string", "n")
```

Formula Language

```
qc_macd(DataN, n, n, "string", n);
```

Formula Column

```
qc_macd(Tn, n, n, "string", n)
```

Definition

Moving Average Convergence/Divergence provides early clues of a trend reversal. The MACD line represents the difference between a fast exponential moving average (Length1) and a slow one (Length2) - check Simple Moving Average to make this line an oscillator. The Signal Line is an exponential moving average of the MACD line. A buy signal occurs when the MACD line crosses above the Signal line; crossing below is a sell signal. A histogram plots their difference. Crossing the zero line confirms the trend.

Examples

Delphi Script

```
function qc_MACD_ex : double;
var mymacd : variant;
    bullishxover, bearishxover : boolean;
    bullcenterxover, bearcenterxover : boolean;
begin

  if not Data1.Valid [0] then
  begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     ItSelf.SuccessEx [5] := false;
     exit;
  end;

  ItSelf.MakeIndicator ('mymacd', 'qc_MACD', ['1'],
                        [param1.str, param2.str, param3.str,
param4.str]);
  mymacd := ItSelf.Indicator ('mymacd');

  if not mymacd.ValidEx [1, 0] then
  begin
     ItSelf.SuccessEx [1] := false;
     ItSelf.SuccessEx [2] := false;
     ItSelf.SuccessEx [3] := false;
     ItSelf.SuccessEx [4] := false;
     ItSelf.SuccessEx [5] := false;
     exit;
  end;

  ItSelf.Plot [1] := mymacd.ValueEx [1, 0];
  ItSelf.Plot [2] := mymacd.ValueEx [2, 0];
  ItSelf.Plot [3] := mymacd.ValueEx [3, 0];

  //bullish moving average crossover
  bullishxover := (mymacd.ValueEx [1, 1] < mymacd.ValueEx [2,
1]) and
                  (mymacd.ValueEx [1, 0] > mymacd.ValueEx [2,
0]);

  //bullish centerline crossover
  bullcenterxover := (mymacd.ValueEx [1, 1] < 0) and
                     (mymacd.ValueEx [1, 0] > 0);

  //bearish moving average crossover
  bearishxover := (mymacd.ValueEx [1, 1] > mymacd.ValueEx [2,
1]) and
                  (mymacd.ValueEx [1, 0] < mymacd.ValueEx [2,
0]);

  //bearish centerline crossover
  bearcenterxover := (mymacd.ValueEx [1, 1] > 0) and
                     (mymacd.ValueEx [1, 0] < 0);

  if bullishxover and bullcenterxover then
     ItSelf.Plot [4] := Data1.High [0]
  else
     ItSelf.SuccessEx [4] := false;

  if bearishxover and bearcenterxover then
     ItSelf.Plot [5] := Data1.Low [0]
  else
     ItSelf.SuccessEx [5] := false;

end;
```

VBScript

```
function qc_MACD_vb()
dim mymacd

   if not Data1.Valid (0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      ItSelf.SuccessEx (5) = false
      exit function
   end if

   ItSelf.MakeIndicator "mymacd", "qc_MACD", Array("1"), _
                        Array(param1.str, param2.str,
param3.str, param4.str)
   mymacd = ItSelf.Indicator ("mymacd")

   if not mymacd.Valid (1, 0) then
      ItSelf.SuccessEx (1) = false
      ItSelf.SuccessEx (2) = false
      ItSelf.SuccessEx (3) = false
      ItSelf.SuccessEx (4) = false
      ItSelf.SuccessEx (5) = false
      exit function
   end if

   ItSelf.Plot (1) = mymacd.ValueEx (1, 0)
   ItSelf.Plot (2) = mymacd.ValueEx (2, 0)
   ItSelf.plot (3) = mymacd.ValueEx (3, 0)

   'MACD cross above singnal line
   bullishxabove = (mymacd.ValueEx (1, 1) < mymacd.ValueEx (2,
1)) and _
                   (mymacd.ValueEx (1, 0) > mymacd.ValueEx (2,
0))

   'MACD cross above center line
   bullcenterxover = (mymacd.ValueEx (1, 1) < 0) and _
                     (mymacd.ValueEx (1, 0) > 0)

   'MACD cross below signal line
   bearishxbelow = (mymacd.ValueEx (1, 1) > mymacd.ValueEx (2,
1)) and _
                   (mymacd.ValueEx (1, 0) < mymacd.ValueEx (2,
0))

   'MACD cross below center line
   bearcenterxover = (mymacd.ValueEx (1, 1) > 0) and _
                     (mymacd.ValueEx (1, 0) < 0)

   if bullishxabove and bullcenterxover then
      ItSelf.Plot (4) = Data1.High (0)
   else
```

```
      ItSelf.Successex (4) = false
   end if

   if bearishxbelow and bearcenterxover then
      ItSelf.Plot (5) = Data1.Low (0)
   else
      ItSelf.SuccessEx (5) = false
   end if

end function
```

Formula Language

```
myMACD0 := qc_MACD.p1 (0, Data1, param1, param2, param3,
param4);
myMACD1 := qc_MACD.p1 (1, Data1, param1, param2, param3,
param4);
mysignal0 := qc_MACD.p2 (0, Data1, param1, param2, param3,
param4);
mysignal1 := qc_MACD.p2 (1, Data1, param1, param2, param3,
param4);

bullishxabove := (myMACD1 < mysignal1) and (myMACD0 >
mysignal0);
bullxcenter   := (myMACD1 < 0) and (myMACD0 > 0);

bearishxbelow := (myMACD1 > mysignal1) and (myMACD0 <
mysignal0);
bearxcenter   := (myMACD1 > 0) and (myMACD0 < 0);

Success4 := if (bullishxabove > 0 and bullxcenter > 0, 1, 0);
Success5 := if (bearishxbelow > 0 and bearxcenter > 0, 1, 0);

plot1 := qc_MACD.p1 (0, Data1, param1, param2, param3, param4);
plot2 := qc_MACD.p2 (0, Data1, param1, param2, param3, param4);
plot3 := qc_MACD.p3 (0, Data1, param1, param2, param3, param4);
plot4 := High;
plot5 := low;
```

Formula Column

```
qc_MACD.p1 (M15, 12, 26, "off", 9)
```

QC Momentum (qc_mo)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_mo', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "qc_mo", Array("N"), Array("n")
```

Formula Language

```
qc_mo (DataN, n);
```

Formula Column

```
qc_mo (Tn, n)
```

Definition

Momentum calculates the spread between the current price and the price Length intervals ago. A line of momentum calculations indicates if prices are changing at an increasing or decreasing rate. Crossing the zero line indicates a change in trend.

Examples

Delphi Script

```
function qc_momentum_ex : double;
var mymo : variant;
begin

  if not Data1.valid [0] then
  begin
    ItSelf.Successex [1] := false;
    ItSelf.SuccessEx [2] := false;
    exit;
  end;

  ItSelf.MakeIndicator ('mymo', 'qc_mo', ['1'], [param1.str]);
  mymo := ItSelf.Indicator ('mymo');

  if (mymo.Value [0] > mymo.Value [1]) and
     (mymo.Value [1] > mymo.value [2]) and
     (mymo.Value [2] > mymo.Value [3]) then
  begin
    Itself.Plot [1] := Data1.High [0];
    ItSelf.SuccessEx [2] := false;
    exit;
  end;

  if (mymo.Value [0] < mymo.Value [1]) and
     (mymo.Value [1] < mymo.Value [2]) and
     (mymo.Value [2] < mymo.Value [3]) then
  begin
    ItSelf.SuccessEx [1] := false;
    ItSelf.Plot [2] := Data1.Low [0];
    exit;
  end;

  ItSelf.SuccessEx [1] := false;
  ItSelf.SuccessEx [2] := false;

end;
```

VBScript

```
function qc_momentum_vb()
dim mymo

  if not Data1.Valid (0) then
    ItSelf.SuccessEx (1) = false
    ItSelf.SuccessEx (2) = false
    exit function
  end if

  ItSelf.MakeIndicator "mymo", "qc_mo", Array("1"),
Array(param1.str)
  mymo = ItSelf.Indicator ("mymo")

  if (mymo.Value (0) > mymo.value (1)) and _
     (mymo.Value (1) > mymo.Value (2)) and _
     (mymo.Value (2) > mymo.Value (3)) then
    ItSelf.Plot (1) = Data1.High (0)
    ItSelf.SuccessEx (2) = false
    exit function
  end if

  if (mymo.Value (0) < mymo.Value (1)) and _
     (mymo.Value (1) < mymo.Value (2)) and _
     (mymo.Value (2) < mymo.Value (3)) then
    ItSelf.SuccessEx (1) = false
    ItSelf.PLot (2) = Data1.Low (0)
    exit function
  end if

  ItSelf.SuccessEx (1) = false
  ItSelf.SuccessEx (2) = false

end function
```

Formula Language

```
mymo := qc_mo(data1, 14);

Success1 := if ((mymo(0) > mymo(1)) and (mymo(1) > mymo(2)) and
                (mymo(2) > mymo(3)), 1, 0);
Success2 := if ((mymo(0) < mymo(1)) and (mymo(1) < mymo(2)) and
                (mymo(2) < mymo(3)), 1, 0);

plot1 := high;
plot2 := low;
```

Formula Column

```
qc_mo(m5, 12)
```

QC Rate of Change (qc_roc)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_roc', ['N'], ['n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "qc_roc", Array("N"), Array("n")
```

Formula Language

```
qc_roc(DataN, n);
```

Formula Column

```
qc_roc(Tn, n)
```

Definition

Rate of Change performs the same calculations as the Momentum study, but expresses the spread as a percentage of the instrument's price.

Examples

Delphi Script

```
function qc_roc_ex : double;
var myroc : variant;
    pline, nline : double;
begin

  if not data1.valid [0] then
  begin
    ItSelf.SuccessEx [1] := false;
    ItSelf.SuccessEx [2] := false;
    exit;
  end;

  pline := param2.real/100;
  nline := param2.real/-100;

  ItSelf.MakeIndicator ('myroc', 'qc_roc', ['1'],
[param1.str]);
  myroc := ItSelf.Indicator ('myroc');

  if (myroc.ValueEx [1, 0] > pline) and
```

```
     (myroc.valueEx [1, 1] < pline) then
  begin
    ItSelf.Plot [1] := Data1.Low [0];
    ItSelf.SuccessEx [2] := false;
    exit;
  end;

  if (myroc.Valueex [1, 0] > nline) and
     (myroc.ValueEx [1, 1] < nline) then
  begin
    ItSelf.SuccessEx [1] := false;
    ItSelf.Plot [2] := Data1.high [0];
    exit;
  end;

  ItSelf.SuccessEx [1] := false;
  ItSelf.SuccessEx [2] := false;

end;
```

VBScript

```
function qc_roc_vb()
dim myroc

  if not Data1.Valid (0) then
    ItSelf.SuccessEx (1) = false
    ItSelf.SuccessEx (2) = false
    exit function
  end if

  pline = param2.real/100
  nline = param2.real/-100

  ItSelf.MakeIndicator "myroc", "qc_roc", Array("1"),
Array(param1.str)
  myroc = ItSelf.Indicator ("myroc")

  if (myroc.Value (0) > pline) and (myroc.value (1) < pline)
then
    ItSelf.Plot (1) = Data1.Low (0)
    ItSelf.SuccessEx (2) = false
    exit function
  end if

  if (myroc.Value (0) < nline) and (myroc.value (1) > nline)
then
    ItSelf.SuccessEx (1) = false
    ItSelf.Plot (2) = Data1.High (0)
    exit function
  end if

  ItSelf.SuccessEx (1) = false
  ItSelf.SuccessEx (2) = false

end function
```

Formula Language

```
pline := param2/100;
nline := param2/-100;

myroc := qc_roc(Data1, param1);

Success1 := if ((myroc > pline) and (myroc(1) < pline), 1, 0);
Success2 := if ((myroc > nline) and (myroc(1) < nline), 1, 0);

plot1 := low;
plot2 := high;
```

Formula Column

```
qc_roc (M5, 12)
```

QC RSI (qc_rsi)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_rsi', ['N'], ['n', 'n',
'n']);
```

VBscript

```
ItSelf.MakeIndicator "string", "qc_rsi", Array("N"), Array("n",
"n", "n")
```

Formula Language

```
qc_rsi(DataN, n, n, n);
```

Formula Column

```
qc_rsi(Tn, n, n, n)
```

Definition

Relative Strength Index is a counter trend Indicator that measures the ratio of upward closes to downward closes as a moving average. Fixed lines at 70 and 30 indicate overbought and oversold conditions. Traditional charting techniques like failure swings support and resistance, and divergence are used with the RSI line to identify trend reversals.

Examples

Delphi Script

```
function qc_rsi_ex : double;
var myrsi : variant;
begin

  if not Data1.Valid [0] then
  begin
    ItSelf.SuccessEx [1] := false;
    ItSelf.SuccessEx [2] := false;
    ItSelf.SuccessEx [3] := false;
    exit
  end;

  ItSelf.MakeIndicator ('myrsi', 'qc_rsi', ['1'],
                        [param1.str, param2.str, param3.str]);
  myrsi := ItSelf.Indicator ('myrsi');

  if (myrsi.ValueEx [1, 0] > myrsi.ValueEx [2, 0]) then
  begin
    ItSelf.Plot [1] := 1;
    ItSelf.Plot [2] := 0;
    ItSelf.SuccessEx [3] := clgreen;
  end
  else if (myrsi.ValueEx [1, 0] < myrsi.ValueEx [3, 0]) then
  begin
    ItSelf.Plot [1] := 1;
    ItSelf.Plot [2] := 0;
    ItSelf.plot [3] := clred;
  end
  else
  begin
    ItSelf.plot [1] := 1;
    ItSelf.plot [2] := 0;
    ItSelf.Plot [3] := clwhite;
  end;

end;
```

VBScript

```
function qc_rsi_vb()
dim myrsi

  if not Data1.Valid (0) then
    ItSelf.SuccessEx (1) = false
    ItSelf.SuccessEx (2) = false
    ItSelf.SuccessEx (3) = false
    exit function
  end if

  ItSelf.MakeIndicator "myrsi", "qc_rsi", Array("1"), _
                       Array(param1.str, param2.str,
param3.str)
  myrsi = ItSelf.Indicator ("myrsi")

  if myrsi.ValueEx (1, 0) > myrsi.ValueEx (2, 0) then
    ItSelf.Plot (1) = 1
    ItSelf.Plot (2) = 0
    ItSelf.Plot (3) = clgreen
  elseif myrsi.ValueEx (1, 0) < myrsi.ValueEx (3, 0) then
    ItSelf.Plot (1) = 1
    ItSelf.Plot (2) = 0
    ItSelf.Plot (3) = clred
  else
    ItSelf.Plot (1) = 1
    ItSelf.Plot (2) = 0
    ItSelf.Plot (3) = clwhite
  end if

end function
```

Formula Language

```
isgreen := qc_rsi.p1 (Data1, param1, param2, param3) >
           qc_rsi.p2 (Data1, param1, param2, param3);
isred   := qc_rsi.p1 (Data1, param1, param2, param3) <
           qc_rsi.p3 (data1, Param1, param2, param3);

plot3 := if(isgreen > 0, clgreen, if (isred > 0, clred,
clwhite));

plot1 := 1;
plot2 := 0;
```

Formula Column

```
qc_rsi (M5, 14, 80, 20)
```

QC StochRSI (qc_stochRSI)

Syntax

Delphi Script

```
ItSelf.MakeIndicator ('string', 'qc_stochRSI', ['N'], ['n',
'n', 'n', 'n', 'n']);
```

VBScript

```
ItSelf.MakeIndicator "string", "qc_stochRSI", Array("N"),
Array("n", "n", "n", "n", "n")
```

Formula Language

```
qc_stochRSI(DataN, n, n, n, n, n)
```

Formula Column

```
qc_stochRSI(Tn, n, n, n, n, n)
```

Definition

The Stochastic RSI is applying %K on to RSI and inturn apply %D smoothing on to %K.StochRSI requires one data series (Link 1).

Examples

Delphi Script

```
function qc_stochRSI_ex : double;
var mystochRSI : variant;
begin

  if not Data1.Valid [0] then
  begin
    ItSelf.SuccessEx [1] := false;
    ItSelf.SuccessEx [2] := false;
    exit;
  end;

  ItSelf.MakeIndicator ('mystochrsi', 'qc_stochRSI', ['1'],
     [param1.str, param2.str, param3.str, param4.str,
param5.str, param6.str]);
  mystochRSI := ItSelf.Indicator ('mystochrsi');

  if (mystochRSI.ValueEx [1, 0] < mystochRSI.ValueEx [3, 0])
and
```

```
     (mystochRSI.ValueEx [1, 1] > mystochRSI.ValueEX [3, 1])
then
  begin
    ItSelf.SuccessEx [1] := false;
    ItSelf.plot [2] := Data1.high [0];
    exit;
  end;

  if (mystochRSI.ValueEx [1, 0] > mystochRSI.ValueEx [4, 0])
and
     (mystochRSI.ValueEx [1, 1] < mystochRSI.ValueEx [4, 1])
then
  begin
    ItSelf.Plot [1] := Data1.Low [0];
    ItSelf.SuccessEx [2] := false;
    exit;
  end;

  ItSelf.SuccessEx [1] := false;
  ItSelf.SuccessEx [2] := false;

end;
```

VBScript

```
function qc_stochrsi_vb()
dim mysrsi

  if not Data1.Valid (0) then
    ItSelf.SuccessEx (1) = false
    ItSelf.SuccessEx (2) = false
    exit function
  end if

  ItSelf.MakeIndicator "mystochrsi", "qc_stochrsi", Array("1"),
_
                       Array(param1.str, param2.str,
param3.str, _
                             param4.str, param5.str,
param6.str)
  mysrsi = ItSelf.Indicator ("mystochrsi")

  if (mysrsi.ValueEx (1, 0) > mysrsi.ValueEx (3, 0)) and _
     (mysrsi.ValueEx (1, 1) < mysrsi.ValueEx (3, 1)) then
    ItSelf.SuccessEx (1) = false
    ItSelf.Plot (2) = Data1.High (0)
    exit function
  end if

  if (mysrsi.ValueEx (1, 0) < mysrsi.ValueEx (4, 0)) and _
     (mysrsi.ValueEx (1, 1) > mysrsi.ValueEx (4, 1)) then
    ItSelf.Plot (1) = Data1.Low (0)
    ItSelf.SuccessEx (2) = false
    exit function
  end if
  ItSelf.SuccessEx (1) = false
  ItSelf.SuccessEx (2) = false

end function
```

Formula Language

```
xupper := (qc_stochRSI.p1 (Data1, param1, param2, param3,
                                 param4, param5, param6) >
           qc_stochRSI.p3 (Data1, param1, param2, param3,
                                 param4, param5, param6)) and
          (qc_stochRSI.p1 (1, Data1, param1, param2, param3,
                                 param4, param5, param6) <
qc_stochRSI.p3 (1, Data1, param1, param2, param3,
                                  param4, param5, param6));
xlower := (qc_stochRSI.p1 (Data1, param1, param2, param3,
                                 param4, param5, param6) <
qc_stochRSI.p4 (Data1, param1, param2, param3,
                                 param4, param5, param6)) and
          (qc_stochRSI.p1 (1, Data1, param1, param2, param3,
                                 param4, param5, param6) >
           qc_stochRSI.p4 (1, Data1, param1, param2, param3,
                                  param4, param5, param6));

success1 := if (xupper > 0, 1, 0);
success2 := if (xlower > 0, 1, 0);

plot1 := low;
plot2 := high;
```

Random (rand)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'rand', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "rand", Array("N"), Array("N"))
```

Formula Language

```
rand(dataN,N)
```

Formula Column

```
rand(Tn,N)
```

Definition

Random takes one parameter Range and returns a random series of integers from 0 to Range - 1. Random requires one specified series (Link 1), which defines the time period when Random returns a valid value.

Examples

Delphi Script

```
function ex_del_random : double;
var ind_random, ind_avg : varaint;
begin
   ind_random := itself.makeindicator('myran', 'rand', ['1'],
[param1.str]);
   ind_avg := itself.makeindicator('myavg', 'average', ['1'],
['10']);

   if not (ind_random.valid[0] and ind_avg.valid[0]) then
   begin
      itself.success := false;
      exit;
   end;

   if ind_random.value[0] > 0 then
      itself.success := false
   else
      result := ind_avg.value [0];
end;
```

VBScript

```
function ex_vbs_random()
dim ind_random, ind_avg
   ind_random = itself.makeindicator("myran", "rand", _
                      Array("1"), Array(param1.str))
   ind_avg = itself.makeindicator("myavg", "average", _
                      Array("1"), Array("10"))

   if not (ind_random.valid(0) and ind_avg.valid(0)) then
      itself.success = false
      exit function
   end if

   if ind_random.value(0) > 0 then
      ex_vbs_random = ind_avg.value(0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
$ind_rand := rand(data1,param1);
plot1 := if($ind_rand>0, average(data1,10), 0);
success1 := if($ind_rand>0, 1, 0);
```

Formula Column

```
rand(m1, 2)
```

Range (range)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'range', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "range", Array("N"), Array(""))
```

Formula Language

```
range(dataN)
```

Formula Column

```
range(Tn)
```

Definition

Returns the range of the current bar of the data series. Range equals to the difference between high and low of a price bar. Range requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_range : double;
var avg_range : variant;
begin
   itself.makeindicator ('myrange', 'range', ['1'], ['']);
   avg_range := itself.makeindicator('myar', 'average',
                                     ['myrange'],
[param1.str]);

   if avg_range.valid[0] then
      result := avg_range.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_range ()
dim avg_range

   itself.makeindicator "myrange", "range", Array("1"),
Array("")
   avg_range = itself.makeindicator("myar", "average", _
                                    Array("myrange"),
Array(param1.str))

   if avg_range.valid(0) then
      ex_vbs_range  = avg_range.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := average(range(data1),param1);
```

Formula Column

