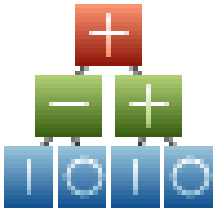


# Adaptrade Builder

## Version 1

# User's Guide

# Disclaimer

HYPOTHETICAL OR SIMULATED PERFORMANCE RESULTS HAVE CERTAIN INHERENT LIMITATIONS. UNLIKE AN ACTUAL PERFORMANCE RECORD, SIMULATED RESULTS DO NOT REPRESENT ACTUAL TRADING. ALSO, SINCE THE TRADES HAVE NOT ACTUALLY BEEN EXECUTED, THE RESULTS MAY HAVE UNDER- OR OVER-COMPENSATED FOR THE IMPACT, IF ANY, OF CERTAIN MARKET FACTORS, SUCH AS LACK OF LIQUIDITY. SIMULATED TRADING PROGRAMS IN GENERAL ARE ALSO SUBJECT TO THE FACT THAT THEY ARE DESIGNED WITH THE BENEFIT OF HINDSIGHT. NO REPRESENTATION IS BEING MADE THAT ANY ACCOUNT WILL OR IS LIKELY TO ACHIEVE PROFITS OR LOSSES SIMILAR TO THOSE SHOWN.



Last Revision: September 2011 (version 1.2.2.0)

# Software License Agreement

These license terms are an agreement between Adaptrade Software and you. Please read them. They apply to the software named above, which includes the media on which you received it, if any. The terms also apply to any

- updates,
- supplements, including EasyLanguage code files for TradeStation, and
- support services

for this software provided by Adaptrade Software, unless other terms accompany those items. If so, those terms apply.

BY CLICKING ON THE "I AGREE" BUTTON WHERE INDICATED, OR BY COPYING, INSTALLING OR OTHERWISE USING THE SOFTWARE, YOU ACCEPT THESE TERMS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT INSTALL OR USE THE PROGRAM AND DESTROY ALL COPIES OF IT.

If you comply with these license terms, you have the rights below.

1. LICENSE MODEL. The software is licensed on a per user basis.
2. INSTALLATION AND USE RIGHTS. You may install and use any number of copies of the software on your devices, provided it is for your use only.
3. SCOPE OF LICENSE. The software is licensed, not sold. This agreement only gives you some rights to use the software. Adaptrade Software reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the software only as expressly permitted in this agreement. You may not
   - reverse engineer, decompile or disassemble the software, except and only to the extent that applicable law expressly permits, despite this limitation;
   - make more copies of the software than specified in this agreement or allowed by applicable law, despite this limitation;
   - publish the software for others to copy;
   - rent, lease or lend the software;
4. BACKUP COPY. You may make two backup copies of the software. You may use these copies only to reinstall the software.
5. EXPORT RESTRICTIONS. The software is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the software. These laws include restrictions on destinations, end users and end use.
6. SUPPORT SERVICES. Support services are as described on the Adaptrade Software web site, www.Adaptrade.com.
7. ENTIRE AGREEMENT. This agreement, and the terms for supplements, updates and support services that you use, are the entire agreement for the software and support services.
8. APPLICABLE LAW.
   a. United States. If you acquired the software in the United States, California state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.
   b. Outside the United States. If you acquired the software in any other country, the laws of that country apply.
9. LEGAL EFFECT. This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the software. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.
10. DISCLAIMER OF WARRANTY. THE SOFTWARE IS LICENSED "AS-IS." YOU BEAR THE RISK OF USING IT. ADAPTRADE SOFTWARE GIVES NO EXPRESS WARRANTIES, GUARANTEES OR CONDITIONS. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, ADAPTRADE SOFTWARE EXCLUDES THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
11. LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. YOU CAN RECOVER FROM ADAPTRADE SOFTWARE ONLY DIRECT DAMAGES UP TO THE AMOUNT PAID FOR THE SOFTWARE. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.

    This limitation applies to
    a. anything related to the software, services, content (including code) on third party Internet sites, or third party programs; and
    b. claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

    It also applies even if Adaptrade Software knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

# Table of Contents

# Chapter 1
## Introduction

### Overview

Adaptrade Builder is a stand-alone Window program that automatically generates trading strategies for TradeStation and MultiCharts. In effect, Builder automates the traditional, manual approach to strategy development in which the trader selects elements of a trading strategy based on prior experience combined with knowledge of technical indicators, entry and exit order types, and strategy design. In the traditional method, a strategy is based on a market "hypothesis"; that is, an idea of how the market works. A viable trading strategy is typically developed through a long trial-and-error process involving numerous iterations, revisions, and testing until acceptable results are achieved.