```
range(m1)
```

Rate of Change (roc)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'roc', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "roc", Array("N"), Array("N"))
```

Formula Language

```
roc(dataN, N)
```

Formula Column

```
roc(Tn, N)
```

Definition

Rate of Change is the percentage change in value based on Period bars ago. It is similar to the Momentum indicator. Rate of Change requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_roc : double;
var ind_roc : variant;
    olevel, myprofit : double;
    mysize, target_point, trialstop_point, stopout_point :
integer;
begin
   mysize := param2.int;
   olevel := param3.real;

   target_point    := 10;
   trialstop_point := 2;
   stopout_point   := 3;

   ind_roc := itself.makeindicator('myroc', 'roc', ['1'],
[param1.str]);

   if not ind_roc.valid[0] then
   begin
      itself.success := false;
      exit;
   end;

   if trade.openpositionflat then
   begin
      trade.cancelallopenorders;

      if (ind_roc.value[0]>0) and
(ntlib.abs(ind_roc.value[0])>olevel) then
         trade.shortatmarket (mysize, 'overbought short
entry');

      if (ind_roc.value[0]<0) and
(ntlib.abs(ind_roc.value[0])>olevel) then
         trade.longatmarket (mysize, 'oversold long entry');
   end
   else
   begin
      if trade.openpositionlong then
      begin
         if trade.openordercount = 0 then
            trade.longexitstop(
                  trade.openpositionentryprice-
stopout_point*trade.minticksize,
                  mysize, otfGoodtilCancel, 'stop lost long');
         myprofit := trade.openpositionbestpricelevel-
                     trade.openpositionentryprice;
         if myprofit>target_point*trade.minticksize then
            trade.longexitstop(
                  (trade.openpositionbestpricelevel-
                  trialstop_point*trade.minticksize),
                  mysize, otffillorKill, 'trial stop long');
      end;

      if trade.openpositionshort then
      begin
         if trade.openordercount = 0 then
            trade.shortexitstop(

trade.openpositionentryprice+stopout_point*trade.minticksize,
                  mysize, otfGoodtilCancel, 'stop lost short');
         myprofit := trade.openpositionentryprice-
                     trade.openpositionbestpricelevel;
         if myprofit>target_point*trade.minticksize then
            trade.shortexitstop(
                 (trade.openpositionbestpricelevel+
                  trialstop_point*trade.minticksize),
                  mysize, otffillorkill, 'trial stop short');
      end;
   end;

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_roc()
```

```
dim ind_roc

   mysize = param2.int
   olevel = param3.real

   target_point    = 10
   trialstop_point = 2
   stopout_point   = 3

   ind_roc = itself.makeindicator("myroc", "roc", _
                                  Array("1"),
Array(param1.str))

   if not ind_roc.valid(0) then
      itself.success = false
      exit function
   end if

   if trade.openpositionflat then
      trade.cancelallopenorders

      if (ind_roc.value(0)>0) and (ind_roc.value(0)>olevel)
then
         trade.shortatmarket mysize, "overbought short"
      end if

      if (ind_roc.value(0)<0) and _
         (ntlib.abs(ind_roc.value(0))>olevel) then
         trade.longatmarket mysize, "oversold long"
      end if
   else
      if trade.openpositionlong then
         if trade.openordercount = 0 then
            trade.longexitstop _
                  trade.openpositionentryprice- _
                  (trade.minticksize*stopout_point), _
                  mysize, otfGoodtilCancel, "long stop"
         end if

         myprofit = trade.openpositionbestpricelevel- _
                    trade.openpositionentryprice
         if myprofit>target_point*trade.minticksize then
            trade.longexitstop _
                  trade.openpositionbestpricelevel- _
                  trade.minticksize*trialstop_point, _
                  mysize, otfFillorKill, "long trial stop"
         end if
      end if

      if trade.openpositionshort then
         if trade.openordercount = 0 then
            trade.shortexitstop _
                  trade.openpositionentryprice+ _
                  (trade.minticksize*stopout_point), _
                  mysize, otfGoodtilCancel, "short stop"
         end if
```

```
          myprofit = trade.openpositionentryprice- _
                     trade.openpositionbestpricelevel
          if myprofit>target_point*trade.minticksize then
             trade.shortexitstop _
                   trade.openpositionbestpricelevel+ _
                   (trade.minticksize*trialstop_point), _
                   mysize, otfFillorKill, "short trial stop"
          end if
       end if
     end if

   ex_vbs_roc = trade.currentequity
end function
```

Formula Language

```
$mysize := param2;
$olevel := param3;

$target_point := 10;
$trialstop_point := 3;
$stopout_point := 2;
$minsize := 0.01;

longatmarket (openpositionflat>0 and (roc(data1,param1)<0) and
              absvalue(roc(data1,param1))>$olevel, $mysize,
"oversold long");
shortatmarket (openpositionflat>0 and (roc(data1,param1)>0) and
              absvalue(roc(data1,param1))>$olevel, $mysize,
"overbought short");

longexitstop (openpositionlong>0 and
              (openpositionbestpricelevel-
openpositionaverageentryprice)>
              $target_point*$minsize,
              openpositionbestpricelevel-
$trialstop_point*$minsize,
              $mysize, "long trial stop");
longexitstop (openpositionlong>0,
              openpositionaverageentryprice-
$stopout_point*$minsize,
              $mysize, "long stop lost");

shortexitstop (openpositionshort>0 and
              (openpositionaverageentryprice-
openpositionbestpricelevel)>
              $target_point*$minsize,

openpositionbestpricelevel+$trialstop_point*$minsize,
              $mysize, "short trial stop");
shortexitstop (openpositionshort>0,

openpositionaverageentryprice+$stopout_point*$minsize,
              $mysize, "short stop lost");
plot1 := currentequity;
```

Formula Column

```
roc(m1, 20)
```

Reference (ref)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'ref', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "ref", Array("N"), Array("N"))
```

Formula Language

```
ref(dataN, N)
```

Formula Column

```
ref(Tn, N)
```

Definition

Returns the value of Link 1 shifted by Offset number of periods. A negative offset refer to values from previous periods. Positive offset is not allowed. Reference requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_ref : double;
var ind_ref, ind_mov : varint;
     myoffset : integer;
begin
   myoffset := -1*param2.int;

   ind_mov := itself.makeindicator('myavg', 'average', ['1'],
[param1.str]);
   ind_ref := itself.makeindicator('myref', 'ref',
                     ['myavg'], [ntlib.integer2str(myoffset)]);

   if not (ind_mov.valid[0] and ind_ref.valid[0]) then
   begin
      itself.success := false;
      exit;
   end;

   result := ind_mov.value[0]-ind_ref.value[0];
end;
```

VBScript

```
function ex_vbs_ref()
dim ind_ref, ind_avg

   myoffset = -1*param2.int

   ind_avg = itself.makeindicator("myavg", "average", _
                     Array("1"), Array(param1.str))
   ind_ref = itself.makeindicator("myref", "ref",
Array("myavg"), _

Array(ntlib.integer2str(myoffset)))

   if not (ind_avg.valid(0) and ind_ref.valid(0)) then
      itself.success = false
      exit function
   end if

   ex_vbs_ref = ind_avg.value(0)-ind_ref.value(0)
end function
```

Formula Language

```
$offset := -1*param2;
plot1 := average(data1, param1)-
ref(average(data1,param1),$offset);
```

Formula Column

```
ref(m1, -10)
```

Reference Time (reftime)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'reftime', ['N'],
['string','string']);
```

VBScript

```
Itself.MakeIndicator("string", "ref", Array("N"),
Array("string","string"))
```

Formula Language

```
ref(dataN, "string", "string")
```

Formula Column

```
ref(Tn, "string", "string")
```

Definition

Reference Time returns the open, high, low, close, volume of the user defined time range. You can also specify the number of trading days ago to return values for. Reference Time requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_reftime  : double;
var ind_reftime, ind_sar : variant;
    mysize : integer;
    target_price, stop_price : double;
begin
   mysize       := param1.int;
   target_price := 20*trade.minticksize;
   stop_price   := 10*trade.minticksize;

   ind_reftime  := itself.makeindicator('myreftime', 'reftime',
['1'],
                                                  ['9:30',
'9:45']);
   ind_sar      := itself.makeindicator('myparabolic',
'parabolic', ['1'],
                                                  ['0.02']);
   if not (ind_reftime.valid[0] and ind_sar.valid[0]) then
   begin
      itself.success := false;
      exit;
   end;

   if trade.openpositionflat then
   begin
      trade.cancelallopenorders;

      if (data1.high[0]>ind_reftime.valueex[2, 0]) and
         (ind_sar.value[0]<data1.low[0]) then
         trade.longatmarket(mysize, 'long 2 signals confirm');

      if (data1.low[0]<ind_reftime.valueex[3, 0]) and
         (ind_sar.value[0]>data1.high[0]) then
         trade.shortatmarket(mysize, 'short 2 signals
confirm');
   end
   else
   begin

      if trade.openordercount=0 then
      begin
         if trade.openpositionlong then
         begin
            trade.longexitstop (trade.openpositionentryprice-
stop_price,
                              mysize, otfgoodtilCancel, 'long
stop loss');
            trade.longexitlimit
(trade.openpositionentryprice+target_price,
                              mysize, otfgoodtilcancel, 'long
exit limit');
         end;

         if trade.openpositionshort then
         begin
            trade.shortexitstop
(trade.openpositionentryprice+stop_price,
                               mysize, otfgoodtilCancel,
'short stop loss');
            trade.shortexitlimit (trade.openpositionentryprice-
target_price,
                               mysize, otfgoodtilCancel,
'short exit limit');
         end;
      end;
   end;

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_reftime ()
```

```
dim ind_reftime, ind_sar

   ind_reftime = itself.makeindicator("myreftime", "reftime", _
                                      Array("1"), Array("9:30",
"9:45"))
   ind_sar = itself.makeindicator("mysar", "parabolic", _
                                      Array("1"),
Array("0.02"))

   mysize = param1.int
   target_price = 10*trade.minticksize
   stop_price = 5*trade.minticksize

   if not (ind_reftime.valid(0) and ind_sar.valid(0)) then
      itself.success = false
      exit function
   end if

   if trade.openpositionflat then
      trade.cancelallopenorders
      if (data1.high(0)>ind_reftime.valueex(2, 0)) and _
         (ind_sar.value(0)<data1.low(0)) then
         trade.longatmarket mysize, "long 2 signals confirm"
      end if
      if (data1.low(0)<ind_reftime.valueex(3, 0)) and _
         (ind_sar.value(0)<data1.low(0)) then
         trade.shortatmarket mysize, "short 2 signals confirm"
      end if
   else
      if trade.openordercount=0 then
         if trade.openpositionlong = true then
            trade.longexitstop trade.openpositionentryprice-
stop_price, _
                               mysize, otfgoodtilcancel, "long
stop loss"
            trade.longexitlimit
trade.openpositionentryprice+target_price, _
                                mysize, otfgoodtilcancel, "long
exit limit"
         end if

         if trade.openpositionshort = true then
            trade.shortexitstop
trade.openpositionentryprice+stop_price, _
                                mysize, otfgoodtilcancel,
"short stop loss"
            trade.shortexitlimit trade.openpositionentryprice-
target_price, _
                                mysize, otfgoodtilcancel,
"short exit limit"
         end if
      end if
   end if

   ex_vbs_reftime  = trade.currentequity
end function
```

Formula Language

```
$myminticksize := 0.01;
$mysize := param1;
$target_price := 10*$myminticksize;
$stop_price := 5*$myminticksize;

longatmarket (openpositionflat>0 and
              h>reftime.plot2(data1, "9:30", "945") and
              parabolic(data1, 0.02)<l, $mysize, "long 2
signals confirm");
shortatmarket (openpositionflat>0 and
               l<reftime.plot3(data1, "9:30", "9:15") and
               parabolic(data1, 0.02)>h, $mysize, "short 2
signals confirm");

longexitstop (openpositionlong>0,
              openpositionaverageentryprice-stop_price,
              $mysize, "long stop lost");
longexitlimit (openpositionlong>0,
               openpositionaverageentryprice+target_price,
               $mysize, "lone exit limit");

shortexitstop (openpositionshort>0,
               openpositionaverageentryprice+stop_price,
               $mysize, "short stop lost");
shortexitlimit (openpositionshort>0,
                openpositionaverageentryprice-targer_price,
                $mysize, "short exit limit");
plot1 := currentequity
```

Formula Column

```
if(reftime.plot2(m1, "9:30", "9:45")<h(m1), 1, 0)
```

Reference Time Ex (reftimeex)

Reftimeex is identical to ***Reference Time (reftime)*** (on page 2113), except reftimeex provides an extra option to reset calculation when trading day changes. Resetting is useful when you prefer reftimeex not to remember values from previous trading day.

Reference Valid (refvalid)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'refvalid', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "refvalid", Array("N"),
Array("N"))
```

Formula Language

```
refvalid(dataN, N)
```

Formula Column

```
refvalid(Tn, N)
```

Definition

Returns the value of Link 1 shifted by Offset number of valid periods. A negative offset refer to values from previous periods. Positive offset is not allowed. Reference requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_refvalid : double;
var ind_macd, ind_refvalid : variant;

begin
   ind_macd := itself.makeindicator ('mymacd', 'MACD', ['1'],
                      ['simple', param1.str, 'simple',
param2.str]);
   ind_refvalid := itself.makeindicator ('myref', 'refvalid',
                                          ['mymacd'],
[param3.str]);

   if not (ind_macd.valid[0] and ind_refvalid.valid[0]) then
   begin
      itself.successall := false;
      exit;
   end;

   itself.plot[1] := ind_macd.Value [0];
   itself.plot[2] := ind_macd.value[0]-ind_refvalid[0];
end;
```

VBScript

```
function ex_vbs_refvalid()
dim ind_refvalid, ind_macd

   ind_macd = itself.makeindicator ("mymacd", "MACD",
Array("1"), _
                    Array("Simple", param1.str, "simple",
param2.str))
   ind_refvalid = itself.makeindicator ("myrefvalid",
"refvalid", _
                        Array("mymacd"), Array(param3.str))

   if not (ind_macd.valid(0) and ind_refvalid.valid(0)) then
      itself.successall = false
      exit function
   end if

   itself.plot(1) = ind_macd.value(0)
   itself.plot(2) = ind_macd.value(0)-ind_refvalid.Value (0)
end function
```

Formula Language

```
$ind_refvalid := refvalid(macd(data1, "simple", param1,
"simple", param2), param3);
$mymacd := macd(data1, "simple", param1, "simple", param2);

plot1 := $mymacd;
plot2 := $mymacd-$ind_refvalid;
```

Formula Column

```
refvalid(m1, -10)
```

Relative Strength (rs)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'rs', ['N'],
['string','string']);
```

VBScript

```
Itself.MakeIndicator("string", "rs", Array("N"),
Array("string","string"))
```

Formula Language

```
rs(dataN, "string", "string")
```

Formula Column

```
rs(Tn, "string", "string")
```

Definition

Relative Strength indicator can "normalize" a source series, say prices of a stock, to a specific starting point in time (e.g. Jan 1, 1995) and plot the percentage or absolute changes since that time period. The starting point for measuring percentage change is 100% while the starting point of absolute change is 0. Relative Strength requires one specified series (Link 1).

Examples

Delphi Script

```
function ex_del_rs : double;
var ind_rs : variant;
begin
   itself.makeindicator ('mymov', 'average', ['1'],
[param1.str]);
   ind_rs := itself.makeindicator('myrs', 'rs', ['mymov'],
                                   [param2.str, param3.str]);

   if not ind_rs.valid[0] then
   begin
      itself.success := false;
      exit;
   end;
   result := ind_rs.Value [0];
end;
```

VBScript

```
function ex_vbs_rs()
dim ind_rs
   itself.makeindicator "mymov", "average", Array("1"),
Array(param1.str)
   ind_rs = itself.makeindicator ("myrs", "rs", Array("mymov"),
_
                                  Array(param2.str,
param3.str))

   if not ind_rs.valid(0) then
      itself.success = false
      exit function
   end if

   ex_vbs_rs = ind_rs.Value (0)
end function
```

Formula Language

```
plot1 := rs(average(data1, param1), param2, param3);
```

Formula Column

```
rs(m1, "Percent", "1/1/1990")
```

Relative Strength Index Modify (rsindexMod)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'rsindexmod', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "rsindexmod", Array("N"),
Array("N"))
```

Formula Language

```
rsindexmod(dataN, N)
```

Formula Column

```
rsindexmod(Tn, N)
```

Definition

The modified version of RSI has a slightly different interpretation of the original RSI. It moves more smoothly than the original version. RSI Modified takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_rsindexmod : double;
var ind_rsi, ind_llb : variant;
    mysize : integer;
    stopprice, exitprice : double;
begin
   ind_rsi := itself.makeindicator ('myrsi', 'rsindexmod',
                                    ['1'], [param1.str]);
   ind_llb := itself.makeindicator ('myllb', 'llb',
                                     ['1.1'], [param2.str]);
   mysize := param3.int;
   stopprice := 50*trade.minticksize;
   exitprice := 50*trade.minticksize;

   if trade.openpositionflat then
   begin
      if (ind_rsi.value(0)<=30) and (ind_llb.value(0)=0) then
      begin
         trade.cancelallopenorders;
         trade.longatmarket (mysize, 'oversold long');
      end;
   end
   else
   begin
      if trade.openpositionlong and (trade.openordercount=0)
then
      begin
         trade.longexitstop (trade.openpositionentryprice-
stopprice,
                             mysize, otfGoodtilCancel, 'exit
stop');
         trade.longexitlimit
(trade.openpositionentryprice+exitprice,
                              mysize, otfGoodtilCancel, 'take
profit');
      end;
   end;

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_rsindexmod()
dim ind_rsi, ind_llb
   ind_rsi = itself.makeindicator ("myrsi", "rsindexmod", _
                                   Array("1"),
Array(param1.str))
   ind_llb = itself.makeindicator ("myllb", "llb", _
                                   Array("1.l"),
Array(param2.str))
   mysize = param3.int
   stopprice = 50*trade.minticksize
   exitprice = 50*trade.minticksize

   if trade.openpositionflat then
      if ind_rsi.value(0)<=30 and ind_llb.value(0)=0 then
         trade.cancelallopenorders
         trade.longatmarket mysize, "oversold long"
      end if
   else
      if trade.openpositionlong and trade.openordercount=0 then
         trade.longexitstop trade.openpositionentryprice-
stopprice, _
                            mysize, otfGoodtilCancel, "stop
lost"
         trade.longexitlimit
trade.openpositionentryprice+exitprice, _
                             mysize, otfGoodtilCancel, "take
profit"
      end if
   end if

   ex_vbs_rsindexmod = Data1.Value (0)
end function
```

Formula Language

```
$myticksize := 0.25;
$mysize := param3;
$exitprice := 50*$myticksize;
$stopprice := 50*$myticksize;

longatmarket (rsindexmod(data1, param1)<=30 and llb(data1.l,
param2)=0,
              $mysize, "oversold long");
longexitstop (openpositionlong>0,
openpositionaverageentryprice-$stopprice,
              $mysize, "stop lost");
longexitlimit (openpositionlong>0,
openpositionaverageentryprice+$exitprice,
               $mysize, "take profit");

plot1 := currentequity;
```

Formula Column

```
rsindexmod(data1, 9)
```

Relative Strength Index (rsindex)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'rsindex', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "rsindex", Array("N"),
Array("N"))
```

Formula Language

```
rsindex(dataN, "N")
```

Formula Column

```
rsindex(Tn, "N")
```

Definition

Relative Strength Index is a price momentum indicator which depends on a single data series. You can also apply RSI to an indicator to study their momentum behavior. RSI takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_rsindex : double;
const UBBAND = 0;
var ind_bband3, ind_rsi : variant;
    mysize : integer;
    stopprice, profitexit : double;
begin

   ind_bband3 := itself.makeindicator ('mybband', 'bbands3',
                                       ['1'], [param2.str,
'1']);
   ind_rsi := itself.makeindicator ('myrsi', 'rsindex', ['1'],
[param1.str]);

   if heap.size = 0 then
   begin
      heap.allocate(1);
      heap.value[UBBAND] := 0;
   end;
   mysize := param3.int;
   stopprice := 50*trade.minticksize;

   if trade.openpositionflat then
   begin
      if (data1.close [0]<ind_bband3.valueex[1, 0]) and
         (ind_rsi.value[0] < 20) then
      begin
         trade.cancelallopenorders;
         trade.longatmarket (mysize, 'long entry');
         heap.value[UBBAND] := ind_bband3.valueex[3, 0];
      end;
   end
   else
   begin
      if trade.openpositionlong and (trade.openordercount=0)
then
      begin
         trade.longexitstop(stopprice, mysize,
                           otfGoodtilCancel, 'long exit
stop');
         profitexit := heap.value[UBBAND];
         trade.longexitlimit(profitexit, mysize,
                            otfGoodtilCancel, 'long exit take
profit');
      end;
   end;

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_rsindex()
const UBBAND = 0
dim ind_bband3, ind_rsi

   ind_bband3 = itself.makeindicator ("mybband", "bbands3", _
                                      Array("1"),
Array(param2.str, "1"))
   ind_rsi = itself.makeindicator ("myrsi", "rsindex", _
                                   Array("1"),
Array(param1.str))

   if heap.size = 0 then
      heap.allocate(1)
      heap.value(UBBAND) = 0
   end if

   mysize = param3.int
   stopprice = 50*trade.minticksize

   if trade.openpositionflat = true then
      if (data1.close (0) < ind_bband3.valueex(3, 1)) and _
         (ind_rsi.value (0) < 20) then
         trade.cancelallopenorders
         trade.longatmarket mysize, "long entry"
         heap.value(UBBAND) = ind_bband3.valueex(3, 0)
      end if
   else
      if trade.openpositionlong and trade.openordercount=0 then
         trade.longexitstop stopprice, mysize, _
                            otfGoodtilCancel, "long exit stop"
         profitexit = heap.value(UBBAND)
         trade.longexitlimit profitexit, mysize, _
                        otfGoodtilCancel, "long exit take
profit "
      end if
   end if

   ex_vbs_rsindex = trade.currentequity
end function
```

Formula Language

```
$mticksize := 0.01;
$mysize := param3;
$stopprice := 50*$mticksize;
longatmarket(c(0)<bbands3.plot1(data1, param2, 1) and
             rsindex(data1, param1)<20,
             $mysize, "long entry");
longexitstop(openpositionlong>0, $stopprice, $mysize, "long
exit stop");

longexitlimit(openpositionlong>0, bbands3.plot3(data1, param2,
1),
                               $mysize, "long exit profit");
plot1 := currentequity;
```

Formula Column

```
bbands3.plot1(m1, 9)
```

Relative Strength Multiple Data (rs_multi)

This is a power indicator that help users apply relative strength on 10 data series in the chart with one indicator. For detail usage, please refer to ***Relative Strength Multiple Data*** (on page 1335).