Builder performs each step of this process automatically. The program generates an initial population of trading strategies by randomly selecting the trading rules and entry and exit order types for each member of the population. The initial population of strategies is then "evolved" over successive generations using a genetic programming algorithm, guided by performance criteria selected by the user. The program builds the strategies over the training or "in-sample" data segment and tests each one over the test or "out-of-sample" data. Each candidate strategy is essentially a hypothesis that is either supported or refuted by the out-of-sample testing.

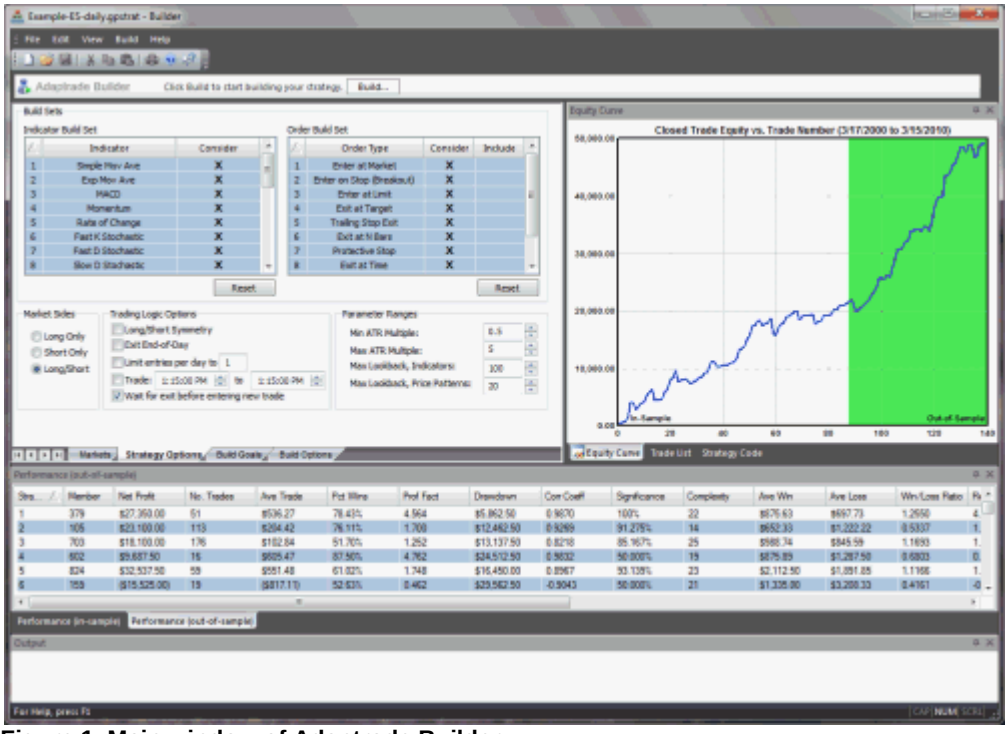


**Figure 1. Main window of Adaptrade Builder.**

The premise behind Builder is that developing a trading strategy is essentially a problem of *statistical inference*. The price data can be thought of as a combination of "signal" and "noise". The signal is the tradable part of the data, and the noise is everything else. In this context, the essential challenge is finding strategies that fit the signal while ignoring the noise and avoiding over-fitting. At the same time, market data is often non-stationary: the statistical properties change over time. A successful strategy is therefore one that fits the stationary elements of the market signal with adequate degrees-of-freedom to avoid over-fitting. Out-of-sample testing is used to verify each of these requirements.

Builder is designed to generate strategies for almost any market and time frame, from tick data to monthly bars, for stocks, futures, forex, ETFs, and other markets. The strategies generated by Builder are complete trading strategies, including rules and trading orders for entering the market, exiting at a profit, and exiting at a loss. The strategy code is provided in open text file format, which can be pasted into the EasyLanguage editor and run in either the TradeStation platform or in MultiCharts.