Sample Daily (SamplingDaily)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'samplingdaily', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "samplingdaily", Array("N"),
Array(""))
```

Formula Language

```
samplingdaily(dataN)
```

Formula Column

```
samplingdaily(Tn)
```

Definition

Sampling Daily returns the first value of each day from an intraday data series (Link 1) and returns invalid for the all the rest of the data points. It requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_samplingdaily : double;
var ind_ma, ind_samplingd : variant;
begin
   ind_samplingd := itself.makeindicator ('mysd',
'samplingdaily', ['1'], ['']);
   ind_ma := itself.makeindicator ('myma', 'average', ['1'],
[param1.str]);
   if heap.size = 0 then
   begin
      heap.allocate(1);
      heap.value [0] := 0;
   end;
   if ind_samplingd.valid[0] then
      heap.value [0] := ind_samplingd.value[0];

   if not ind_ma.valid [0] or (heap.value [0]=0) then
   begin
      itself.success := false;
      exit;
   end;
   result := ind_ma.value[0]-heap.value [0];
end;
```

VBScript

```
function ex_vbs_samplingdaily()
dim ind_ma, ind_samplingd

   ind_samplingd = itself.makeindicator ("mysd",
"samplingdaily", _
                                          Array("1"), Array(""))
   ind_ma = itself.makeindicator ("myma", "average", _
                                 Array("1"),
Array(param1.str))

   if heap.size = 0 then
      heap.allocate(1)
      heap.value (0) = 0
   end if

   if ind_samplingd.valid(0) then
      heap.value(0) = ind_samplingd.value(0)
   end if

   if not ind_ma.valid(0) or (heap.value(0)=0) then
      itself.success = false
      exit function
   end if

   ex_vbs_samplingdaily = ind_ma.value(0)-heap.value(0)
end function
```

Formula Language

```
plot1 := average(data1, param1)-samplingdaily(data1);
```

Formula Column

```
average(m5,20)-samplingdaily(m5)
```

Sample Monthly (SamplingMonthly)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'samplingmonthly', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "samplingmonthly", Array("N"),
Array(""))
```

Formula Language

```
samplingmonthly(dataN)
```

Formula Column

```
samplingmonthly(Tn)
```

Definition

Sampling Monthly returns the first value of each day from lower time frame data series (Link 1) and returns invalid for the all the rest of the data points. It requires one series (Link 1).

Delphi Script

```
function ex_del_samplingmonthly : double;
var ind_wma, ind_samplingm : variant;
begin

   ind_samplingm := itself.makeindicator ('mysm',
'samplingmonthly',
                                                    ['1'],
['']);
   ind_wma := itself.makeindicator ('mywma', 'waverage', ['1'],
[param1.str]);

   if heap.size = 0 then
   begin
      heap.allocate(1);
      heap.value [0] := 0;
   end;

   if ind_samplingm.valid [0] then
      heap.value [0] := ind_samplingm.value [0];

   if not ind_wma.valid [0] or (heap.value[0]=0) then
   begin
```

```
      itself.success := false;
      exit;
   end;

   result := ind_wma.value [0]-heap.value[0];
end;
```

VBScript

```
function ex_vbs_samplingmonthly()
dim ind_wma, ind_samplingm

   ind_samplingm = itself.makeindicator ("mysm",
"samplingmonthly", _
                                         Array("1"), Array(""))
   ind_wma = itself.makeindicator ("mywma", "waverage", _
                                   Array("1"),
Array(param1.str))

   if heap.size = 0 then
      heap.allocate(1)
      heap.value(0) = 0
   end if

   if ind_samplingm.valid(0) then
      heap.value(0) = ind_samplingm.value(0)
   end if

   if not ind_wma.valid(0) or (heap.value(0)=0) then
      itself.success = false
      exit function
   end if

   ex_vbs_samplingmonthly = ind_wma.value(0)-heap.value(0)
end function
```

Formula Language

```
plot1 := waverage(data1, param1)-samplingmonthly(data1);
```

Formula Column

```
waverage(m1, 10)-samplingmonthly(m1)
```

Sample Weekly (SamplingWeekly)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'samplingweekly', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "samplingweekly", Array("N"),
Array(""))
```

Formula Language

```
samplingweekly(dataN)
```

Formula Column

```
samplingweekly(Tn)
```

Definition

Sampling Weekly returns the first value of each week from lower time frame data series (Link 1) and returns invalid for the all the rest of the data points. It requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_samplingweekly : double;
var ind_vma, ind_samplingw : variant;
begin
   ind_samplingw := itself.makeindicator ('mysw',
'samplingweekly',
                                             ['1'], ['']);
   ind_vma := itself.makeindicator ('myvma', 'vaverage', ['1'],
[param1.str]);

   if heap.size = 0 then
   begin
      heap.allocate(1);
      heap.value [0] := 0;
   end;

   if ind_samplingw.valid [0] then
      heap.value [0] := ind_samplingw.value[0];
   if (heap.value [0]=0) or not ind_vma.valid[0] then
   begin
      itself.success := false;
      exit;
   end;

   result := ind_vma.value[0]-heap.value[0];
end;
```

VBScript

```
function ex_vbs_samplingweekly()
dim ind_vma, ind_samplingw

   ind_samplingw = itself.makeindicator ("myws",
"samplingweekly", _
                                         Array("1"), Array(""))
   ind_vma = itself.makeindicator ("myvma", "vaverage", _
                                   Array("1"),
Array(param1.str))

   if heap.size = 0 then
      heap.allocate(1)
      heap.value(0) = 0
   end if

   if ind_samplingw.valid(0) then
      heap.value (0) = ind_samplingw.value(0)
   end if

   if heap.value(0)=0 or not ind_vma.valid(0) then
      itself.success = false
      exit function
   end if

   ex_vbs_samplingweekly = ind_vma.value(0)-heap.value(0)
end function
```

Formula Language

```
plot1 := vaverage(data1, param1)-samplingweekly(data1);
```

Formula Column

```
vaverage(D, 10)-samplingweekly(D)
```

Scan Sample Daily Report (scan_sample_daily)

This indicator is specifically for use in pattern scanner windows. It prints to a report all indicators calculated result on every symbol in a scan. This is a legacy indicator, because pattern scanner can display individual indicator calculation results.

Sign (Sign)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'sign', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "sgin", Array("N"), Array(""))
```

Formula Language

```
sgin(dataN)
```

Formula Column

```
sign(Tn)
```

Definition

Returns 1, 0, -1 based on the sign of Link 1. If Link 1 is positive then Sign returns 1. If Link 1 is 0, then Sign returns 0. If Link 1 is negative, then Sign returns -1.

Examples

Delphi Script

```
function ex_del_sign : double;
var ind_macd, ind_sign : variant;
begin
   ind_macd := itself.makeindicator ('mymacd', 'qc_MACD',
['1'],
                                      ['12', '26', 'No', '9']);
   ind_sign := itself.makeindicator ('mysign', 'sign',
                                    ['mymacd.plot3'], ['']);
   if not ind_macd.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;
   itself.plot [1] := ind_macd.valueex [3, 0];
   itself.plot [2] := 0;
   if ind_sign.value [0] > 0 then
      itself.plot [3] := param1.int
   else
      itself.plot [3] := param2.int;
end;
```

VBScript

```
function ex_vbs_sign()
dim ind_macd, ind_sign

   ind_macd = itself.makeindicator ("mymacd", "qc_macd",
Array("1"), _
                                   Array("12", "26", "NO",
"9"))
   ind_sign = itself.makeindicator ("mysign", "sign", _
                                    Array("mymacd.plot3"),
Array(""))

   if not ind_macd.valid(0) then
      itself.successall = false
      exit function
   end if

   itself.plot(1) = ind_macd.valueex (3, 0)
   itself.plot(2) = 0

   if ind_sign.value(0) > 0 then
      itself.plot(3) = param1.int
   else
      itself.plot(3) = param2.int
   end if
end function
```

Formula Language

```
plot1 := qc_macd.plot3(data1, 12, 26, "NO", 9);
plot2 := 0;
plot3 := if(sign(qc_macd.plot3(data1, 12, 26, "NO", 9))>0,
param1, param2);
```

Formula Column

```
sign(m1)
```

Signal (Signal)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'signal', ['N','N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "signal", Array("N","N"),
Array("N"))
```

Formula Language

```
signal(dataN, dataN, N)
```

Formula Column

```
N/A
```

Definition

Signal is a conditional indicator: if Link 1 equals 0, then Signal does not return a value; if Link 1 is not 0, Signal returns Link 2 + Offset.

Examples

Delphi Script

```
function ex_del_signal : double;
var ind_signalup, ind_signaldown : varint;
begin
   itself.makeindicator ('mymas', 'average', ['1'],
[param1.str]);
   itself.makeindicator ('mymal', 'average', ['1'],
[param2.str]);

   itself.makeindicator ('myxup', 'xabove', ['mymas', 'mymal'],
['']);
   itself.makeindicator ('myxdown', 'xbelow', ['mymas',
'mymal'], ['']);

   ind_signalup := itself.makeindicator ('mysignu', 'signal',
                        ['myxup', '1.h'], ['1']);
   ind_signaldown := itself.makeindicator ('mysignd', 'signal',
                        ['myxdown', '1.l'], ['-1']);
   itself.plot [1] := ind_signalup.value [0];
   itself.plot [2] := ind_signaldown.value [0];

   if not ind_signalup.valid [0] then
      itself.successex [1] := false;

   if not ind_signaldown.valid [0] then
      itself.successex [2] := false;

end;
```

VBScript

```
function ex_vbs_signal()
dim ind_signalup, ind_signaldown

   itself.makeindicator "mymas", "average", _
                        Array("1"), Array(param1.str)
   itself.makeindicator "mymal", "average", _
                        Array("1"), Array(param2.str)
   itself.makeindicator "crossup", "xabove", _
                        Array("mymas", "mymal"), Array("")
   itself.makeindicator "crossdown", "xbelow", _
                        Array("mymas", "mymal"), Array("")

   ind_signalup = itself.makeindicator ("myup", "signal", _
                         Array("crossup", "1.h"), Array("1"))
   ind_signaldown = itself.makeindicator ("mydown", "signal", _
                         Array("crossdown", "1.l"), Array("-
1"))

   itself.plot (1) = ind_signalup.value (0)
   itself.plot (2) = ind_signaldown.value (0)

   if not ind_signalup.valid (0) then
      itself.successex (1) = false
   end if

   if not ind_signaldown.valid (0) then
      itself.successex (2) = false
   end if
end function
```

Formula Language

```
plot1 := signal(xabove(average(data1, param1), average(data1,
param2)), h, 1);
plot2 := signal(xbelow(average(data1, param1), average(data1,
param2)), l, -1);
success1 := if(xabove(average(data1, param1), average(data1,
param2))>0, 1, 0);
success2 := if(xbelow(average(data1, param1), average(data1,
param2))>0, 1, 0);
```

Formula Column

```
N/A
```

Simple Moving Average (average)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'average', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "average", Array("N"),
Array("N"))
```

Formula Language

```
average(dataN, N);
```

Formula Column

```
average(Tn, N)
```

Definition

Returns the simple moving average over Period. Simple Moving Average requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_average : double;
var xo, xu : variant;
begin
   Itself.makeindicator ('fma', 'average', ['1'],
[param1.str]);
   Itself.makeindicator ('sma', 'average', ['1'],
[param2.str]);
   xo := Itself.Makeindicator('xo1', 'xabove', ['fma', 'sma'],
['']);
   xu := Itself.Makeindicator('xu1', 'xbelow', ['fma', 'sma'],
['']);
   if xo.value[0] > 0 then
      trade.longatmarket (100, 'Enter long')
   else if xu.value[0] > 0 then
      trade.shortatmarket (100, 'Enter short');
   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_average()
dim xu, xo
    Itself.Makeindicator "fma", "average", Array("1"),
Array(param1.str)
    Itself.Makeindicator "sma", "average", Array("1"),
Array(param2.str)
    xo = Itself.MakeIndicator("xo1", "xabove", Array("fma",
"sma"), Array(""))
    xu = Itself.MakeIndicator("xu1", "xbelow", Array("fma",
"sma"), Array(""))
    if xo.value(0) > 0 then
       trade.longatmarket 100, "Enter long"
    elseif xu.value(0) > 0 then
       trade.shortatmarket 100, "Enter short"
    end if
    ex_vbs_average = trade.currentequity
end function
```

Formula Language

```
LongAtMarket(xabove(average(data1, param1),average(data1,
param2)), 100);
ShortAtMarket(xbelow(average(data1, param1),average(data1,
param2)), 100);
plot1 := currentequity;
```

Formula Column

```
average(D1, 30)
```

Standard Deviation (stddev)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'stddev', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "stddev", Array("N"),
Array("N"))
```

Formula Language

```
stddev(dataN, N)
```

Formula Column

```
stddev(Tn, N)
```

Definition

Returns the Standard Deviation assuming complete space is given. Standard Deviation takes one parameter Period and require one series (Link 1).

Examples

Delphi Script

```
function ex_del_stddev : double;
var ind_ma, ind_stddev : variant;
begin
   ind_ma := itself.makeindicator ('myma', 'average', ['1'],
[param2.str]);
   ind_stddev := itself.makeindicator ('mysd', 'stddev',
                                        ['myma'],
[param1.str]);

   if not (ind_ma.valid [0] and ind_stddev.valid [0]) then
   begin
      itself.successall := false;
      exit;
   end;

   itself.plot [1] := ind_ma.value [0];
   itself.plot [2] := ind_ma.value [0]+ind_stddev.value [0];
   itself.plot [3] := ind_ma.value [0]-ind_stddev.value [0];
end;
```

VBScript

```
function ex_vbs_stddev()
dim ind_ma, ind_stddev
   ind_ma = itself.makeindicator ("myma", "average", _
                                  Array("1"),
Array(param2.str))
   ind_stddev = itself.makeindicator ("mysd", "stddev", _
                                  Array("myma"),
Array(param1.str))

   if not (ind_ma.valid (0) and ind_stddev.valid (0)) then
      itself.successall = false
      exit function
   end if

   itself.plot (1) = ind_ma.value (0)
   itself.plot (2) = ind_ma.value (0) + ind_stddev.value (0)
   itself.plot (3) = ind_ma.value (0) - ind_stddev.value (0)
end function
```

Formula Language

```
$myma := average(data1, param2);
$mysd := stddev(average(data1, param2), param1);
plot1 := $myma;
plot2 := $myma+$mysd;
plot3 := $myma-$mysd;
```

Formula Column

```
stddev(m5, 10)
```

Standard Deviation Variable Length (stddevvar)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'stddevvar', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "stddevvar", Array("N"),
Array(""))
```

Formula Language

```
stddevvar(dataN)
```

Formula Column

```
stddevvar(Tn)
```

Definition

Returns the Standard Deviation. The length of the calculation is all the data available in the series excluding invalid points.

Examples

Delphi Script

```
function ex_del_stddevvar : double;
var ind_ma, ind_sd : variant;
begin
   ind_ma := itself.makeindicator ('myma', 'average', ['1'],
[param1.str]);
   ind_sd := itself.makeindicator ('mysd', 'stddevvar',
['myma'], ['']);
   if not (ind_ma.valid [0] and ind_sd.valid [0]) then
   begin
      itself.successall := false;
      exit;
   end;
   itself.plot [1] := ind_ma.value [0];
   itself.plot [2] := ind_ma.value [0]+ind_sd.value [0];
   itself.plot [3] := ind_ma.value [0]-ind_sd.value [0];
end;
```

VBScript

```
function ex_vbs_stddevvar()
dim ind_ma, ind_sd
   ind_ma = itself.makeindicator ("myma", "average", _
                                  Array("1"),
Array(param1.str))
   ind_sd = itself.makeindicator ("mysd", "stddevvar",
Array("myma"), Array(""))
   if not (ind_ma.valid (0) and ind_sd.valid (0)) then
      itself.successall = false
      exit function
   end if
   itself.plot (1) = ind_ma.value (0)
   itself.plot (2) = ind_ma.value (0)+ ind_sd.value (0)
   itself.plot (3) = ind_ma.value (0)- ind_sd.value (0)
end function
```

Formula Language

```
$myma := average(data1, param1);
$mysd := stddevvar(average(data1, param1));
plot1 := $myma;
plot2 := $myma+$mysd;
plot3 := $myma-$mysd;
```

Formula Column

```
stddevvar(m5)
```

Stochastic FastD (fastd)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'fastd', ['N'], ['N','N']);
```

VBScript

```
Itself.MakeIndicator("string", "fastd", Array("N"),
Array("N","N"))
```

Formula Language

```
fastd(dataN, N, N)
```

Formula Column

```
sign(Tn, N, N)
```

Definition

Stochastic FastD indicator of a specified period. Stochastic FastD requires one specified series (Link 1).

Examples

Delphi Script

```
function ex_del_fastd : double;
var ind_fastd, ind_cdh, ind_cdl : variant;
    ind_macd, ind_macdsignal, ind_xbelow : variant;
    oversold_level : double;
    mysize : integer;
begin
   oversold_level := 30;
   mysize := param6.int;

   ind_fastd := itself.makeindicator ('myfastd', 'fastd',
['1'],
                                      [param1.str,
param2.str]);
   ind_macd := itself.makeindicator ('mymacd', 'MACD', ['1'],
                             ['simple', param3.str, 'simple',
param4.str]);
   ind_macdsignal := itself.makeindicator ('mysignal',
'average',
                                       ['mymacd'],
[param5.str]);
   ind_xbelow := itself.makeindicator ('myxbelow', 'xbelow',
                                         ['mymacd', 'mysignal'],
['']);
   ind_cdh := itself.makeindicator ('mycdh', 'currdhigh',
['1.h'], ['']);
   ind_cdl := itself.makeindicator ('mycdl', 'currdlow',
['1.l'], ['']);

   if trade.openpositionflat then
   begin
      if trade.openordercount > 0 then
      begin
         if (trade.openorders [trade.openordercount-
1].OrderType =
              otLongStop) and
             (trade.openorders [trade.openordercount-1].price >
              data1.high [0]) then
         begin
            trade.cancelorder(
                  trade.openorders [trade.openordercount-
1].Orderid);
            trade.longstop (data1.high [0], mysize,
otfGoodtilCancel,
                             'rebuystop' +
ntlib.integer2str(data1.barsnum[0]));
         end;

         if trade.openorders [trade.openordercount-1].OrderType
<>
            otLongStop then
            trade.cancelorder(
                  trade.openorders [trade.openordercount-
1].Orderid);
      end
      else
         if (ind_macdsignal.value [0] > ind_macd.value [0]) and
            (ind_fastd.value [0] < oversold_level) then
             trade.longstop (ind_cdh.value [0], mysize,
otfGoodtilCancel,
                          'buystop' +
ntlib.integer2str(data1.barsnum [0]));
   end
   else
   begin
      if trade.openordercount = 0 then
         trade.longexitstop (ind_cdl.value [0], mysize,
otfGoodtilCancel,
                             'stoplost' +
ntlib.integer2str(data1.barsnum [0]));

      if ind_xbelow.value [0] > 0 then
      begin
         trade.cancelallopenorders;
         trade.longexitatmarket (mysize, 'exitlong');
      end;
   end;
   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_fastd()
dim ind_fastd, ind_cdh, ind_cdl, ind_macd, ind_macdsignal,
ind_xbelow

   oversold_level = 30
   mysize = param6.int

   ind_fastd = itself.makeindicator ("myfastd", "fastd",
Array("1"), _
                                     Array(param1.str,
param2.str))
   ind_macd = itself.makeindicator ("mymacd", "MACD",
Array("1"), _
                           Array("simple", param3.str,
"simple", param4.str))
   ind_macdsignal = itself.makeindicator ("mysignal",
"average", _
                           Array("mymacd"), Array(param5.str))
   ind_xbelow = itself.makeindicator ("myxbelow", "xbelow", _
                           Array("mymacd", "mysignal"),
Array(""))
   ind_cdh = itself.makeindicator ("mycdh", "currdhigh", _
                           Array("1.h"), Array(""))
   ind_cdl = itself.makeindicator ("mycdl", "currdlow", _
                           Array("1,l"), Array(""))

   if trade.openpositionflat then
      if trade.openordercount > 0 then
         if (trade.openorders (trade.openordercount-
1).OrderType = _
             otLongStop) and _
            (trade.openorders (trade.openordercount-1).price >
_
             data1.high (0)) then
             trade.cancelorder ( _
                   trade.openorders (trade.openordercount-
1).Orderid)
             trade.longstop data1.high (0), mysize,
otfGoodtilCancel, _
                            "rebuystop" +
ntlib.integer2str(data1.barsnum (0))
         end if

         if trade.openorders (trade.openordercount-1).OrderType
<> _
            otLongStop then
            trade.cancelorder ( _
                  trade.openorders (trade.openordercount-
1).orderid)
         end if
      else
         if (ind_macdsignal.value (0) > ind_macd.value (0)) and
_
            (ind_fastd.value (0) < oversold_level) then
            trade.longstop ind_cdh.value (0), mysize,
otfGoodtilCancel, _
                           "buystop" +
ntlib.integer2str(data1.barsnum (0))
         end if
      end if
   else
      if trade.openordercount = 0 then
         trade.longexitstop ind_cdl.value (0), mysize,
otfGoodtilCancel, _
                            "stoplost" +
ntlib.integer2str(data1.barsnum(0))
      end if

      if ind_xbelow.value (0) > 0 then
         trade.cancelallopenorders
         trade.longexitatmarket mysize, "exitlong"
      end if
   end if
   ex_vbs_fastd = trade.currentequity
end function
```

Formula Language

```
$oversold_level := 30;
$mysize := param6;
$mymacd := MACD(data1, "simple", param3, "simple", param4);
$mysignal := average(MACD(data1, "simple", param3, "simple",
param4), param5);
$myxbelow := xbelow(MACD(data1, "simple", param3, "simple",
param4),
                    average(MACD(data1, "simple", param3,
"simple", param4),
                     param5));
longstop ($mysignal > $mymacd and
          fastd(data1, param1, param2) < $oversold_level,
          currdhigh(data1.h), $mysize, "buystop");
longexitstop (openpositionlong > 0, currdlow(data1.l),
              $mysize, "exitstop");
longexitatmarket (openpositionlong > 0 and $myxbelow > 0,
                  mysize, "exitlong");
plot1 := currentequity;
```

Formula Column

```
fastd(m1, 5, 3)
```

Stochastic FastK (fastk)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'fastk', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "fastk", Array("N"), Array("N"))
```

Formula Language

```
fastk(dataN, N)
```

Formula Column

```
fastk(Tn, N)
```

Definition

Stochastic FastK indicator of a specified period. Stochastic FastK requires one specified series (Link 1).

Examples

Delphi Script

```
function ex_del_fastk : double;
var ind_stochrsi : variant;
begin
   itself.makeindicator ('myrsi', 'rsindex', ['1'],
[param2.str]);

   ind_stochrsi := itself.makeindicator ('mystoch', 'fastk',
                                          ['myrsi'],
[param1.str]);

   if not ind_stochrsi.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := ind_stochrsi.value [0];
end;
```

VBScript

```
function ex_vbs_fastk()
dim ind_stochrsi

   itself.makeindicator "myrsi", "rsindex", Array("1"),
Array(param2.str)

   ind_stochrsi = itself.makeindicator ("mysotch", "fastk", _
                                         Array("myrsi"),
Array(param1.str))

   if not ind_stochrsi.valid (0) then
      itself.success = false
      exit function
   end if

   ex_vbs_fastk = ind_stochrsi.Value (0)
end function
```

Formula Language

```
plot1 := fastk(rsindex(data1, param2), param1);
```

Formula Column

```
fastk(m1, 5)
```

Stochastic SlowD (slowd)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'slowd', ['N'], ['N','N']);
```

VBScript

```
Itself.MakeIndicator("string", "slowd", Array("N"),
Array("N","N"))
```

Formula Language

```
slowd(dataN, N, N)
```

Formula Column

```
slowd(Tn, N, N)
```

Definition

Stochastic SlowD indicator of a specified period. Stochastic SlowD requires one specified series (Link 1).