Some of the user options in Builder include generating code for either current versions of TradeStation or TradeStation 2000i; specifying strategies as long-only, short-only, or combined long and short trading; requiring the long and short entry rules to be logical opposites; excluding specific indicators, entry rules, and exit rules from the build process; and specifying various aspects of the genetic programming process as well as other features to be included or excluded from the generated strategies.

## Genetic Programming

Builder uses a computational technique called genetic programming (GP),[1] which belongs to a class of techniques called evolutionary algorithms. Evolutionary algorithms and GP in particular were developed by researchers in artificial intelligence based on the biological concepts of reproduction and evolution. A GP algorithm "evolves" a population of trading strategies from an initial population of randomly generated members. Members of the population compete against each other based on their "fitness." The fitter members are selected as "parents" to produce a new member of the population, which replaces a weaker (less fit) member.

Two parents are combined using a technique called crossover, which mimics genetic crossover in biological reproduction. In crossover, part of one parent's genome is combined with part of the other parent's genome to produce the child genome. In Builder, genomes represent the trading rules and order logic of the strategy.

Other members of the population are produced via mutation, is which one member of the population is selected to be modified by randomly changing parts of its genome. Typically, a majority (e.g., 90%) of new members of the population are produced via crossover, with the remaining members produced via mutation.

Over successive generations of reproduction, the overall fitness of the population tends to increase. The process is stopped after some number of generations or when the fitness stops increasing. The solution is generally taken as the fittest member of the resulting population.

The initial GP population might have as few as 50 members or as many as 1000 or more. A typical build process might progress over anywhere from 10 to 100 generations or more. The number of strategies constructed and evaluated during the build process is equal to the size of the population multiplied by the number of generations.

In the context of building trading strategies, GP enables the synthesis of strategies given only a high level set of performance goals. The GP process does the rest. This approach has several significant benefits, including:

- Reduces the need for knowledge of technical indicators and strategy design. The GP algorithm selects the individual trading rules, indicators, and other elements of the strategy for you.
- The rule construction process allows for considerable complexity, including nonlinear trading rules.
- The GP process eliminates the most labor intensive and tedious elements of the traditional strategy development process; namely, coming up with a new trading idea, programming it, verifying the code, testing the strategy, modifying the code, and repeating. This is all done automatically in GP.
- The GP process is unbiased. Whereas most traders have developed biases for or against specific indicators and/or trading logic, GP is guided only by what works.
- By incorporating proper trading rule semantics, the GP process in Builder is designed to produce logically correct trading rules and error-free code.
- The GP process often produces results that are not only unique but non-obvious. In many cases, these *hidden gems* would be nearly impossible to find any other way.
- By automating the build process, the time required to develop a viable strategy can be reduced from weeks or months to a matter of minutes in some cases, depending on the length of the input price data file and other build settings.

Genetic programming has been successfully used in a variety of fields, including signal and image processing, process control, bioinformatics, data modeling, programming code generation, computer games, and economic modeling; see, for example Poli et al.[2] An overview of using GP in finance is provided by Chen.[3] Colin[4] was one of the first to explain how to use GP for optimizing combinations of rules for a trading strategy.

Various academic studies have demonstrated the benefits of GP in trading. For example, Karjalainen[5] found that price pattern trading rules evolved using GP for S&P 500 futures provided an advantage over buy-and-hold returns in out-of-sample testing. Similarly, Potvin et al.[6] found that rules generated through a GP process for individual stocks outperformed buy-and-hold in out-of-sample testing during falling and sideways markets. Kaucic[7] combined a genetic algorithm with other learning methods to generate simple trading rules for the S&P 500 index and found positive results compared to buy-and-hold on out-of-sample testing.

Until recently, most applications of genetic programming to trading strategy generation have been academic studies based on limited rule sets, overly simple entry and exit logic, and custom-written code, making the results unsuitable for most traders. At the same time, most available software that implements GP for market trading has either been targeted to professional traders and priced accordingly or is very complicated to set up and use. Adaptrade Builder was designed to make GP simple to use for any trader, individual or professional, who has a basic understanding of strategy trading and the TradeStation or MultiCharts platform.

### References

1. J. Koza. Genetic Programming. The MIT Press, Cambridge, MA. 1992.

2. R. Poli, W. B. Langdon, and N. F. McPhee. A field guide to genetic programming. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. (With contributions from J. R. Koza).

3. Shu-Heng Chen (Editor). Genetic Algorithms and Genetic Programming in Computational Finance. Kluwer Academic Publishers, Norwell, MA. 2002.
4. A. Colin. Genetic algorithms for financial modeling, Trading on the Edge. 1994, Pages 165-168. John Wiley & Sons, Inc. New York.

5. Risto Karjalainen. Evolving technical trading rules for S&P 500 futures, Advanced Trading Rules, 2002, Pages 345-366. Elsevier Science, Oxford, UK.

6. Jean-Yves Potvin, Patrick Soriano, Maxime Vallee. Generating trading rules on the stock markets with genetic programming. Computers & Operations Research, Volume 31, Issue 7, June 2004, Pages 1033-1047.

7. Massimiliano Kaucic. Investment using evolutionary learning methods and technical rules. European Journal of Operational Research, Volume 207, Issue 3, 16 December 2010, Pages 1717-1727.

# Build Algorithm

The build algorithm used in Adaptrade Builder is illustrated below. The gray-shaded boxes represent the input data, which includes the price data for the market of interest, the indicators and order types in the so-called *build set*, and the options and performance criteria (build goals) selected by the user.

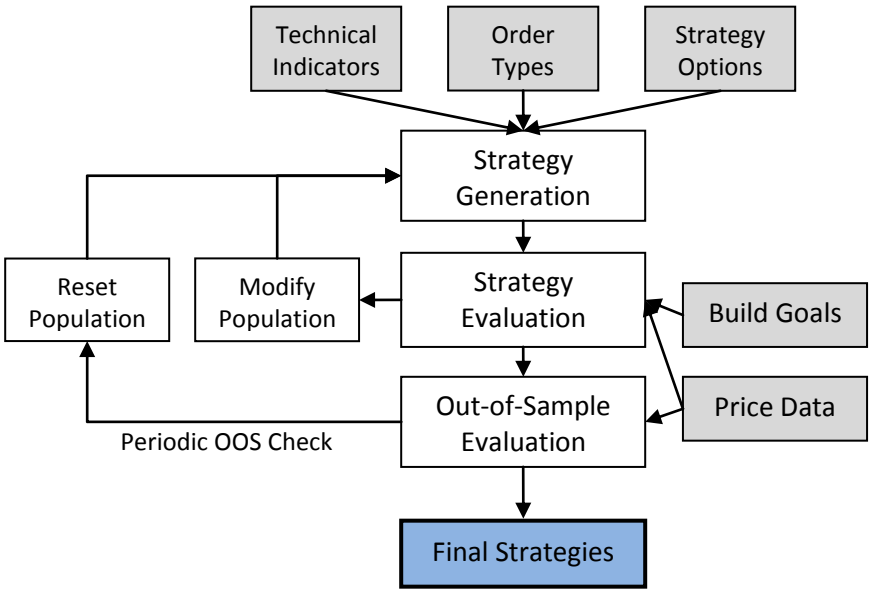



**Figure 2. Build algorithm in Adaptrade Builder.**

The algorithm starts with the Strategy Generation step. An initial population of trading strategies is randomly developed from the available technical indicators and rule types in the build set. Any options that the user has selected, such as exiting all positions at end-of-day, are applied at this point. Each strategy is then evaluated over the price data for the market of interest, and a fitness value is assigned based on a weighted average of the build goals

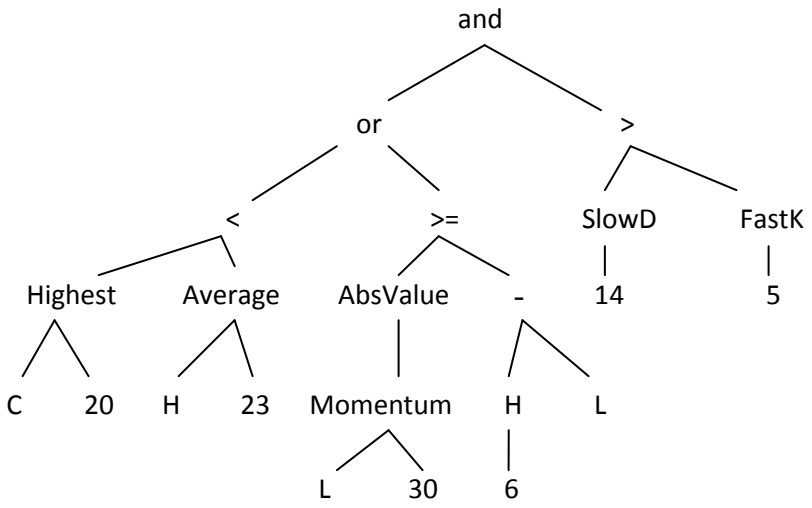
specified by the user. For example, you might select net profit and drawdown as the two performance metrics and weight each one equally. The fitness would then be the average of the net profit and drawdown.