Examples

Delphi Script

```
function ex_del_slowd : double;
var ind_stoch, ind_signal : varint;
begin
   itself.makeindicator ('myd', 'slowd', ['1'], [param1.str,
param2.str]);
   itself.makeindicator ('myhhd', 'hhv', ['myd'],
[param3.str]);
   itself.makeindicator ('mylld', 'llv', ['myd'],
[param3.str]);
   itself.makeindicator ('mynorm', 'diff', ['1', 'mylld'],
['']);
   itself.makeindicator ('mydenorm', 'diff', ['myhhd',
'mylld'], ['']);

   ind_stoch := itself.makeindicator ('mystoch', 'div',
                                       ['mynorm', 'mydenorm'],
['']);
   ind_signal := itself.makeindicator ('mysign', 'average',
                                       ['mystoch'],
[param4.str]);

   if not ind_stoch.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   itself.plot [1] := ind_stoch.value [0];
   itself.plot [2] := ind_signal.value [0];
end;
```

VBScript

```
function ex_vbs_slowd()
dim ind_stoch, ind_signal

   itself.makeindicator "myd", "slowd", Array("1"), _
                        Array(param1.str, param2.str)
   itself.makeindicator "myhhd", "hhv", Array("myd"),
Array(param3.str)
   itself.makeindicator "mylld", "llv", Array("myd"),
Array(param3.str)
   itself.makeindicator "mynorm", "diff", Array("1", "mylld"),
Array("")
   itself.makeindicator "mydenorm", "diff", Array("myhhd",
"mylld"), Array("")

   ind_stoch = itself.makeindicator ("mystoch", "div", _
                                     Array("mynorm",
"mydenorm"), Array(""))
   ind_signal = itself.makeindicator ("mysign", "average", _
                                      Array("mystoch"),
Array(param4.str))

   if not ind_stoch.valid (0) then
      itself.successall = false
      exit function
   end if

   itself.plot (1) = ind_stoch.value (0)
   itself.plot (2) = ind_signal.value (0)
end function
```

Formula Language

```
$denorm := (hhv(slowd(data1, param1, param2), param3)-
            llv(slowd(data1, param1, param2), param3));
mystoch := if ($denorm = 0, 0,
               (c-llv(slowd(data1, param1, param2),
param3))/$denorm);
plot1 := mystoch;
plot2 := average(mystoch, param4);
success1 := if ($denorm = 0, 0, 1);
success2 := if ($denorm = 0, 0, 1);
```

Formula Column

```
slowd(m1, 3, 5)
```

Stochastic SlowK (slowk)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'slowk', ['N'], ['N','N']);
```

VBScript

```
Itself.MakeIndicator("string", "slowk", Array("N"),
Array("N","N"))
```

Formula Language

```
slowk(dataN, N, N)
```

Formula Column

```
slowk(Tn, N, N)
```

Definition

Stochastic SlowK indicator of a specified period. Stochastic SlowK requires one specified series (Link 1).

Examples

Delphi Script

```
function ex_del_slowk  : double;
var ind_hhk, ind_llk : variant;
begin
   itself.makeindicator ('myk', 'slowk', ['1'], [param1.str]);
   ind_hhk := itself.makeindicator ('myhhk', 'hhv', ['myk'],
[param2.str]);
   ind_llk := itself.makeindicator ('myhhl', 'llv', ['myk'],
[param3.str]);

   if not (ind_hhk.valid [0] and ind_llk.valid [0]) then
   begin
      itself.success := false;
      exit;
   end;

   if (ind_hhk.value [0]-ind_llk.value [0]) = 0 then
   begin
      itself.success := false;
      exit;
   end;

   result := (Data1.Value [0]-ind_llk.value [0])/
             (ind_hhk.value [0]-ind_llk.value [0]) ;
end;
```

VBScript

```
function ex_vbs_slowk()
dim ind_hhk, ind_llk

   itself.makeindicator "myk", "slowk", Array("1"), _
                              Array(param1.str, param2.str)
   ind_hhk = itself.makeindicator ("myhhv", "hhv",
Array("myk"), _
                                  Array(param3.str))
   ind_llk = itself.makeindicator ("myllv", "llv",
Array("myk"), _
                                  Array(param4.str))

   if not (ind_hhk.valid (0) and ind_llk.valid (0)) then
      itself.success = false
      exit function
   end if

   if (ind_hhk.value (0)-ind_llk.value (0)) = 0 then
       itself.success = false
       exit function
   end if

   ex_vbs_slowk = (Data1.Value (0) - ind_llk.value (0))/ _
                  (ind_hhk.value (0) - ind_llk.value (0))
end function
```

Formula Language

```
$denorm :=hhv(slowk(data1, param1, param2), param3)-
          llv(slowk(data1, param1,param2), param4);

plot1 := if ($denorm = 0, 0,
             (c-llv(slowk(data1, param1, param2),
param3))/$denorm);
success1 := if ($denorm = 0, 0, 1);
```

Formula Column

```
slowk(m1, 3, 5)
```

Sub Series (subseries)

Sub Series returns a meta data series having only bars that meet the filter parameter condition. Sub Series requires one series (Link 1).

For detail usage, please refer to help file section: ***Tutorial: Visualize Trading Idea with Sub Series Indicator and Meta Plot Style*** (on page 231).

Subtration (diff)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'diff', ['N', 'N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "diff", Array("N", "N"),
Array(""))
```

Formula Language

```
diff(dataN, DataN)
```

Formula Column

```
N/A
```

Definition

Subtracts one series from another (Link 1 - Link 2).

Examples

Delphi Script

```
function ex_del_diff : double;
var ind_macd, ind_signal : variant;
begin
   itself.makeindicator ('mysma', 'average', ['1'],
[param1.str]);
   itself.makeindicator ('mylma', 'average', ['1'],
[param2.str]);

   ind_macd := itself.makeindicator ('mymacd', 'diff',
                                      ['mysma', 'mylma'],
['']);
   ind_signal := itself.makeindicator ('mysign', 'average',
                                        ['mymacd'],
[param3.str]);

   if not ind_macd.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;
   itself.plot [1] := ind_macd.value [0];
   itself.plot [2] := ind_signal.value [0];
   itself.plot [3] := ind_macd.value [0] - ind_signal.value
[0];

end;
```

VBScript

```
function ex_vbs_diff()
dim ind_macd, ind_signal

   itself.makeindicator "mysma", "average", Array("1"),
Array(param1.str)
   itself.makeindicator "mylma", "average", Array("1"),
Array(param2.str)

   ind_macd = itself.makeindicator ("mymacd", "diff", _
                                    Array("mysma", "mylma"),
Array(""))
   ind_signal = itself.makeindicator ("mysign", "average", _
                                      Array("mymacd"),
Array(param3.str))

   if not ind_macd.valid (0) then
      itself.successall = false
      exit function
   end if

   itself.plot (1) = ind_macd.value (0)
   itself.plot (2) = ind_signal.value (0)
   itself.plot (3) = ind_macd.value (0) - ind_signal.value (0)
end function
```

Formula Language

```
plot1 := diff(average(data1, param1), average(data1, param2));
plot2 := average(plot1, param3);
plot3 := plot1 - plot2;
```

Formula Column

N/A

Summation (summation)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'summation', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "summation", Array("N"),
Array("N"))
```

Formula Language

```
summation(dataN, N)
```

Formula Column

```
summation(Tn, N)
```

Definition

Returns the Summation over Period. Summation requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_summation : double;
var ind_rsi, ind_sumrsi : variant;
begin
   ind_rsi := itself.makeindicator ('myrsi', 'rsindex',
                                    ['1'], [param1.str]);
   ind_sumrsi := itself.makeindicator ('mysumrsi', 'summation',
                                    ['myrsi'], [param2.str]);

   if not ind_rsi.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   itself.plot [1] := ind_rsi.value [0];
   itself.plot [2] := ind_sumrsi.value [0]/param2.int;
end;
```

VBScript

```
function ex_vbs_summation ()
dim ind_rsi, ind_sumrsi

    ind_rsi = itself.makeindicator ("myrsi", "rsindex", _
                                             Array("1"),
Array(param1.str))
    ind_sumrsi = itself.makeindicator ("mysumrsi", "summation",
_
                                      Array("myrsi"),
Array(param2.str))

    if not ind_rsi.valid (0) then
       itself.successall = false
       exit function
    end if

    itself.plot (1) = ind_rsi.value (0)
    itself.plot (2) = ind_sumrsi.value (0)/ param2.int
end function
```

Formula Language

```
plot1 := rsindex(data1, param1);
plot2 := summation(rsindex(data1, param1), param2)/ param2;
```

Formula Column

```
summation(m1, 10)
```

Sum XY (sumxy)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'sumxy', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "sumxy", Array("N"), Array(""))
```

Formula Language

```
sumxy(dataN, N)
```

Formula Column

```
sumxy(Tn, N)
```

Definition

Returns the sum of x multiply by y where x is 1 to N and y is the data series value. Sum XY is useful for the calculation of linear regression and other similar type of mathematical formulas.

Examples

Delphi Script

```
function ex_del_sumxy : double;
var ind_sumxy : variant;
begin

   ind_sumxy := itself.makeindicator ('mysumxy', 'sumxy',
                                      ['1'], [param1.str]);

   if param1.int = 0 then
   begin
      itself.success := false;
      exit;
   end;
   if not ind_sumxy.valid [0] then
   begin
     itself.success := false;
     exit;
   end;
   result := ind_sumxy.value [0]/param1.int;
end;
```

VBScript

```
function ex_vbs_sumxy()
dim ind_sumxy
   ind_sumxy = itself.makeindicator ("mysumxy", "sumxy", _
                                     Array("1"),
Array(param1.str))

   if param1.int = 0 then
      itself.success = false
      exit function
   end if
   if not ind_sumxy.valid (0) then
      itself.success = false
      exit function
   end if
   ex_vbs_sumxy = ind_sumxy.value (0)/param1.int
end function
```

Formula Language

```
plot1 := if(param1 = 0, 0, sumxy(data1, param1)/param1);
success1 := if (param1 = 0, 0, 1);
```

Formula Column

```
sumxy(m1, 5)
```

Swing High Bar(swinghighbar)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'swinghighbar', ['N'], ['N',
'N', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "swinghighbar", Array("N"),
Array("N", "N", "N"))
```

Formula Language

```
swinghighbar(dataN, N, N, N)
```

Formula Column

```
swinghighbar(Tn, N, N, N)
```

Definition

Swing High Bar returns the bars ago of the Nth occurence (Occur) of the swing high point of Strength bars over Length period.

Examples

Delphi Script

```
function ex_del_swinghighbar : double;
var ind_swinghighbar : variant;
begin
   ind_swinghighbar := itself.makeindicator ('myswhigh',
'swinghighbar',
                              ['1.h'], [param1.str, param2.str,
param3.str]);


   if ind_swinghighbar.value [0] = param2.int then
      result := Data1.high [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_swinghighbar()
```

```
dim ind_swinghighbar

   ind_swinghighbar = itself.makeindicator ("myswhigh",
"swinghighbar", _
                      ArraY("1.h"), Array(param1.str,
param2.str, param3.str))

   if ind_swinghighbar.value (0) = param2.int then
      ex_vbs_swinghighbar = Data1.high (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := if(swinghighbar(data1.h, param1, param2, param3) =
param2, h, 0);
success1 := if(swinghighbar(data1.h, param1, param2, param3) =
param2, 1, 0);
```

Formula Column

```
swinghighbar(m15, 1, 3, 50)
```

Swing High (swinghigh)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'swinghigh', ['N'], ['N', 'N',
'N']);
```

VBScript

```
Itself.MakeIndicator("string", "swinghigh", Array("N"),
Array("N", "N", "N"))
```

Formula Language

```
swinghigh(dataN, N, N, N)
```

Formula Column

```
swinghigh(Tn, N, N, N)
```

Definition

Swing High returns the Nth occurence (Occur) of the swing high point of Strength bars over Length period.

Examples

Delphi Script

```
function ex_del_swinghigh : double;
var ind_ValD, ind_swingslowk : variant;
    mysize : double;
begin
   mysize := param5.int;
   itself.makeindicator ('myslowk', 'slowk', ['1'],
[param1.str, param2.str]);
   ind_ValD := itself.makeindicator ('myavgslowk', 'average',
                                            ['myslowk'],
[param3.str]);
   ind_swingslowk := itself.makeindicator ('myswing',
'swinghigh',
                                             ['myslowk'], ['1',
'2', '50']);

   if trade.openpositionflat then
   begin
      trade.cancelallopenorders;
      if (ind_ValD.value [0] < ind_ValD.value [1]) and
         (ind_ValD.value [1] < ind_ValD.value [2]) and
         (ind_swingslowk.value [1] <> -1) then
         trade.shortatmarket (mysize, 'sell swing high');
   end
   else
   begin
      if (data1.barsnum [0] - trade.openpositionentrybar) >
param4.int then
         trade.shortexitatmarket (mysize, 'short exit');
   end;
   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_swinghigh()
dim ind_ValD, ind_swingslowk

   mysize = param5.int
   exitbar = param4.int

   itself.makeindicator "myslowk", "slowk", Array("1"), _
                        Array(param1.str, param2.str)
   ind_ValD = itself.makeindicator ("myavgslowk", "average", _
                        Array("myslowk"), Array(param3.str))
   ind_swingslowk = itself.makeindicator ("myswing",
"swinghigh", _
                           Array("myslowk"), Array("1", "2",
"50"))

   if trade.openpositionflat then
      trade.cancelallopenorders

      if (ind_ValD.value (0) < ind_ValD.value (1)) and _
         (ind_ValD.value (1) < ind_valD.value (2)) and _
         (ind_swingslowk.value (1) <> -1) then
         trade.shortatmarket mysize, "sell short"
      end if
   else
      if (data1.barsnum (0) - trade.openpositionentrybar) >=
exitbar then
          trade.shortexitatmarket mysize, "short exit"
      end if
   end if

   ex_vbs_swinghigh = trade.currentequity
end function
```

Formula Language

```
$mypbar := if (openpositionflat = 1, 0, $mypbar+1);
$mysize := param5;
$myexitbar := param4;
ind_ValD := average(slowk(data1, param1, param2), param3);
```

```
ind_slowkswing := swinghigh(slowk(data1, param1, param2), 1, 2,
50);
shortatmarket ((Openpositionflat = 1) and
               (ind_ValD (0) < ind_ValD (1)) and
               (ind_ValD (1) < ind_ValD (2)) and
               (ind_slowkswing (1) <> -1), $mysize, "short at
market");
shortexitatmarket ($mypbar >= $myexitbar, $mysize, "short
exit");
plot1 := currentequity;
```

Formula Column

```
swinghigh (D1, 1, 2, 50)
```

Swing Index (swingindex)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'swingindex', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "swingindex", Array("N"),
Array("N"))
```

Formula Language

```
swingindex(dataN, N)
```

Formula Column

```
swingindex(Tn, N)
```

Definition

Swing Index is a special indicator developed by Welles Wilder as described in his book New Concepts in Technical Trading Systems. It is usually used through its derivative, Accumulation Swing Index. Swing Index takes one parameter Limit and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_swingindex : double;
var ind_smoothswingindex : variant;
begin
   itself.makeindicator ('myswingidx', 'swingindex', ['1'],
[param1.str]);

   ind_smoothswingindex := itself.makeindicator ('mysmooth',
'average',
                                                ['myswingidx'],
[param2.str]);

   if not ind_smoothswingindex.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := ind_smoothswingindex.Value [0];
end;
```

VBScript

```
function ex_vbs_swingindex()
dim ind_smoothswingindex

   itself.makeindicator "myswingidx", "swingindex", Array("1"),
_
                                      Array(param1.str)
   ind_smoothswingindex = itself.makeindicator ("mysmooth",
"average", _
                                 Array("myswingidx"),
Array(param2.str))

   if not ind_smoothswingindex.valid (0) then
      itself.success = false
      exit function
   end if

   ex_vbs_swingindex = ind_smoothswingindex.Value (0)
end function
```

Formula Language

```
plot1 := average(swingindex(data1, param1), param2);
```

Formula Column

```
swingindex(m10, 9999)
```

Swing Low Bar (swinglowbar)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'swinglowbar', ['N'], ['N', 'N',
'N']);
```

VBScript

```
Itself.MakeIndicator("string", "swinglowbar", Array("N"),
Array("N", "N", "N"))
```

Formula Language

```
swinglowbar(dataN, N, N, N)
```

Formula Column

```
swinglowbar(Tn, N, N, N)
```

Definition

Swing Low Bar returns the bars ago of the Nth occurence (Occur) of the swing low point of Strength bars over Length period.

Examples

Delphi Script

```
function ex_del_swinglowbar  : double;
var ind_slb : variant;
begin
   ind_slb := itself.makeindicator ('myslb', 'swinglowbar',
['1'],
                                    [param1.str, param2.str,
param3.str]);

   if not ind_slb.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   if ind_slb.value [0] = param2.int then
      result := Data1.low [0]
   else
      itself.success := false;
```

```
end;
```

VBScript

```
function ex_vbs_swinglowbar()
dim ind_slb
   ind_slb = itself.makeindicator ("myslb", "swinglowbar",
Array("1"), _
                                Array(param1.str, param2.str,
param3.str))

   if not ind_slb.valid (0) then
      itself.success = false
      exit function
   end if

   if ind_slb.value (0) = param2.int then
      ex_vbs_swinglowbar = Data1.low (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := if(swinglowbar(data1, param1, param2, param3)= param2,
low, 0);
success1 := if(swinglowbar(data1, param1, param2, param3)=
param2, 1, 0);
```

Formula Column

```
swinglowbar(m15, 1, 3, 50)
```

Swing Low (swinglow)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'swinglow', ['N'], ['N', 'N',
'N']);
```

VBScript

```
Itself.MakeIndicator("string", "swinglow", Array("N"),
Array("N", "N", "N"))
```

Formula Language

```
swinglow(dataN, N, N, N)
```

Formula Column

```
swinglow(Tn, N, N, N)
```

Definition

Swing Low returns the Nth occurrence (Occur) of the swing low point of Strength bars over Length period.

Examples

Delphi Script

```
function ex_del_swinglow : double;
var ind_ValD, ind_swingslowk : variant;
    mysize, myExitBar : integer;
begin
   mysize := param5.int;
   myExitBar := param4.int;

   itself.makeindicator ('myslowk', 'slowk', ['1'],
                                       [param1.str, param2.str]);
   ind_ValD := itself.makeindicator ('myavg', 'average',
                                       ['myslowk'],
[param3.str]);
   ind_swingslowk := itself.makeindicator ('myswing',
'swinglow',
                                       ['myslowk'], ['1', '3',
'50']);

   if not ind_ValD.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   if trade.openpositionflat then
   begin
      trade.cancelallopenorders;
      if (ind_ValD.value [0] < ind_ValD.value [1]) and
         (ind_ValD.value [1] < ind_ValD.value [2]) and
         (ind_swingslowk.value [1] <> -1) then
         trade.longatmarket (mysize, 'long swing low');
   end
   else
   begin
      if (data1.barsnum [0] - trade.openpositionentrybar) >
myExitBar then
         trade.longexitatmarket (mysize, 'long exit');
   end;

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_swinglow()
dim ind_ValD, ind_slowkswing
   mysize = param5.int
   myexitbar = param4.int

   itself.makeindicator "myslowk", "slowk", Array("1"), _
                                   Array(param1.str,
param2.str)
   ind_ValD = itself.makeindicator ( "myavg", "average",
Array("1"), _

Array(param3.str))
   ind_slowkswing = itself.makeindicator ( "myswingk",
"swinglow", _
                           Array("myslowk"), Array("1", "3",
"50"))

   if not ind_ValD.valid (0) then
      itself.success = false
      exit function
   end if

   if trade.openpositionflat then
      trade.cancelallopenorders

      if (ind_ValD.value (0) < ind_ValD.value (1)) and _
         (ind_ValD.value (1) < ind_ValD.value (2)) and _
         (ind_slowkswing.value (1) <> -1) then
         trade.longatmarket mysize, "long swing low"
      end if
   else
```

```
      if (data1.barsnum (0) - trade.openpositionentrybar) >
myexitbar then
         trade.longexitatmarket mysize, "long exit"
      end if
   end if

   ex_vbs_swinglow = trade.currentequity
end function
```

Formula Language

```
$mypbar := if(openpositionflat = 1, 0, $mypbar+1);
$mysize := param5;
$myexitbar := param4;
ind_ValD := average(slowk(data1, param1, param2), param3);
ind_slowkswing := swinglow(slowk(data1, param1, param2), 1, 3,
50);

longatmarket ((ind_ValD (0) > ind_ValD (1)) and
              (ind_ValD (1) > ind_valD (2)) and
              (ind_slowkswing (1) <> -1), $mysize, "long
swinglow");

longexitatmarket ($mypbar >= $myexitbar, $mysize, "long exit");
plot1 := currentequity
```

Formula Column

```
swinglow(m15, 1, 3, 50)
```

Three Line Break (threelinebreak)

Three line break ex is an obsolete indicator, three line break is now a directly supported chart type.

Please refer to ***Time Chart Operation Guide, Three Line Break Chart*** (see "Three Line Break Chart" on page 1176) for more information.

Three Line Break Ex (threelinebreakex)

Three line break ex is an obsolete indicator, three line break is now a directly supported chart type.

Please refer to ***Time Chart Operation Guide, Three Line Break Chart*** (see "Three Line Break Chart" on page 1176) for more information.

Tick (tick)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'tick', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "tick", Array("N"), Array(""))
```

Formula Language

```
tick(dataN)
```

Formula Column

```
tick(Tn)
```

Definition

Returns the tick volume of Link 1. Tick volume is the number of data units that constitute to a bar. The value is undefined if Link 1 does not contain any tick volume information, e.g. Link 1 is an indicator.

This indicator is the same as TickVol.

Examples

Delphi Script

```
function ex_del_tick : double;
var ind_tick : variant;
begin
   ind_tick := itself.makeindicator ('mytick', 'tick', ['1'],
['']);

   if not ind_tick.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   if ind_tick.value [0] > param1.int then
   begin
      itself.plot [1] := Data1.high [0];
      itself.plot [2] := data1.low [0];
      itself.plot [3] := param2.color;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
   function ex_vbs_tick()
   dim ind_tick
      ind_tick = itself.makeindicator ("mytick", "tick",
   ArraY("1"), Array(""))

      if not ind_tick.valid (0) then
         itself.successall = false
         exit function
      end if

      if ind_tick.value (0) > param1.int then
         itself.plot (1) = data1.high (0)
         itself.plot (2) = data1.low (0)
         itself.plot (3) = param2.color
      else
         itself.successall = false
      end if
   end function
```

Formula Language

```
   plot1 := if (tick(data1) > param1, h, 0);
   plot2 := if (tick(data1) > param1, l, 0);
   plot3 := if (tick(data1) > param1, param2, 0);
   success1 := if (tick(data1) > param1, 1, 0);
   success2 := if (tick(data1) > param1, 1, 0);
   success3 := if (tick(data1) > param1, 1, 0);
```

Formula Column

```
   tick(m1)
```

TickVol (tickvol)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'tickvol', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "tickvol", Array("N"),
Array(""))
```

Formula Language

```
tick(dataN)
```

Formula Column

```
tick(Tn)
```

Definition

Returns the tick volume of Link 1. Tick volume is the number of data units that constitute to a bar. The value is undefined if Link 1 does not contain any tick volume information, e.g. Link 1 is an indicator.

This indicator is the same as Tick.

Examples

Delphi Script

```
function ex_del_tickvol : double;
var ind_tick : variant;
begin
   ind_tick := itself.makeindicator ('mytick', 'tickvol',
['1'], ['']);

   if not ind_tick.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   if ind_tick.value [0] > param1.int then
   begin
      itself.plot [1] := Data1.high [0];
      itself.plot [2] := data1.low [0];
      itself.plot [3] := param2.color;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function ex_vbs_tick()
dim ind_tick
   ind_tick = itself.makeindicator ("mytick", "tickvol",
ArraY("1"), Array(""))

   if not ind_tick.valid (0) then
      itself.successall = false
      exit function
   end if

   if ind_tick.value (0) > param1.int then
      itself.plot (1) = data1.high (0)
      itself.plot (2) = data1.low (0)
      itself.plot (3) = param2.color
   else
      itself.successall = false
   end if
end function
```

Formula Language

```
plot1 := if (tickvol(data1) > param1, h, 0);
plot2 := if (tickvol(data1) > param1, l, 0);
plot3 := if (tickvol(data1) > param1, param2, 0);
success1 := if (tickvol(data1) > param1, 1, 0);
success2 := if (tickvol(data1) > param1, 1, 0);
success3 := if (tickvol(data1) > param1, 1, 0);
```

Formula Column

```
tickvol(m1)
```

Time Frame Period (timeframeperiod)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'timeframeperiod', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "timeframeperiod", Array("N"),
Array(""))
```

Formula Language

```
timeframeperiod(dataN)
```

Formula Column

```
timeframeperiod(Tn)
```

Definition

Time Frame Period returns the bar size of the linked series. Time Frame Period requires using one series (Link 1).

Examples

Delphi Script

```
function ex_del_timeframeperiod : double;
var ind_tfp, ind_myavg : variant;
    myperiod : string;
begin
   ind_tfp := itself.makeindicator ('mytfp', 'timeframeperiod',
['1'], ['']);

   myperiod := ntlib.integer2str (ind_tfp.value [0]);

   ind_myavg := itself.makeindicator ('myavg', 'average',
['1'], [myperiod]);

   if not ind_myavg.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   result := ind_myavg.Value [0];
```

```
end;
```

VBScript

```
function ex_vbs_timeframeperiod()
dim ind_tfp, ind_avg
   ind_tfp = itself.makeindicator ("mytfp", "timeframeperiod",
_
                                            Array("1"),
Array(""))
   myperiod = ntlib.integer2str (ind_tfp.value (0))

   ind_avg = itself.makeindicator ("myavg", "average", _
                                  Array("1"), Array(myperiod))

   if not ind_avg.valid (0) then
      itself.success = false
      exit function
   end if

   ex_vbs_timeframeperiod = ind_avg.Value (0)
end function
```

Formula Language

```
plot1 := average(data1, timeframeperiod(data1));
```

Formula Column

```
timeframeperiod(m31)
```

Time Series Forecast (tsf)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'tsf', ['N'], ['N','N']);
```

VBScript

```
Itself.MakeIndicator("string", "tsf", Array("N"),
Array("N","N"))
```

Formula Language

```
tsf(dataN, N, N)
```

Formula Column

```
tsf(Tn, N, N)
```

Definition

It is equivalent to linear regression value. Time Series Forcast takes two parameters Period and Projection. Time Series Forecast requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_tsf : double;
var myxover, myxunder : variant;
    mysize : double;
begin
   itself.makeindicator ('mytsf', 'tsf', ['1'], [param1.str,
'0']);
   itself.makeindicator ('mymov', 'average', ['1'],
[param2.str]);

   myxover := itself.makeindicator ('xover', 'xabove',
                                    ['mytsf', 'mymov'], ['']);
   myxunder := itself.makeindicator ('xunder', 'xbelow',
                                    ['mytsf', 'mymov'], ['']);

   mysize := param3.real;

   if myxover.value [0] > 0 then
      trade.longatmarket (mysize, 'long entry');
   if myxunder.value [0] > 0 then
      trade.shortatmarket (mysize, 'short entry');

   result := trade.currentequity;
end;
```

VBScript

```
function ex_vbs_tsf()
dim myxover, myxunder
   mysize = param3.real

   itself.makeindicator "mytsf", "tsf", Array("1"),
Array(param1.str, "0")
   itself.makeindicator "mymov", "average", Array("1"),
Array(param2.str)

   myxover = itself.makeindicator ("myxo", "xabove", _
                                   Array("mytsf", "mymov"),
Array(""))
   myxunder = itself.makeindicator ("myxu", "xbelow", _
                                    Array("mytsf", "mymov"),
Array(""))

   if myxover.value (0) > 0 then
      trade.longatmarket mysize, "long entry"
   end if

   if myxunder.value (0) > 0 then
      trade.shortatmarket mysize, "short entry"
   end if

   ex_vbs_tsf = trade.currentequity
end function
```

Formula Language

```
$mysize := param3;
longatmarket (xabove (tsf (data1, param1, 0), average (data1,
param2)) >0,
              $mysize, "long entry");
shortatmarket (xbelow (tsf (data1, param1, 0), average (data1,
param2)) >0,
              $mysize, "short entry");
plot1 := currentequity
```

Formula Column

```
tsf(m1, 5, 10)
```

TP Bid Ask Trade N (tpbidasktraden)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'tpbidasktraden', ['N'],
['N','string','string']);
```

VBScript

```
Itself.MakeIndicator("string", "tpbidasktraden", Array("N"),
Array("N","string","string"))
```

Formula Language

```
tpbidasktraden(dataN, N, "string", "string")
```

Formula Column

```
tpbidasktraden(Tn, N, "string", "string")
```

Definition

TP Bid Ask Trade N returns the sum of Bid Ask Trade value over the last N ticks. TP Bid Ask Trade N is one of the Tick Precise (TP) indicators which requires using Tick Replay to obtain correct historical values. TP Bid Ask Trade N requires 1 series (Link 1).

Note: When calling TP indicators within scripts, these scripts must be update by tick scripts in order for TP indicators to produce value.

See also ***Tick Precise Indicators*** (on page 996).

Examples

Delphi Script

```
function ex_del_tpbidasktraden : double;
var mytpbidasktraden : variant;
    alertlevel : integer;
begin
   alertlevel := param1.int;

   mytpbidasktraden := itself.makeindicator ('tpbat',
'tpbidasktraden', ['1'],
                                                    ['16', 'N Tick',
'No Break']);
```

```
   if not mytpbidasktraden.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   if (alertlevel > 0) and
      (mytpbidasktraden.value [0] > alertlevel) then
   begin
      itself.alert ('high', 'break upper level', clgreen,
clblack, false);
      result := Data1.high [0];
   end
   else if (alertlevel < 0) and
           (mytpbidasktraden.value [0] < alertlevel) then
   begin
      itself.alert ('high', 'break lower level', clred,
clblack, false);
      result := data1.low [0];
   end
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_tpbidasktraden()
dim mytpbidasktraden

   alertlevel = param1.int

   mytpbidasktraden = itself.makeindicator ("tpbatn",
"tpbidasktraden", _
                              Array("1"), Array("16", "Tick N",
"No Break"))

   if not mytpbidasktraden.valid (0) then
      itself.success = false
      exit function
   end if

   if (alertlevel > 0) and _
      (mytpbidasktraden.value (0) > alertlevel) then
      itself.alert "high", "break upper level", clgreen,
clblack, false
      ex_vbs_tpbidasktraden = Data1.high (0)
   elseif (alertlevel < 0) and _
           (mytpbidasktraden.value (0) < alertlevel) then
      itself.alert "high", "break low level", clred, clblack,
false
      ex_vbs_tpbidasktraden = Data1.high (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
$mytpbat := tpbidasktraden( data1, 16, "Tick N", "No Break");
$alertlevel := param1;
alert (islastbar > 0 and
       $alertlevel > 0 and $mytpbat > $alertlevel,
       "High", "break upper level", cllime, clblack, false);

alert (islastbar > 0 and
       $alertlevel < 0 and $mytpbat < $alertlevel,
       "high", "break lower level", clred, clblack, false);

plot1 := if($alertlevel > 0 and $mytpbat>$alertlevel, h,
             if($alertlevel < 0 and $mytpbat < $alertlevel, l,
0));
success1 := if(absvalue($mytpbat) > absvalue($alertlevel), 1,
0);
```

Formula Column

```
tpbidasktraden (m1, 16, "N Tick", "No Break");
```

TP Bid Ask Trade Vol N (tpbidasktradevoln)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'tpbidasktradevoln', ['N'],
['N','string','string']);
```

VBScript

```
Itself.MakeIndicator("string", "tpbidasktradevoln", Array("N"),
Array("N","string","string"))
```

Formula Language

```
tpbidasktradevoln(dataN, N, "string", "string")
```

Formula Column

```
tpbidasktradevoln(Tn, N, "string", "string")
```

Definition

TP Bid Ask Trade Vol N returns the sum of volume based on Bid Ask Trade value over the last N ticks. TP Bid Ask Trade Vol N is one of the Tick Precise (TP) indicators which requires using Tick Replay to obtain correct historical values. TP Bid Ask Trade Vol N requires 1 series (Link 1).

Note: When calling TP indicators within scripts, these scripts must be update by tick scripts in order for TP indicators to produce value.

See also ***Tick Precise Indicators*** (on page 996).

Examples

Delphi Script

```
function ex_del_tpbidasktradevoln : double;
var myxo : variant;
begin
   itself.makeindicator ('tpbatv', 'tpbidasktradevoln',
                                   ['1'], ['16', 'N Tick', 'No
Break']);

   itself.makeindicator ('avgvol', 'average', ['tpbatv'],
['10']);
```

```
   myxo := itself.makeindicator ('xo', 'xabove', ['tpbatv',
'avgvol'], ['']);

   if myxo.value [0] > 0 then
   begin
      itself.alert ('high', 'ma crossover', clgreen, clblack,
false);
      result := Data1.Value [0];
   end
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_tpbidasktradevoln()
dim myxo
   itself.makeindicator "tpbatv", "tpbidasktradevoln",
Array("1"), _
                                  Array("16", "N Tick", "No
Break")
   itself.makeindicator "avgvol", "average", _
                        Array("tpbatv"), Array("10")
   myxo = itself.makeindicator ("myxo", "xabove", _
                                Array("tpbatv", "avgvol"),
Array(""))

   if myxo.value (0) > 0 then
      itself.alert "high", "ma crossover", clgreen, clblack,
false
      ex_vbs_tpbidasktradevoln = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
tpbatv := tpbidasktradevoln (data1, 16, "N Tick", "No Break");
avgvol := average(tpbatv, 10);
$myxo  := xabove(tpbatv, avgvol);
plot1 := if($myxo > 0, c, 0);
success1 := if($myxo >0, 1, 0);
```

Formula Column

```
tpbidasktradevoln (m1, 16, "N Tick", "No Break");
```

TP Tick N (tptickn)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'tptickn', ['N'],
['N','string','string','string']);
```

VBScript

```
Itself.MakeIndicator("string", "tptickn", Array("N"),
Array("N","string","string","string"))
```

Formula Language

```
tptickn(dataN, N, "string", "string", "string")
```

Formula Column

```
tptickn(Tn, N, "string", "string", "string")
```

Definition

TP Tick N returns the sum of Tick value over the last N ticks. TP Tick N is one of the Tick Precise (TP) indicators which requires using Tick Replay to obtain correct historical values. TP Tick N requires 1 series (Link 1).

Note: When calling TP indicators within scripts, these scripts must be update by tick scripts in order for TP indicators to produce value.

See also ***Tick Precise Indicators*** (on page 996).

Examples

Delphi Script

```
function ex_del_tptickn : double;
var mytptickn : variant;
    alertvalue : integer;
begin
   alertvalue := param1.int;

   if pheap.size = 0 then
   begin
      pheap.allocate (1);
      pheap.value [0] := 0;
   end;
   mytptickn := itself.makeindicator ('mytn', 'tptickn', ['1'],
                                      ['16', 'UD', 'N Tick',
'No Break']);

   if not mytptickn.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   if mytptickn.value [0] > alertvalue then
   begin
      result := data1.value [0];

      if data1.islastbar and (pheap.value [0] <> data1.barsnum
[0]) then
         ntlib.playsound ('boing.wav');
   end
   else
     itself.success := false;
   pheap.value [0] := data1.barsnum [0];
end;
```

VBScript

```
function ex_vbs_tptickn()
dim mytptickn

   alertvalue = param1.int

   if pheap.size = 0 then
      pheap.allocate (1)
      pheap.value (0) = 0
   end if

   mytptickn = itself.makeindicator ("mytn", "tptickn",
Array("1"), _
                                    Array("16", "UD", "N
Tick", "No Break"))

   if not mytptickn.valid (0) then
      itself.success = false
      exit function
   end if

   if mytptickn.value (0) > alertvalue then
      ex_vbs_tptickn = Data1.Value (0)

      if data1.islastbar and _
         (pheap.value (0) <> data1.barsnum (0)) then
         ntlib.playsound ("boing.wav")
      end if
   else
      itself.success = false
   end if
   pheap.value (0) = data1.barsnum (0)
end function
```

Formula Language

```
$mytptickn := tptickn (data1, 16, "UD", "Tick N", "No Break");
$alertvalue := param1;

playsound (islastbar > 0 and samebarupdate(data1) <= 0 and
           $mytptickn > $alertvalue, "boing.wav");

plot1 := if($mytptickn > $alervalue, data1, 0);
success1 := if($mytptickn > $alervalue, 1, 0);
```

Formula Column

```
tptickn (m1, 16, "UD", "N Tick", "No Break");
```

TP Tick Vol N (tptickvoln)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'tptickvoln', ['N'],
['N','string','string','string']);
```

VBScript

```
Itself.MakeIndicator("string", "tptickvoln", Array("N"),
Array("N","string","string","string"))
```

Formula Language

```
tptickvoln(dataN, N, "string", "string", "string")
```

Formula Column

```
tptickvoln(Tn, N, "string", "string", "string")
```

Definition

TP Tick Vol N returns the sum of volume based on Tick value over the last N ticks. TP Tick Vol N is one of the Tick Precise (TP) indicators which requires using Tick Replay to obtain correct historical values. TP Tick Vol N requires 1 series (Link 1).

Note: When calling TP indicators within scripts, these scripts must be update by tick scripts in order for TP indicators to produce value.

See also ***Tick Precise Indicators*** (on page 996).

Examples

Delphi Script

```
function ex_del_tptickvoln : double;
var mytptickvoln, myavg : variant;
begin
   mytptickvoln := itself.makeindicator ('mytpvoln',
'tptickvoln', ['1'],
                          ['16', 'UD', 'N Tick', 'No Break']);

   myavg := itself.makeindicator ('myavg', 'average',
['mytpvoln'], ['20']);

   if not (mytptickvoln.valid [0] and myavg.valid [0]) then
```

```
      begin
         itself.successall := false;
         exit;
      end;

      itself.plot [1] := mytptickvoln.value [0];
      itself.plot [2] := 0;

      if mytptickvoln.value [0] > myavg.value [0] then
         itself.plot [3] := clgreen
      else if mytptickvoln.value [0] < myavg.value [0] then
         itself.plot [3] := clred
      else
         itself.plot [3] := clyellow;
   end;
```

VBScript

```
function ex_vbs_tptickvoln()
dim mytptickvoln, myavg
   mytptickvoln = itself.makeindicator ("mytpvol",
"tptickvoln", Array("1"), _
                         Array("16", "UD", "N Tick", "No
Break"))

   myavg = itself.makeindicator ("myavg", "average",
Array("mytpvol"), _
                                          Array("20"))

   if not (mytptickvoln.valid (0) and myavg.valid (0)) then
      itself.successall = false
      exit function
   end if

   itself.plot (1) = mytptickvoln.value (0)
   itself.plot (2) = 0

   if mytptickvoln.value (0) > myavg.value (0) then
      itself.plot (3) = clgreen
   elseif mytptickvoln.value (0) < myavg.value (0) then
      itself.plot (3) = clred
   else
      itself.plot (3) = clyellow
   end if
end function
```

Formula Language

```
plot1 := tptickvoln(data1, 16, "UD", "N Tick", "No Break");
plot2 := 0;
$myavg := average (plot1, 20);
plot3 := if(plot1 > $myavg, clgreen, if(plot1 < $myavg, clred,
clyellow));
```

Formula Column

```
tptickvoln (m1, 16, "UD", "N Tick", "No Break");
```

Trade Simulator (tradesim)

This indicator generates equity curve from a single instrument trade simulator record.

For usage details, please refer to ***Trade Simulator Indicator*** (on page 1341).

Trade Simulator Portfolio (tradesim_port)

This indicator generates equity curve from multiple instruments trade simulator record.

For usage details, please refer to ***Trade Simulator Portfolio Indicator*** (on page 1343).

Triangle Moving Average (taverage)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'taverage', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "taverage", Array("N"),
Array("N"))
```

Formula Language

```
taverage(dataN, N)
```

Formula Column

```
taverage(Tn, N)
```

Definition

Returns the triangle moving average over Period. Triangle Moving Average requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_taverage : double;
var mydiff, mysignal : variant;
begin
   itself.makeindicator ('sma', 'taverage', ['1'],
[param1.str]);
   itself.makeindicator ('lma', 'taverage', ['1'],
[param2.str]);

   mydiff := itself.makeindicator ('mydiff', 'diff', ['sma',
'lma'], ['']);
   mysignal := itself.makeindicator ('mysignal', 'average',
                                                  ['mydiff'],
[param3.str]);

   if not (mydiff.valid [0] and mysignal.valid [0]) then
   begin
      itself.successall := false;
      exit;
   end;
   itself.plot [1] := mydiff.value [0];
   itself.plot [2] := mysignal.value [0];
   itself.plot [3] := mydiff.value [0] - mysignal.value [0];
end;
```

VBScript

```
function ex_vbs_taverage ()
dim mydiff, mysignal
   itself.makeindicator "sma", "taverage", Array("1"),
Array(param1.str)
   itself.makeindicator "lma", "taverage", Array("1"),
Array(param2.str)

   mydiff = itself.makeindicator ("myd", "diff", _
                                  Array("sma", "lma"),
Array(""))
   mysignal = itself.makeindicator ("mys", "average", _
                                    Array("myd"),
Array(param3.str))

   if not (mydiff.valid (0) and mysignal.valid (0)) then
      itself.successall = false
      exit function
   end if

   itself.plot (1) = mydiff.value (0)
   itself.plot (2) = mysignal.value (0)
   itself.plot (3) = mydiff.value (0) - mysignal.value (0)
end function
```

Formula Language

```
sma := taverage (data1, param1);
lma := taverage (data1, param2);
plot1 := sma-lma;
plot2 := average(plot1, param3);
plot3 := plot1-plot2;
```

Formula Column

```
taverage(m1, 5)
```

Triple Exponential Moving Average (tema)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'tema', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "tema", Array("N"), Array("N"))
```

Formula Language

```
tema(dataN, N)
```

Formula Column

```
tema(Tn, N)
```

Definition

Returns TEMA as described in Technical Analysis Stocks and Commodities magazine.

Examples

Delphi Script

```
function ex_del_tema : double;
var sma, lma : variant;
begin
   sma := itself.makeindicator ('shortma', 'tema', ['1'],
[param1.str]);
   lma := itself.makeindicator ('longma', 'tema', ['1'],
[param2.str]);

   if not (sma.valid [0] and lma.valid [0]) then
   begin
      itself.success := false;
      exit;
   end;
   result := sma.Value [0]-lma.value [0];
end;
```

VBScript

```
function ex_vbs_tema()
dim sma, lma
```

```
   sma = itself.makeindicator ("sma", "tema", Array("1"),
Array(param1.str))
   lma = itself.makeindicator ("lma", "tema", Array("1"),
Array(param2.str))

   if not (sma.valid (0) and lma.valid (0)) then
      itself.success = false
      exit function
   end if

   ex_vbs_tema = sma.Value (0) - lma.value (0)
end function
```

Formula Language

```
plot1 := tema(data1, param1) - tema(data1, param2);
```

Formula Column

```
tema(m1, 5)
```

TRIX (trix)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'trix', ['N'], ['N', 'N', 'N',
'N']);
```

VBScript

```
Itself.MakeIndicator("string", "trix", Array("N"), Array("N",
"N", "N","N"))
```

Formula Language

```
trix(dataN, N, N, N, N)
```

Formula Column

```
trix(Tn, N, N, N, N)
```

Definition

TRIX, the percent rate-of-change of a triple smoothed moving average. TRIX requires using one series (Link 1).

Examples

Delphi Script

```
function ex_del_trix : double;
var myxo : variant;
begin
   itself.makeindicator ('mytrix', 'trix', ['1'],
                     [param1.str, param1.str, param1.str, '1']);
   itself.makeindicator ('mytsignal', 'average', ['mytrix'],
[param2.str]);
   myxo := itself.makeindicator ('myxo', 'xabove',
                                          ['mytrix',
'mytsignal'], ['']);

   if myxo.value [0] > 0 then
      result := Data1.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_trix()
dim myxo
   itself.makeindicator "mytrix", "trix", Array("1"), _
                     Array(param1.str, param1.str,
param1.str, "1")
   itself.makeindicator "mytsignal", "average",
Array("mytrix"), Array("10")
   myxo = itself.makeindicator ("xo", "xabove", _
                             Array("mytrix", "mytsignal"),
Array(""))

   if myxo.value (0) > 0 then
      ex_vbs_trix = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
mytrix := trix(data1, param1, param1, param1, 1);
$myxo := xabove(mytrix, average(mytrix, param2));
plot1 := if($myxo >0, c, 0);
success1 := if($myxo >0, 1, 0);
```

Formula Column

```
trix(m1, 10, 10, 10, 1)
```

True High (truehigh)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'truehigh', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "truehigh", Array("N"),
Array(""))
```

Formula Language

```
truehigh(dataN)
```

Formula Column

```
truehigh(Tn)
```

Definition

Returns True High of the current bar. True High requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_truehigh : double;
var hth : variant;
begin
   itself.makeindicator ('myth', 'truehigh', ['1'], ['']);
   hth := itself.makeindicator ('hth', 'hhv', ['myth'],
[param1.str]);

   result := hth.Value [0];
end;
```

VBScript

```
function ex_vbs_truehigh()
dim hth
   itself.makeindicator "myth", "truehigh", Array("1"),
Array("")
   hth = itself.makeindicator ("hth", "hhv", Array("myth"),
Array(param1.str))
   ex_vbs_truehigh = hth.Value (0)
end function
```

Formula Language

```
plot1 := hhv(truehigh(data1), 10);
```

Formula Column

```
truehigh(m1)
```

True Low (truelow)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'truelow', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "truelow", Array("N"),
Array(""))
```

Formula Language

```
truelow(dataN)
```

Formula Column

```
truelow(Tn)
```

Definition

Returns True Low of the current bar. True Low requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_truelow : double;
var ltl : variant;
begin
   itself.makeindicator ('mytl', 'truelow', ['1'], ['']);
   ltl := itself.makeindicator ('ltl', 'llv', ['mytl'],
[param1.str]);
   result := ltl.Value [0];
end;
```

VBScripe

```
function ex_vbs_truelow()
dim ltl
   itself.makeindicator "mytl", "truelow", Array("1"),
Array("")
   ltl = itself.makeindicator ("ltl", "llv", Array("mytl"),
Array(param1.str))
   ex_vbs_truelow = ltl.Value (0)
end function
```

Formula Language

```
plot1 := llv(truelow(data1), param1);
```

Formula Column

```
truelow(m1)
```

True Range (truerange)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'truerange', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "truerange", Array("N"),
Array(""))
```

Formula Language

```
truerange(dataN)
```

Formula Column

```
truerange(Tn)
```

Definition

Returns the True Range of a price bar. True Range requires one data series (Link 1).

Examples

Delphi Script

```
function ex_del_truerange : double;
var myavg : variant;
begin
   itself.makeindicator ('mytr', 'truerange', ['1'], ['']);
   myavg := itself.makeindicator ('myavg', 'qc_xaverage',
                                              ['mytr'],
[param1.str]);

   if myavg.valid [0] then
      result := myavg.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_truerange()
dim myavg
   itself.makeindicator "mytr", "truerange", Array("1"),
Array("")
```

```
   myavg = itself.makeindicator ("myavg", "qc_xaverage", _
                                           Array("mytr"),
Array(param1.str))

   if myavg.valid (0) then
      ex_vbs_truerange = myavg.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := qc_xaverage(truerange(data1), param1);
```

Formula Column

```
truerange(m1)
```

Ultimate Oscillator (ultimate)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'ultimate', ['N'],
['N','N','N']);
```

VBScript

```
Itself.MakeIndicator("string", "ultimate", Array("N"),
Array("N","N","N"))
```

Formula Language

```
ultimate(dataN, N, N, N)
```

Formula Column

```
ultimate(Tn, N, N, N)
```

Definition

An oscillator based on the weighted combination of oscillator values from three different time periods. Ultimate Oscillator takes three parameters Period1, Period2 and Period3; Ultimate Oscillator requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_ultimate : double;
var xo, xu as variant;
begin
   itself.makeindicator ('myu', 'ultimate', ['1'], ['4', '14',
'28']);

   xu := itself.makeindicator ('myxu', 'xbelowconst', ['myu'],
[param1.str]);
   xo := itself.makeindicator ('myxo', 'xaboveconst', ['myu'],
[param2.str]);

   if xo.value [0] > 0 then
   begin
      itself.plot [1] := Data1.high [0];
      itself.successex [2] := false;
   end
   else if xu.value [0] > 0 then
   begin
      itself.successex [1] := false;
      itself.plot [2] := data1.low [0];
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function ex_vbs_ultimate()
dim xu, xo

   itself.makeindicator "myu", "ultimate", _
                        Array("1"), Array("4", "12", "28")
   xo = itself.makeindicator ("myxo", "xaboveconst",
Array("myu"), _

Array(param1.str))
   xu = itself.makeindicator ("myxu", "xbelowconst",
Array("myu"), _

Array(param2.str))

   if xo.value (0) > 0 then
      itself.plot (1) = data1.high (0)
      itself.successex (2) = false
   elseif xu.value (0) > 0 then
      itself.successex (1) = false
      itself.plot (2) = data1.low (0)
   else
      itself.successall = false
   end if
end function
```

Formula Language

```
$xo := xaboveconst(ultimate(data1, 4, 12, 28), param1);
$xu := xbelowconst(ultimate(data1, 4, 12, 28), param2);
plot1 := if ($xo > 0, h, 0);
plot2 := if ($xu > 0, l, 0);
success1 := if ($xo > 0, 1, 0);
success2 := if ($xu > 0, 1, 0);
```

Formula Column

```
ultimate(m1, 4, 12, 28)
```

Up Down Tick (updowntick)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'updowntick', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "updowntick", Array("N"),
Array(""))
```

Formula Language

```
updowntick(dataN)
```

Formula Column

```
updowntick(Tn)
```

Definition

Up Down Tick returns 4 plots for Up Tick, Down Tick, Up Volume, Down Volume. As most data vendors do not send these information with their historical data, the information is generated from tick data or real-time tick-by-tick update only. To get correct historical calculation, use tick replay in Chart Manager. Up Down Tick requires one series (Link 1)

Examples

Delphi Script

```
function ex_del_updowntick : double;
var udvol, avol : variant;
begin
   udvol := itself.makeindicator ('udtv', 'updowntick', ['1'],
['']);
   avol := itself.makeindicator ('avol', 'average', ['1.v'],
[param1.str]);

   if not (udvol.validex [3, 0] and avol.valid [0]) then
   begin
      itself.success := false;
      exit;
   end;

   if udvol.valueex [3, 0] > avol.value [0] then
      result := Data1.Value [0]
   else
```

```
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_updowntick()
dim udvol, avol
   udvol = itself.makeindicator ("udv", "updowntick",
Array("1"), Array(""))
   avol = itself.makeindicator ("avol", "average", _
                                Array("1.v"),
Array(param1.str))

   if not (udvol.validex (3, 0) and avol.valid (0)) then
      itself.success = false
      exit function
   end if

   if udvol.valueex (3, 0) > avol.value (0) then
      ex_vbs_updowntick = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
$udvol := updowntick.plot3 (data1);
$avol  := average (v, param1);
plot1 := if ( $udvol> $avol, c, 0);
success1 := if ( $udvol> $avol, 1, 0);
```

Formula Column

```
updowntick.polt3 (m1)
```

Valid (valid)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'valid', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "valid", Array("N"), Array("N"))
```

Formula Language

```
valid(dataN, N)
```

Formula Column

```
valid(Tn, N)
```

Definition

Returns the valid state of Link 1 of Period ago. The value is either 1 (for valid) or 0 (for invalid). Valid requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_valid : double;
var myvalid : variant;
begin
   myvalid := itself.makeindicator ('myv', 'valid', ['1'],
[param1.str]);

   if myvalid.value [0] < 1 then
      result := Data1.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_valid ()
dim myvalid
   myvalid = itself.makeindicator ("myv", "valid", _
                                   Array("1"),
Array(param1.str))
   if myvalid.value (0) < 1 then
```

```
      ex_vbs_valid  = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
plot1 := if (valid(data1, param1) < 1, c, 0);
success1 := if (valid(data1, param1) < 1, 1, 0)
```

Formula Column

```
valid(m1, 10)
```

Valid Bar (validb)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'validb', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "validb", Array("N"), Array(""))
```

Formula Language

```
validb(dataN)
```

Formula Column

```
validb(Tn)
```

Definition

Valid Bar returns the number of bars ago that there is a valid bar. Valid Bar requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_validb : double;
var myvalidb, myxo : variant;
begin
   myvalidb := itself.makeindicator ('myvb', 'validb', ['1'],
['']);

   itself.makeindicator ('mysma', 'average', ['1'], ['10']);
   itself.makeindicator ('mylma', 'average', ['1'], ['20']);

   myxo := itself.makeindicator ('myxo', 'xabove', ['mysma',
'mylma'], ['']);

   if (myvalidb.value [0] = 0) and (myxo.value (0) > 0) then
      result := Data1.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_validb()
dim myvalidb, myxo
   myvalidb = itself.makeindicator ("myvb", "validb",
Array("1"), Array(""))

   itself.makeindicator "mysma", "average", Array("1"),
Array("10")
   itself.makeindicator "mylma", "average", Array("1"),
Array("20")

   myxo = itself.makeindicator ("myxo", "xabove", _
                                Array("mysma", "mylma"),
Array(""))

   if myvalidb.value (0) < 1 and myxo.value (0) > 0 then
      ex_vbs_validb = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
$myxo := xabove(average(data1, 10), average(data1, 20));
plot1 := if($myxo > 0 and validb(data1) = 0, c, 0);
success1 := if($myxo > 0 and validb(data1) = 0, c, 0);
```

Formula Column

```
validb(m1)
```

Valid Count (validc)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'validc', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "validc", Array("N"),
Array("N"))
```

Formula Language

```
validc(dataN, N)
```

Formula Column

```
validc(Tn, N)
```

Definition

Valid Count returns the number of valid bars over the Period specified by the parameter. Valid Count requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_validc : double;
var myvalidc, myxo : variant;
begin
   myvalidc := itself.makeindicator ('myvc', 'validc', ['1'],
[param1.str]);

   itself.makeindicator ('myqcrsi', 'qc_rsi',
                                     ['1'], [param1.str, '70',
'30']);
   myxo := itself.makeindicator ('myxo', 'xaboveconst',
['myqcrsi'], ['70']);

   if (myxo.value [0] > 0) and
      (myvalidc.value [0] = param1.int)  then
      result := Data1.Value [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_validc()
dim myvalidc, myxo
   myvalidc = itself.makeindicator ("myvc", "validc", _
                                    Array("1"),
Array(param1.str))

   itself.makeindicator "myqcrsi", "qc_rsi", Array("1"), _
                                              Array(param1.str,
"70", "30")
   myxo = itself.makeindicator ("myxo", "xaboveconst", _
                                        Array("myqcrsi"),
Array("70"))

   if myxo.value (0) > 0 and myvalidc.value (0) = param1.int
then
      ex_vbs_validc = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
$myxo := xaboveconst(qc_rsi(data1, param1, 70, 30), 70);
plot1 := if($myxo > 0, c, 0);
success1 := if($myxo > 0 and validc(data1, param1) = param1, 1,
0);
```

Formula Column

```
validc(m1, 10)
```

Valid Percent (validp)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'validp', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "validp", Array("N"),
Array("N"))
```

Formula Language

```
validp(dataN, N)
```

Formula Column

```
validp(Tn, N)
```

Definition

Valid Percent returns the percentage of valid bars over the Period specified by the parameter. Valid Percent requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_validp : double;
var myvpl, myvps, myxo : variant;
begin
   itself.makeindicator ('sma', 'average', ['1'], ['10']);
   itself.makeindicator ('lma', 'average', ['1'], ['20']);

   myvps := itself.makeindicator ('vps', 'validp', ['sma'],
['10']);
   myvpl := itself.makeindicator ('vpl', 'validp', ['lma'],
['20']);

   myxo := itself.makeindicator ('xo', 'xabove', ['sma',
'lma'], ['']);

   if (myvps.value [0] > 90) and (myvpl.value [0] > 90) and
      (myxo.value [0] > 0) then
      result := Data1.Value [0]
   else
      itself.success := false;
```

```
end;
```

VBScript

```
function ex_vbs_validp()
dim myvpl, myvps, myxo
   itself.makeindicator "sma", "average", Array("1"),
Array("10")
   itself.makeindicator "lma", "average", Array("1"),
Array("20")

   myvps = itself.makeindicator ("vps", "validp", Array("sma"),
Array("10"))
   mylps = itself.makeindicator ("vpl", "validp", Array("lma"),
Array("20"))

   myxo = itself.makeindicator ("xo", "xabove", Array("sma",
"lma"), Array(""))

   if myvps.value (0) > 90 and mylps.value (0) > 90 and _
      myxo.value (0) > 0 then
      ex_vbs_validp = Data1.Value (0)
   else
      itself.success = false
   end if
end function
```

Formula Language

```
sma := average(data1, 10);
lma := average(data1, 20);
$vps := validp(sma, 10);
$vpl := validp(lma, 20);
$myxo := xabove(sma, lma);
plot1 := if($vps>90 and $vpl>90 and $myxo>0, c, 0);
success1 := if($vps>90 and $vpl>90 and $myxo>0, 1, 0);
```

Formula Column

```
validp(m1, 10)
```

Value When (valuewhen)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'valuewhen', ['N'],
['N','formula','formula']);
```

VBScript

```
Itself.MakeIndicator("string", "valuewhen", Array("N"),
Array("N","formula", "formula"))
```

Formula Language

```
valuewhen(dataN, N, "formula", "formula")
```

Formula Column

```
valuewhen(Tn, N, "formula", "formula")
```

Definition

Returns the value of data series when the n-th most recent occurrence of the formula condition is true. ValueWhen requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_valuewhen : double;
var myvalwhen, myxo : variant;
begin
   myvalwhen := itself.makeindicator('myvalwhen', 'valuewhen',
['1'],
                     ['2', 'xabove(average(data1,10),
average(data1,20))', 'h']);

   itself.makeindicator('sma', 'average', ['1'], ['10']);
   itself.makeindicator('lma', 'average', ['1'], ['20']);

   myxo := itself.makeindicator ('myxo', 'xabove', ['sma',
'lma'], ['']);

   if not (myvalwhen.valid [0] and myxo.valid [0]) then
   begin
      itself.success := false;
      exit;
   end;

   if myxo.value [0] > 0 then
      result := data1.high [0] - myvalwhen.value [0]
   else
      itself.success := false;
end;
```

VBScript

```
   function ex_vbs_valuewhen ()
   dim myvalwhen, myxo
      myvalwhen = itself.makeindicator ("myvw", "valuewhen",
   Array("1"), _
                  Array("2",
   "xabove(average(data1,10),average(data1,20))", "h"))

      itself.makeindicator "sma", "average", Array("1"),
   Array("10")
      itself.makeindicator "lam", "average", Array("1"),
   Array("20")

      myxo = itself.makeindicator ("myxo", "xabove", _
                                   Array("sma", "lma"), Array(""))

      if not (myvalwhen.valid (0) and myxo.valid (0)) then
         itself.success = false
         exit function
      end if

      if myxo.value (0) > 0 then
         ex_vbs_valuewhen  = Data1.high (0) - myvalwhen.value (0)
      else
         itself.success = false
      end if
   end function
```

Formula Language

```
   $myxo := xabove(average(data1, 10), average(data1, 20));
   $myvalwhen := valuewhen(data1, 2,
                 "xabove(average(data1,10),average(data1,20))",
   "h");
   plot1 := if($myxo > 0, h-$myvalwhen, 0);
   success1 := if($myxo >0, 1, 0);
```

Formula Column

```
   valuewhen(m1, 2, "xabove(average(data1,10), average(data1,
   20)", "h")
```

Variable Highest High Bar (varhhb)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'varhhb', ['N','N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "varhhb", Array("N","N"),
Array(""))
```

Formula Language

```
varhhb(dataN, dataN)
```

Formula Column

```
N/A
```

Definition

Variable Highest High Bar returns the bars ago of the highest high value in Link 1 based on the number of bars (Link 2) to include in the calculation. Variable Highest High Bar requires two series.

See also ***Tutorial: Variable Period Indicators*** (on page 239).

Examples

Delphi Script

```
function ex_del_varhhb : double;
var myvarhhb : variant;
begin
   itself.makeindicator ('mybs', 'barssince', ['1'],
                          ['xaboveconst(rsindex(data1, 10),70)
> 0']);

   myvarhhb := itself.makeindicator ('myvhhb', 'varhhb', ['1',
'mybs'], ['']);

   if not myvarhhb.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := myvarhhb.value [0];
```

```
end;
```

VBScript

```
function ex_vbs_varhhb()
dim myvarhhb
   itself.makeindicator "mybs", "barssince", Array("1"), _

Array("xaboveconst(reindex(data1,10), 70)>0")

   myvarhhb = itself.makeindicator ("myvarhhb", "varhhb", _
                                    Array("1", "mybs"),
Array(""))

   if not myvarhhb.valid (0) then
      itself.success = false
      exit function
   end if
   ex_vbs_varhhb = myvarhhb.Value (0)
end function
```

Formula Language

```
plot1 :=
varhhb(data1,barssince(data1,"xaboveconst(rsindex(data1,10),70)
>0"));
```

Formula Column

```
N/A
```

Variable Linear Regression Slope (varlinregslope)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'varlinregslope', ['N','N'],
['']);
```

VBScript

```
Itself.MakeIndicator("string", "varlinregslope",
Array("N","N"), Array(""))
```

Formula Language

```
varlinregslope(dataN, dataN)
```

Formula Column

```
N/A
```

Definition

Variable Linear Regression Slope returns the slope value based on the number of bars (Link 2) to include in the linear regression calculation using least-square approximation method. Variable Linear Regression Slope requires two series.

See also ***Tutorial: Variable Period Indicators*** (on page 239).

Examples

Delphi Script

```
function ex_del_verlinregslope : double;
var myvlrslope : variant;
begin
   itself.makeindicator ('mylb', 'lowestb', ['1.l'], ['']);

   myvlrslope := itself.makeindicator ('myvlrs',
'varlinregslope',
                                        ['1', 'mylb'], ['']);

   if not myvlrslope.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := myvlrslope.Value [0];
```

```
end;
```

VBScript

```
function ex_vbs_varlinregslope()
dim myvlrslope
   itself.makeindicator "mylb", "lowestb", Array("1"),
Array("")

   myvlrslope = itself.makeindicator ("myvlrs",
"varlinregslope", _
                       Array("1", "mylb"), Array(""))

   if not myvlrslope.valid (0) then
      itself.success = false
      exit function
   end if
   ex_vbs_varlinregslope = myvlrslope.Value (0)
end function
```

Formula Language

```
plot1 := varlinregslope(data1, lowestb(data1));
```

Formula Column

N/A

Variable Linear Regression Value (varlinreg)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'varlinreg', ['N','N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "varlinreg", Array("N","N"),
Array(""))
```

Formula Language

```
varlinreg(dataN, dataN)
```

Formula Column

```
N/A
```

Definition

Variable Linear Regression returns the projected value based on the number of bars (Link 2) to include in the linear regression calculation using least-square approximation method. Variable Linear Regression Value requires two series.

See also ***Tutorial: Variable Period Indicators*** (on page 239).

Examples

Delphi Script

```
function ex_del_varlinreg : double;
var myvlr : variant;
begin
   itself.makeindicator ('myhb', 'highestb', ['1'], ['']);

   myvlr := itself.makeindicator ('myvlr', 'varlinreg',
['1','myhb'], ['0']);
   if not myvlr.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := myvlr.Value [0];
end;
```

VBScript

```
function ex_vbs_varlinreg()
dim myvlr
   itself.makeindicator "myhb", "highestb", Array("1"),
Array("")
   myvlr = itself.makeindicator ("myvlr", "varlinreg", _
                                 Array("1", "myhb"),
Array("0"))
   if not myvlr.valid (0) then
      itself.success = false
      exit function
   end if
   ex_vbs_varlinreg = myvlr.Value (0)
end function
```

Formula Language

```
plot1 := varlinreg(data1, highestb(data1), 0);
```

Formula Column

```
N/A
```

Variable Lowest Low Bar (varllb)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'varllb', ['N','N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "varllb", Array("N","N"),
Array(""))
```

Formula Language

```
varllb(dataN, dataN)
```

Formula Column

```
N/A
```

Definition

Variable Lowest Low Bar returns the bars ago of the lowest low value in Link 1 based on the number of bars (Link 2) to include in the calculation. Variable Lowest Low Bar requires two series.

See also ***Tutorial: Variable Period Indicators*** (on page 239).

Examples

Delphi Script

```
function ex_del_varllb : double;
var myrsi, myvarllb : variant;
    mylb, mycurrb : integer;
begin
   itself.makeindicator ('mybs', 'barssince', ['1'],
                              ['xbelowconst(rsindex(data1, 10), 30)
> 0']);

   myvarllb := itself.makeindicator ('myvarllb', 'varllb',
                                        ['1.l', 'mybs'], ['']);
   myrsi := itself.makeindicator ('myrsi', 'rsindexmod', ['1'],
['10']);

   if heap.size = 0 then
   begin
      heap.allocate (1);
      heap.value [0] := 0;
   end;

   if not myrsi.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   mylb := data1.barsnum [0] - heap.value [0];

   if myrsi.value [0] < 30 then
   begin
      mylb := myvarllb.value [0];
      heap.value [0] := data1.barsnum [myvarllb.value [0]];
   end;
   result := data1.low [mylb];
end;
```

VBScript

```
function ex_vbs_varllb()
dim myvarllb, myrsi

   itself.makeindicator "mybs", "barssince", Array("1"), _
                        Array("xbelowconst(rsindexmod(data1,
10), 30) > 0")

   myvarllb = itself.makeindicator ("myvllb", "varllb", _
                                    Array("1.l", "mybs"),
Array(""))

   myrsi = itself.makeindicator ("myrsi", "rsindexmod", _
                                          Array("1"),
Array("10"))

   if heap.size = 0 then
      heap.allocate (1)
      heap.value (0) = 0
   end if

   if not myrsi.valid (0) then
      itself.success = false
      exit function
   end if
   mylb = data1.barsnum (0) - heap.value (0)
   if myrsi.value (0) < 30 then
      mylb = myvarllb.value (0)
      heap.value (0) = data1.barsnum (myvarllb.value (0))
   end if
   ex_vbs_varllb = data1.low (mylb)
end function
```

Formula Language

```
$myvarllb := varllb (l,
            barssince(data1, "xbelowconst(rsindexmod(data1,
10), 30)"));
$mylb := if(rsindexmod(data1, 10)<30, $myvarllb, $mylb+1);
plot1 := low($mylb);
```

Formula Column

N/A

Variable Persist Condition (varpersistcond)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'varpersistcond', ['N','N'],
['']);
```

VBScript

```
Itself.MakeIndicator("string", "varpersistcond",
Array("N","N"), Array(""))
```

Formula Language

```
varpersistcond(dataN, dataN)
```

Formula Column

```
N/A
```

Definition

Variable Persist Condition returns 1 when Link 1 is greater than 0 for the number of bars specified in Link 2. Variable Persist Condition requires two series.

See also ***Tutorial: Variable Period Indicators*** (on page 239).

Examples

Delphi Script

```
function ex_del_varpersistcond : double;
var myvpc : variant;
begin
   itself.makeindicator ('myhhb', 'hhb', ['1'], ['10']);

   itself.makeindicator ('mycond', 'fml', ['1'],
                         ['rsindexmod(data1, 10) > 70']);

   myvpc := itself.makeindicator ('myvpc', 'varpersistcond',
                                  ['mycond', 'myhhb'], ['']);

   if not myvpc.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := myvpc.Value [0];
end;
```

VBScript

```
function ex_vbs_varpersistcond()
dim myvpc
   itself.makeindicator "myhhb", "hhb", Array("1"), Array("10")
   itself.makeindicator "mycond", "fml", Array("1"), _
                        Array("rsindexmod(data1, 10) > 70")
   myvpc = itself.makeindicator ("myvpc", "varpersistcond", _
                                 Array("mycond", "myhhb"),
Array(""))

   if not myvpc.valid (0) then
      itself.success = false
      exit function
   end if
   ex_vbs_varpersistcond = myvpc.Value (0)
end function
```

Formula Language

```
plot1 := varpersistcond(fml(data1, "rsindexmod(data1, 10) >
70"),
                        hhb(data1, 10));
```

Formula Column

N/A

Variable RSquare (varrsquare)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'varrsquare', ['N','N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "varrsquare", Array("N","N"),
Array(""))
```

Formula Language

```
varrsquare(dataN, dataN)
```

Formula Column

```
N/A
```

Definition

Variable RSquare returns the R2 value of data in Link 1 based on the number of bars (Link 2) to include in the calculation. Variable RSquare requires two series.

See also ***Tutorial: Variable Period Indicators*** (on page 239).

Examples

Delphi Script

```
function ex_del_varrsquare : double;
var myvrs : variant;
begin
   itself.makeindicator ('myhhb', 'hhb', ['1'], ['10']);

   myvrs := itself.makeindicator ('myvrs', 'varrsquare',
                                              ['1', 'myhhb'],
['']);

   if not myvrs.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := myvrs.Value [0];
end;
```

VBScript

```
function ex_vbs_varrsquare()
dim myvrs
   itself.makeindicator "myhhb", "hhb", Array("1"), Array("10")

   myvrs = itself.makeindicator ("myvrs", "varrsquare", _
                                 Array("1", "myhhb"), 
Array(""))

   if not myvrs.valid (0) then
      itself.success = false
      exit function
   end if

   ex_vbs_varrsquare = myvrs.Value (0)
end function
```

Formula Language

```
plot1 := varrsquare(data1, hhb(data1, 10));
```

Formula Column

N/A

Variable Summation (varsum)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'varsum', ['N','N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "varsum", Array("N","N"),
Array(""))
```

Formula Language

```
varsum(dataN, dataN)
```

Formula Column

```
N/A
```

Definition

Variable Summation returns the sum of values in Link 1 based on the number of bars (Link 2) to include in the summation calculation. Variable Summation requires two series.

See also ***Tutorial: Variable Period Indicators*** (on page 239).

Examples

Delphi Script

```
function ex_del_varsum : double;
var myvs : variant;
begin
   itself.makeindicator ('myllb', 'llb', ['1.l'], ['10']);

   myvs := itself.makeindicator ('myvs', 'varsum', ['1',
'myllb'], ['']);

   if not myvs.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := myvs.Value [0];
end;
```

VBScript

```
function ex_vbs_varsum()
dim myvs
   itself.makeindicator "myllb", "llb", Array("1.1"),
Array("10")

   myvs = itself.makeindicator ("myvs", "varsum", _
                                Array("1", "myllb"), Array(""))

   if not myvs.valid (0) then
      itself.success = false
      exit function
   end if
   ex_vbs_varsum = myvs.Value (0)
end function
```

Formula Language

```
plot1 := varsum(data1, llb(data1, 10));
```

Formula Column

```
N/A
```

Volatility (vola)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'vola', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "vola", Array("N"), Array("N"))
```

Formula Language

```
vola(dataN, N)
```

Formula Column

```
vola(Tn, N)
```

Definition

Volatility returns the Welles Wilder's version of volatility measurement. Volatility takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_vola : double;
var myav : variant;
begin
   itself.makeindicator ('myvola', 'vola', ['1'],
[param1.str]);

   myav := itself.makeindicator ('myav', 'average', ['myvola'],
[param2.str]);

   if not myav.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := myav.Value [0];
end;
```

VBScript

```
function ex_vbs_vola()
```

```
dim myav
   itself.makeindicator "myvola", "vola", Array("1"),
Array(param1.str)

   myav = itself.makeindicator ("myav", "average", _
                                Array("myvola"),
Array(param2.str))

   if not myav.valid (0) then
      itself.success = false
      exit function
   end if
   ex_vbs_vola = myav.Value (0)
end function
```

Formula Language

```
plot1 := average(vola(data1, param1), param2);
```

Formula Column

```
vola(m1, 10)
```

Volume Accumulation Oscillator (vol_acc_osc)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'vol_acc_osc', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "vol_acc_osc", Array("N"),
Array("N"))
```

Formula Language

```
vol_acc_osc(dataN, N)
```

Formula Column

```
vol_acc_osc(Tn, N)
```

Definition

Volume Accumulation Oscillator is a volume momentum indicator developed by Mark Chaikin. Volume Accumulated Oscillator takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_vol_acc_osc : double;
var myvao : variant;
    dp : integer;
begin
   dp := param2.int;

   myvao := itself.makeindicator ('myvao', 'vol_acc_osc',
                                              ['1'],
[param1.str]);

   if not myvao.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   if ((data1.value [dp] > data1.value [0]) and
       (myvao.value [dp] < myvao.value [0])) or
```

```
        ((data1.value [dp] < data1.value [0]) and
         (myvao.value [dp] > myvao.value [0])) then
        result := 1
     else
        result := 0;
  end;
```

VBScript

```
function ex_vbs_vol_acc_osc()
dim myvao

   dp = param2.int

   myvao = itself.makeindicator ("myvao", "vol_acc_osc", _
                                 Array("1"), Array(param1.str))

   if not myvao.valid (0) then
      itself.success = false
      exit function
   end if

   if ((data1.value (dp) > data1.value (0)) and _
       (myvao.value (dp) < myvao.value (0))) or _
      ((data1.value (dp) < data1.value (0)) and _
       (myvao.value (dp) > myvao.value (0))) then
      ex_vbs_vol_acc_osc = 1
   else
      ex_vbs_vol_acc_osc = 0
   end if
end function
```

Formula Language

```
myvao := vol_acc_osc(data1, param1);
plot1 := if(((data1(param2)<data1) and (myvao(param2)>myvao))
or
            ((data1(param2)>data1) and (myvao(param2)<myvao)),
1, 0);
```

Formula Column

```
vol_acc_osc(m1, 10)
```

Volume Distribution (voldist)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'voldist', ['N'], ['string',
'N', 'formula', 'formula', 'string', 'Datetime', 'Datetime',
'N', 'N', 'N', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "voldist", Array("N"),
Array("string", "N", "formula", "formula", "string",
"Datetime", "Datetime", "N", "N", "N", "N"))
```

Formula Language

```
voldist(dataN, "string", N, "formula", "formula", "string",
"Datetime", "Datetime", N, N, N, N)
```

Formula Column

```
voldist(Tn, "string", N, "formula", "formula", "string",
"Datetime", "Datetime", N, N, N, N)
```

Definition

Volume Distribution returns the normal distribution region covering Mean +/- SD. You can choose from one of the caluclation model. Normal model returns the historical average as the approx. mean and historical standard deviation as the approx. SD. Gaussian model returns the predicted Normal Distribution based on the distribution cumulated so far. Volume Distribution requires one link (Link 1).

See also ***Power Indicators Guide, Volume Distribution*** (see "Volume Distribution" on page 1345).

Examples

Delphi Script

```
function ex_del_voldist : double;
var myvoldist : variant;
begin
   myvoldist := Itself.MakeIndicator('string', 'voldist',
['1'],
                ['Normal', param1.str, 'c', 'v', 'Period',
'01/01/2002',
                  '01/01/2002', '100', '100', '0.25', '0.55']);

   if not myvoldist.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   if data1.value [0] > myvoldist.valueex [1, 0] then
      result := 1
   else if data1.value [0] < myvoldist.valueex [3, 0] then
      result := -1
   else
      result := 0;
end;
```

VBScript

```
function ex_vbs_voldist()
dim myvoldist

   myvoldist = Itself.MakeIndicator("myvd", "voldist",
Array("1"), _
               Array("Normal", param1.str, "c", "v", "Period",
_
               "01/01/2002", "01/01/2001", "100", "100",
"0.25", "0.55"))

   if not myvoldist.valid (0) then
      itself.success = false
      exit function
   end if

   if data1.value (0) > myvoldist.valueex (1, 0) then
      ex_vbs_voldist = 1
   elseif data1.value (0) < myvoldist.valueex (3, 0) then
      ex_vbs_voldist = -1
   else
      ex_vbs_voldist = 0
   end if
end function
```

Formula

```
myvdupper := voldist.plot1(data1, "Normal", param1, "c", "v",
"Period",
                    "01/01/2002", "01/01/2002", 100, 100, 0.25,
0.55);
myvdlower := voldist.plot3(data1, "Normal", param1, "c", "v",
"Period",
                     "01/01/2002", "01/01/2002", 100, 100,
0.25, 0.55);
plot1 := choose(data1>myvdupper, 1, data1<myvdlower, -1, 0);
```

Formula Column

```
voldist(0,M1,"Normal",2,"c","v","Period","2002/1/1","2002/1/1",
100,100,0.1,0.55)
```

Volume Formula (volfml)

This indicator colors mark volume bar according to parameter in formula, the main application of this indicator is visual only, when call in a script return same values as volume indicator.

Volume (vol)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'vol', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "vol", Array("N"), Array(""))
```

Formula Language

```
vol(dataN)
```

Formula Column

```
vol(Tn)
```

Definition

Returns the volume of Link 1. The value is undefined if Link 1 does not contain any volume information, e.g. Link 1 is an indicator.

Examples

Delphi Script

```
function ex_del_vol : double;
var myvol, myavgvol : variant;
begin
   myvol := itself.makeindicator ('myvol', 'vol', ['1'], ['']);

   myavgvol := itself.makeindicator ('myav', 'average',
                                   ['myvol'], [param1.str]);

   if not (myvol.valid [0] and myavgvol.valid [0]) then
   begin
      itself.success := false;
      exit;
   end;
   result := myvol.value [0] -myavgvol.Value [0];
end;
```

VBScript

```
function ex_vbs_vol()
dim myvol, myavgvol
```

```
   myvol = itself.makeindicator ("myvol", "vol", Array("1"),
Array(""))
   myavgvol = itself.makeindicator ("myav", "average",
Array("myvol"), _

Array(param1.str))

   if not (myvol.valid (0) and myavgvol.valid (0)) then
      itself.success = false
      exit function
   end if

   ex_vbs_vol = myvol.Value (0) - myavgvol.value (0)
end function
```

Formula Language

```
plot1 := vol(data1)-average(vol(data1),param1);
```

Formula Column

```
vol(m1)-vol(1, m1)
```

Volume Oscillator (vosc)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'vosc', ['N'], ['N', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "vosc", Array("N"), Array("N",
"N"))
```

Formula Language

```
vosc(dataN, N, N)
```

Formula Column

```
vosc(Tn, N, N)
```

Definition

Volume Oscillator is the difference between two moving averages of the volume. Volume Oscillator takes two parameter: Period Fast and Period Slow. Volume Oscillator requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_vosc : double;
var myxo, myxu : variant;
begin
   itself.makeindicator ('myvosc', 'vosc', ['1'], [param1.str,
param2.str]);
   itself.makeindicator ('mysignal', 'average', ['myvosc'],
[param3.str]);

   myxo := itself.makeindicator ('xo', 'xabove',
                                  ['myvosc', 'mysignal'],
['']);
   myxu := itself.makeindicator ('xu', 'xbelow',
                                  ['myvosc', 'mysignal'],
['']);

  if myxo.value [0] > 0 then
  begin
     itself.plot [1] := data1.low [0]-0.01;
```

```
      itself.successex [2] := false;
   end
   else if myxu.value [0] > 0 then
   begin
      itself.successex [1] := false;
      itself.plot [2] := data1.high [0]+0.01;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function ex_vbs_vosc()
dim myxo, myxu
   itself.makeindicator "myvosc", "vosc", _
                        Array("1"), Array(param1.str,
param2.str)
   itself.makeindicator "mysignal", "average", _
                        Array("1"), Array(param3.str)

   myxo = itself.makeindicator ("xo", "xabove", _
                                Array("myvosc", "mysignal"),
Array(""))
   myxu = itself.makeindicator ("xu", "xbelow", _
                                Array("myvosc", "mysignal"),
Array(""))

   if myxo.value (0) > 0 then
      itself.plot (1) = data1.low (0)-0.01
      itself.successex (2) = false
   elseif myxu.value (0) > 0 then
      itself.successex (1) = false
      itself.plot (2) = data1.high (0)+0.01
   else
      itself.successall = false
   end if
end function
```

Formula

```
myvosc := vosc(data1, param1, param2);
mysignal := average(myvosc, param3);
myxo := xabove(myvosc, mysignal);
myxu := xbelow(myvosc, mysignal);
plot1 := if(myxo > 0, l-0.01, 0);
success1 := if(myxo > 0, 1, 0);
plot2 := if(myxu > 0, h+0.01, 0);
success2 := if(myxu > 0, 1, 0);
```

Formula Column

```
vosc(m5, 13, 30)
```

Volume Rate of Change (vroc)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'vroc', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "vroc", Array("N"), Array("N"))
```

Formula Language

```
vroc(dataN, N)
```

Formula Column

```
vroc(Tn, N)
```

Definition

Similar to the Rate of Change indicator except using volume instead of the price value. Volume Rate of Change takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_vroc : double;
var avgvroc : variant;
begin
   itself.makeindicator ('myvroc', 'vroc', ['1'],
[param1.str]);

   avgvroc := itself.makeindicator ('myavg', 'average',
['myvroc'], [param2.str]);

   if not avgvroc.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := avgvroc.Value [0];
end;
```

VBScript

```
function ex_vbs_vroc()
```

```
dim avgvroc
   itself.makeindicator "myvroc", "vroc", Array("1"),
Array(param1.str)

   avgvroc = itself.makeindicator ("mysmoothing", "average", _
                                   Array("myvroc"),
Array(param2.str))

   if not avgvroc.valid (0) then
      itself.success = false
      exit function
   end if

   ex_vbs_vroc = avgvroc.Value (0)
end function
```

Formula

```
plot1 := average(vroc(data1, param1), param2);
```

Formula column

```
vroc(m5, 10)
```

Weight Close (wclose)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'wclose', ['N'], ['']);
```

VBScript

```
Itself.MakeIndicator("string", "wclose", Array("N"), Array(""))
```

Formula Language

```
wclose(dataN)
```

Formula Column

```
wclose(Tn)
```

Definition

Returns Weighted Close. Weighted Close requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_wclose : double;
var mywmacd : variant;
begin
   itself.makeindicator ('mywc', 'wclose', ['1'], ['']);

   mywmacd := itself.makeindicator ('wmacd', 'macdpr',
['mywc'],
                           ['exponential', '12', 'exponential',
'26']);

   if not mywmacd.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   result := mywmacd.Value [0];
end;
```

VBScript

```
function ex_vbs_wclose()
```

```
dim mywmacd
   itself.makeindicator "mywc", "wclose", Array("1"), Array("")

   mywmacd = itself.makeindicator ("wmacd", "macdpr",
Array("mywc"), _
                            Array("exponential", "12",
"exponential", "26"))

   if not mywmacd.valid (0) then
      itself.success = false
      exit function
   end if
   ex_vbs_wclose = mywmacd.Value (0)
end function
```

Formula

```
plot1 := macdpr(wclose(data1), "exponential", "12",
"exponential", "26");
```

Formula Column

```
wclose(m5)
```

Weight Moving Average (waverage)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'waverage', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "waverage", Array("N"),
Array("N"))
```

Formula Language

```
waverage(dataN, N)
```

Formula Column

```
waverage(Tn, N)
```

Definition

Weighted Moving Average applied fixed ratio onto the current Period of bars with heaviest weighting on the latest bars. Weighted Moving Average takes one parameter Period and requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_waverage : double;
var myxup, myxdown : variant;
begin
   itself.makeindicator ('fwma', 'waverage', ['1'],
[param1.str]);
   itself.makeindicator ('swma', 'waverage', ['1'],
[param2.str]);

   myxup := itself.makeindicator ('myxu', 'xabove', ['fwma',
'swma'], ['']);
   myxdown := itself.makeindicator ('myxd', 'xbelow', ['fwma',
'swma'], ['']);

   if myxup.value [0] > 0 then
   begin
      itself.plot [1] := data1.low [0] - 0.01;
```

```
         itself.successex [2] := false;
      end
      else if myxdown.value [0] > 0 then
      begin
         itself.successex [1] := false;
         itself.plot [2] := data1.high [0] + 0.01;
      end
      else
         itself.successall := false;
   end;
```

VBScript

```
function ex_vbs_waverage()
dim myxup, myxdown
   itself.makeindicator "fwma", "waverage", Array("1"),
Array(param1.str)
   itself.makeindicator "swma", "waverage", Array("1"),
Array(param2.str)

   myxup = itself.makeindicator ("myxu", "xabove", _
                                 Array("fwma", "swma"),
Array(""))
   myxdown = itself.makeindicator ("myxd", "xbelow", _
                                 Array("fwma", "swma"),
Array(""))

   if myxup.value (0) > 0 then
      itself.plot (1) = data1.low (0) - 0.01
      itself.successex (2) = false
   elseif myxdown.value (0) > 0 then
      itself.successex (1) = false
      itself.plot (2) = data1.high (0) + 0.01
   else
      itself.successall = false
   end if
end function
```

Formula

```
myfwma := waverage (data1, param1);
myswma := waverage (data1, param2);
plot1 := if(xabove(myfwma, myswma) > 0, l-0.01, 0);
success1 := if(xabove(myfwma, myswma) > 0, 1, 0);
plot2 := if(xbelow(myfwma, myswma) > 0, h+0.01, 0);
success2 := if(xbelow(myfwma, myswma) > 0, 1, 0);
```

Formula Column

```
waverage(m5, 12)
```

Weight Spread (wspread)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'wspread', ['N','N'],
['N','N']);
```

VBScript

```
Itself.MakeIndicator("string", "wspread", Array("N","N"),
Array("N","N"))
```

Formula Language

```
wspread(dataN, dataN, N, N)
```

Formula Column

```
N/A
```

Definition

Spread analysis with weighting parameter adjustment. Weighted Spread takes two parameters: Weight1 and Weight2. Weighted Spread requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_wspread : double;
var mywso, mywsh, mywsl, mywsc : variant;
begin
   mywso := itself.makeindicator ('wsopen', 'wspread',
                                  ['1.o', '2.o'], [param1.str,
param2.str]);
   mywsh := itself.makeindicator ('wshigh', 'wspread',
                                  ['1.h', '2.h'], [param1.str,
param2.str]);
   mywsl := itself.makeindicator ('wslow', 'wspread',
                                  ['1.l', '2.l'], [param1.str,
param2.str]);
   mywsc := itself.makeindicator ('wsclose', 'wspread',
                                  ['1.c', '2.c'], [param1.str,
param2.str]);
   if not mywso.valid [0] or not mywsh.valid [0] or
      not mywsl.valid [0] or not mywsc.valid [0] then
   begin
```

```
      itself.success := false;
      exit;
   end;
   itself.plot [1] := mywso.value [0];
   itself.plot [2] := mywsh.value [0];
   itself.plot [3] := mywsl.value [0];
   itself.plot [4] := mywsc.value [0];
end;
```

VBScript

```
function ex_vbs_wspread()
dim mywso, mywsh, mywsl, mywsc
   mywso = itself.makeindicator ("wsopen", "wspread", _
                     Array("1.o", "2.o"), Array(param1.str, 
param2.str))
   mywsh = itself.makeindicator ("wshigh", "wspread", _
                     Array("1.h", "2.h"), Array(param1.str, 
param2.str))
   mywsl = itself.makeindicator ("wslow", "wspread", _
                     Array("1.l", "2.l"), Array(param1.str, 
param2.str))
   mywsc = itself.makeindicator ("wsclose", "wspread", _
                     Array("1.c", "2.c"), Array(param1.str, 
param2.str))

   if not mywso.valid (0) or not mywsh.valid (0) or _
      not mywsl.valid (0) or not mywsc.valid (0) then
      itself.success = false
      exit function
   end if

   itself.plot (1) = mywso.value (0)
   itself.plot (2) = mywsh.value (0)
   itself.plot (3) = mywsl.value (0)
   itself.plot (4) = mywsc.value (0)
end function
```

Formula

```
plot1 := wspread(data1.o, data2.o, param1, param2);
plot2 := wspread(data1.h, data2.h, param1, param2);
plot3 := wspread(data1.l, data2.l, param1, param2);
plot4 := wspread(data1.c, data2.c, param1, param2);
```

Formula Column

```
N/A
```

Weighted Index (weighted_index)

Weighted index cannot be called within an indicator, because it counts and use all data series presented in a chart. There is no way to pass such information to NeoTicker® when calling indicators within another indicator.

For detail usage of this indicator, refer to ***Power Indicators Guide>Weighted Index*** (see "Weighted Index" on page 1359).

Wide Range (wr)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'wr', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "wr", Array("N"), Array("N"))
```

Formula Language

```
wr(dataN, N)
```

Formula Column

```
wr(Tn, N)
```

Definition

Wide Range returns 1 when the current bar is the widest range over the Period specified by the parameter. Wide Range requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_wr : double;
var mywr : variant;
begin
   mywr := itself.makeindicator ('mywr', 'wr', ['1'],
[param1.str]);

   if not mywr.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   if mywr.value [0] > 0 then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low [0];
      itself.plot [3] := cllime;
   end
   else
      itself.successall := false;
```

```
end;
```

VBScript

```
function ex_vbs_wr()
dim mywr
   mywr = itself.makeindicator ("mywr", "wr", Array("1"),
Array(param1.str))

   if not mywr.valid (0) then
      itself.successall = false
      exit function
   end if

   if mywr.value (0) > 0 then
      itself.plot (1) = data1.high (0)
      itself.plot (2) = data1.low (0)
      itself.plot (3) = cllime
   else
      itself.successall = false
   end if
end function
```

Formula

```
$mywr := wr(data1, param1);
plot1 := if($mywr > 0, h, 0);
plot2 := if($mywr > 0, l, 0);
plot3 := if($mywr > 0, cllime, 0);
success1 := if ($mywr >0, 1, 0);
success2 := if ($mywr >0, 1, 0);
success3 := if ($mywr >0, 1, 0);
```

Formula Column

```
wr(m5, 7)
```

Wide Range N Bar (wrn)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'wrn', ['N'], ['N', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "wrn", Array("N"), Array("N",
"N"))
```

Formula Language

```
wrn(dataN, N, N)
```

Formula Column

```
wrn(Tn, N, N)
```

Definition

Wide Range N Bars returns 1 when the current N bars is the widest range over the Period specified by the parameter. Wide Range N Bars requires one series (Link 1).

Examples

Delphi Script

```
function ex_del_wrn : double;
var mywr, mywrn : variant;
begin
   mywr := itself.makeindicator ('mywr', 'wr', ['1'], ['7']);
   mywrn := itself.makeindicator ('mywrn', 'wrn', ['1'], ['2',
'7']);

   if not (mywr.valid [0] and mywrn.valid [0]) then
   begin
      itself.successall := false;
      exit;
   end;

   if (mywr.value [0] = mywrn.value [0]) and
      (mywr.value [0] > 0) then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low [0];
      itself.plot [3] := clyellow;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function ex_vbs_wrn()
dim mywr, mywrn
   mywr = itself.makeindicator ("mywr", "wr", Array("1"),
Array("7"))
   mywrn = itself.makeindicator ("mywrn", "wrn", Array("1"),
Array("2", "7"))

   if not (mywr.valid (0) and mywrn.valid (0)) then
      itself.successall = false
      exit function
   end if

   if (mywr.value (0) = mywrn.value (0)) and (mywr.value (0) >
0) then
      itself.plot (1) = data1.high (0)
      itself.plot (2) = data1.low (0)
      itself.plot (3) = clyellow
   else
      itself.successall = false
   end if
end function
```

Formula

```
$mywr := wr(data1, 7);
$mywrn := wrn(data1, 2, 7);
plot1 := if($mywr > 0 and $mywrn > 0, h, 0);
plot2 := if($mywr > 0 and $mywrn > 0, l, 0);
plot3 := if($mywr > 0 and $mywrn > 0, clyellow, 0);
success1 := if($mywr > 0 and $mywrn > 0, 1, 0);
success2 := if($mywr > 0 and $mywrn > 0, 1, 0);
success3 := if($mywr > 0 and $mywrn > 0, 1, 0);
```

Formula Column

```
wrn(m5, 2, 7)
```

William Percent R (percentr)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'percentr', ['N'], ['N']);
```

VBScript

```
Itself.MakeIndicator("string", "percentr", Array("N"),
Array("N"))
```

Formula Language

```
percentr(dataN, N)
```

Formula Column

```
percentr(Tn, N)
```

Definition

William Percent R equals to 100 - Stochastic FastK indicator. William Percent R requires one specified series (Link 1).

Examples

Delphi Script

```
function ex_del_percentr : double;
var mywpr : variant;
begin
   mywpr := itself.makeindicator ('mywpr', 'percentr', ['1'],
[param1.str]);

   if not mywpr.valid [0] then
   begin
      itself.successall := false;
      exit;
   end;

   if (mywpr.value [0] >= 80) then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low [0];
      itself.plot [3] := clred;
   end
   else if (mywpr.value [0] <= 20) then
   begin
      itself.plot [1] := data1.high [0];
      itself.plot [2] := data1.low [0];
      itself.plot [3] := clgreen;
   end
   else
      itself.successall := false;
end;
```

VBScript

```
function ex_vbs_percentr()
dim mywpr
   mywpr = itself.makeindicator ("mywpr", "percentr", _
                              Array("1"), Array(param1.str))

   if not mywpr.valid (0) then
      itself.successall = false
      exit function
   end if

   if mywpr.value (0) > 80 then
      itself.plot (1) = data1.high (0)
      itself.plot (2) = data1.low (0)
      itself.plot (3) = clred
   elseif mywpr.value (0) < 20 then
      itself.plot (1) = data1.high (0)
      itself.plot (2) = data1.low (0)
      itself.plot (3) = clgreen
   else
      itself.successall = false
   end if
end function
```

Formula

```
plot1 := if(percentr(data1, param1) > 80 or
            percentr(data1, param1) < 20, h, 0);
plot2 := if(percentr(data1, param1) > 80 or
            percentr(data1, param1) < 20, l, 0);
plot3 := if(percentr(data1, param1) > 80, clred,
            if(percentr(data1, param1) < 20, clgreen, 0));
success1 := if(percentr(data1, param1) > 80 or
               percentr(data1, param1) < 20, 1, 0);
success2 := if(percentr(data1, param1) > 80 or
               percentr(data1, param1) < 20, 1, 0);
success3 := if(percentr(data1, param1) > 80 or
               percentr(data1, param1) < 20, 1, 0);
```

Formula Column

```
percentr(m5, 7)
```

Zig Zag (zigzag)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'zigzag', ['N'], ['string', 'N',
'string', 'string', 'string', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "zigzag", Array("N"),
Array("string", "N", "string", "string", "string", "N"))
```

Formula Language

```
zigzag(dataN, "string", N, "string", "string", "string", N)
```

Formula Column

```
zigzag(Tn, "string", N, "string", "string", "string", N)
```

Definition

Zig Zag automatically draws trendlines using either percentage-driven or point-driven minimum change specified by the user. Non-confirmed move at the end of the data series is represented by special color for easy identification. Zig Zag returns the swing values in the first plot, swing bar location (bars ago) in the second plot, and the current swing direction in the third plot. Zig Zag requires one data series (Link 1)

Examples

Delphi Script

```
function ex_del_zigzag : double;
var myzz : variant;
begin
   myzz := itself.makeindicator ('zz', 'zigzag', ['1'],
                     ['Percent', '5', 'clBlue', 'clRed',
'clGreen', '2']);

   if heap.size = 0 then
   begin
      heap.allocate (1);
      heap.value [0] := 0;
   end;

   if not myzz.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   if heap.value [0] = 0 then
   begin
      heap.value [0] := myzz.valueex [1, 0];
      itself.success := false;
      exit;
   end;

   if myzz.valueex [1, 0] <> myzz.valueex [1, 1] then
      heap.value [0] := myzz.valueex [1, 1];

   result := heap.value [0] - myzz.valueex [1, 0];
end;
```

VBScript

```
function ex_vbs_zigzag()
dim myzz
   myzz = itself.makeindicator ("zz", "zigzag", Array("1"), _
                  Array("Percent", "5", "clBlue", "clRed",
"clGreen", "2"))

   if heap.size = 0 then
      heap.allocate (1)
      heap.value (0) = 0
   end if

   if not myzz.valid (0) then
      itself.success = false
      exit function
   end if

   if heap.value (0) = 0 then
      heap.value (0) = myzz.value (0)
      itself.success = false
      exit function
   end if
   if myzz.valueex (1, 0) <> myzz.valueex (1, 1) then
      heap.value (0) = myzz.value (1, 1)
   end if
   ex_vbs_zigzag = heap.Value (0) - myzz.valueex (1, 0)
end function
```

Formula

```
myzz := zigzag(data1, "Percent", 5, "clBlue", "clRed",
"clGreen", 2);
$prevzz := if($prevzz = 0 or myzz <> myzz(1), myzz(1),
$prevzz);
plot1 := $prevzz-myzz;
```

Formula Column

```
zigzag(m5, "Percent", 5, "clBlue", "clRed", "clGreen", 2)
```

Zig Zag High (zigzaghigh)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'zigzaghigh', ['N'], ['N',
'string', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "zigzaghigh", Array("N"),
Array("N", "string", "N"))
```

Formula Language

```
zigzaghigh(dataN, N, "string", N)
```

Formula Column

```
zigzaghigh(Tn, N, "string", N)
```

Definition

Zig Zag High returns the Nth High occurrence using either percentage-driven or point-driven minimum change specified by the user. Zig Zag High requires one data series (Link 1)

Examples

Delphi Script

```
function ex_del_zigzaghigh : double;
var myzzh : variant;
begin
   myzzh := itself.makeindicator ('myzzh', 'zigzaghigh', ['1'],
                                   ['1', 'Percent', '5']);

   if heap.size = 0 then
   begin
      heap.allocate (1);
      heap.value [0] := 0;
   end;

   if (heap.value [0] = 0) or (myzzh.value [0] <> myzzh.value
[1]) then
      heap.value [0] := myzzh.value [1];

   result := heap.Value [0]-myzzh.value [0];
```

```
end;
```

VBScript

```
function ex_vbs_zigzaghigh()
dim myzzh
   myzzh = itself.makeindicator ("myzzh", "zigzaghigh", 
Array("1"), _
                                 Array("1", "Percent", "5"))

   if heap.size = 0 then
      heap.allocate (1)
      heap.value (0) = 0
   end if

   if (heap.value (0) = 0) or (myzzh.value (1) <> myzzh.value 
(0)) then
      heap.value (0) = myzzh.value (1)
   end if
   ex_vbs_zigzaghigh = heap.Value (0) - myzzh.value (0)
end function
```

Formula

```
myzzh := zigzaghigh (data1, 1, "Percent", 5);
$prevzzh := if(myzzh <> myzzh(1), myzzh(1), $prevzzh);
plot1 := $prezzh-myzzh;
```

Formula Column

```
zigzaghigh(m5, 1, "Percent", 5)
```

Zig Zag High Bar (zigzaghighbar)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'zigzaghighbar', ['N'], ['N',
'string', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "zigzaghighbar", Array("N"),
Array("N", "string", "N"))
```

Formula Language

```
zigzaghighbar(dataN, N, "string", N)
```

Formula Column

```
zigzaghighbar(Tn, N, "string", N)
```

Definition

Zig Zag High Bar returns the number of bars ago of the Nth High occurrence using either percentage-driven or point-driven minimum change specified by the user. Zig Zag High Bar requires one data series (Link 1)

Examples

Delphi Script

```
function ex_del_zigzaghighbar : double;
var myzzhb : variant;
begin
   myzzhb := itself.makeindicator ('myzzhb', 'zigzaghighbar',
['1'],
                                    ['1', 'Percent', '5']);

   if heap.size = 0 then
   begin
      heap.allocate (1);
      heap.value [0] := 0;
   end;

   if myzzhb.value [0] < myzzhb.value [1] then
      heap.value [0] := data1.high [0];

   if heap.value [0] > 0 then
```

```
      result := heap.Value [0] - data1.high [0]
   else
      itself.success := false;
end;
```

VBScript

```
function ex_vbs_zigzaghighbar()
dim myzzhb
   myzzhb = itself.makeindicator ("myzzhb", "zigzaghighbar",
Array("1"), _
                                  Array("1", "Percent", "5"))

   if heap.size = 0 then
      heap.allocate (1)
      heap.value (0) = 0
   end if

   if myzzhb.value (0) < myzzhb.value (1) then
      heap.value (0) = data1.high (1)
   end if
   ex_vbs_zigzaghighbar = heap.Value (0) - data1.high (0)
end function
```

Formula

```
myzzhb := zigzaghighbar (data1, 1, "Percent", 5);
$myphigh := if (myzzhb < myzzhb(1), data1.high (0), $myphigh);
plot1 := $myphigh-high;
```

Formula Column

```
zigzaghighbar(m5, 1, "Percent", 5)
```

Zig Zag Low (zigzaglow)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'zigzaglow', ['N'], ['N',
'string', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "zigzaglow", Array("N"),
Array("N", "string", "N"))
```

Formula Language

```
zigzaglow(dataN, N, "string", N)
```

Formula Column

```
zigzaglow(Tn, N, "string", N)
```

Definition

Zig Zag Low returns the Nth Low occurrence using either percentage-driven or point-driven minimum change specified by the user. Zig Zag Low requires one data series (Link 1)

Examples

Delphi Script

```
function ex_del_zigzaglow : double;
var myzzl : variant;
begin
   myzzl := itself.makeindicator ('myzzl', 'zigzaglow', ['1'],
                                  ['1', 'Percent', '5']);

   if heap.size = 0 then
   begin
      heap.allocate (1);
      heap.value [0] := 0;
   end;

   if not myzzl.valid [0] then
   begin
      itself.success := false;
      exit;
   end;
   if heap.value [0] = 0 then
      heap.value [0] := myzzl.value [0];

   if myzzl.value [0] <> myzzl.value [1] then
      heap.value [0] := myzzl.value [1];
   result := heap.Value [0] - myzzl.value [0];
end;
```

VBScript

```
function ex_vbs_zigzaglow()
dim myzzl
   myzzl = itself.makeindicator ("myzzl", "zigzaglow",
Array("1"), _
                                 Array("1", "Percent", "5"))

   if heap.size = 0 then
      heap.allocate (1)
      heap.value (0) = 0
   end if

   if not myzzl.valid (0) then
      itself.success = false
      exit function
   end if

   if heap.value (0) = 0 then
      heap.value (0) = myzzl.value (0)
   end if

   if myzzl.value (0) <> myzzl.value (1) then
      heap.value (0) = myzzl.value (1)
   end if

   ex_vbs_zigzaglow = heap.Value (0) - myzzl.value (0)
end function
```

Formula

```
myzzl := zigzaglow (data1, 1, "Percent", 5);
$myprevl := if (myzzl <> myzzl(1), myzzl(1), $myprevl);
plot1 := $myprevl-myzzl;
```

Formula Column

```
zigzaglow(m5, 1, "Percent", 5)
```

Zig Zag Low Bar (zigzaglowbar)

Syntax

Delphi Script

```
Itself.MakeIndicator('string', 'zigzaglowbar', ['N'], ['N',
'string', 'N']);
```

VBScript

```
Itself.MakeIndicator("string", "zigzaglowbar", Array("N"),
Array("N", "string", "N"))
```

Formula Language

```
zigzaglowbar(dataN, N, "string", N)
```

Formula Column

```
zigzaglowbar(Tn, N, "string", N)
```

Definition

Zig Zag Low Bar returns the number of bars ago of the Nth Low occurrence using either percentage-driven or point-driven minimum change specified by the user. Zig Zag Low Bar requires one data series (Link 1)

Examples

Delphi Script

```
function ex_del_zigzaglowbar : double;
var myzzlb : variant;
begin
   myzzlb := itself.makeindicator ('myzzlb', 'zigzaglowbar',
['1'],
                                                ['1', 'Percent',
'5']);

   if heap.size = 0 then
   begin
      heap.allocate (1);
      heap.value [0] := 0;
   end;

   if not myzzlb.valid [0] then
   begin
      itself.success := false;
      exit;
   end;

   if myzzlb.value [0] < myzzlb.value [1] then
      heap.value [0] := data1.low [myzzlb.value [0]];

   if heap.value [0] = 0 then
      itself.success := false
   else
      result := heap.value [0] - Data1.low [0];
end;
```

VBScript

```
function ex_vbs_zigzaglowbar()
dim myzzlb
   myzzlb = itself.makeindicator ("myzzlb", "zigzaglowbar",
Array("1"), _
                                    Array("1", "Percent", "5"))

   if heap.size = 0 then
      heap.allocate (1)
      heap.value (0) = 0
   end if

   if not myzzlb.valid (0) then
      itself.success = false
      exit function
   end if

   if myzzlb.value (0) < myzzlb.value (1) then
      heap.value (0) = data1.low(myzzlb.value(1))
   end if

   if heap.value (0) = 0 then
      itself.success = false
   else
      ex_vbs_zigzaglowbar = heap.value (0) - Data1.low (0)
   end if
end function
```

Formula

```
myzzlb := zigzaglowbar(data1, 1, "Percent", 5);
$prevl := if(myzzlb < myzzlb(1), low(myzzlb(1)), $prevl);
plot1 := $prevl-low;
```

Formula Column

```
zigzaglowbar(m5, 1, "Percent", 5)
```

Index