To generate new members of the population, members of the current population are selected at random, and the fitter ones are chosen as parents for crossover and mutation. A less fit member is selected at random to be replaced by the new member. The process is repeated until as many new members have been created as there are members in the current population. This step represents one generation.

If the user has selected the Reset on Out-of-Sample Performance option, the out-of-sample results for the top strategy are checked after each specified number of generations. If the results are not above the threshold chosen by the user, the process is reset, which causes the population to be re-initialized and the generation count to be reset to zero. After the specified number of generations has been successfully completed, the top strategies are listed in the Performance table in order of fitness.

# Entry and Exit Conditions

The GP process evolves three essential strategy elements simultaneously: entry conditions, exit conditions, and orders for entry and exit. The entry and exit conditions are represented as tree structures, as shown below in Fig. 3.




((Highest(C, 20) < Average(H, 23)) or (AbsValue(Momentum(L, 30)) >= H[6] – L)) and (SlowD(14) > FastK(5))

**Figure 3. Entry condition example, showing tree structure and corresponding EasyLanguage code.**

The tree structure enables the generation of entry and exit conditions with considerable complexity. Each node in the tree has between zero and three inputs, each of which leads to further branching. The tree is constructed recursively starting at the top with a logical operator (and, or, >, <, etc.) and proceeding to technical indicator functions, prices, and constants, such as indicator lengths. Each branch is terminated with a node that has no inputs. Available indicators in Builder are listed in Table 1, below. These are described in detail in the appendix.

**Table 1. Available Indicators in Builder**

| | |
|---|---|
| Simple moving average | True range (TR) |
| Exponential moving average | Average true range (ATR) |
| Moving average convergence divergence (MACD) | Lowest |
| Momentum | Highest |
| Rate of change (ROC) | Volume |
| Fast K stochastic | Accumulation/distribution |
| Fast D stochastic | Chaiken oscillator |
| Slow D stochastic | Crosses Above/Below |
| Relative strength indicator (RSI) | Price patterns (O, H, L, C, O[N], etc.) |
| Directional indicator (DI+/DI-) | Day of week |
| Directional movement index (DMI) | Time of day |
| Average directional index(ADX) | Absolute value |

The crossover operator of the GP process replaces a subtree in one parent with a subtree from the other parent. For example, the subtree on the right of Fig. 3, starting with ">" (i.e., SlowD(14) > FastK(5)), might be replaced with a different subtree from another member of the population. Mutation changes individual nodes in the tree. For example, the "Average" node might be replaced with "Lowest" so that the subtree Average(H, 23) becomes Lowest(H, 23).

An entry or exit condition is evolved separately for long and short trades unless the user selects "Long only", "Short only", or "Long/Short Symmetry". Each condition is a logical statement; it evaluates to either true or false. A value of true means the condition is satisfied for that market side (long or short), which is necessary for the order to be placed. Entry conditions are applied to all types of entry orders. In other words, the entry condition must be true for the entry order to be placed. Exit conditions only apply to market exit orders. If the exit condition is true, the trade is exited at market on the next bar, assuming the strategy includes the market exit order type.

In order to generate meaningful entry conditions, Builder applies both *syntactic* and *semantic* rules when building the conditions. Syntactic rules ensure that each node containing a function satisfies the input requirements for the function. For example, the Momentum function requires a price as the first input and a length as the second input. Semantic rules ensure that comparisons between different nodes are meaningful. For example, it makes sense to compare the Highest(C, 20) to a moving average since both functions return a price. However, it would not be meaningful to compare the closing price to the time of day or to compare a stochastic, which has a value between 0 and 100, to a moving average of price. The semantic rules enforce these requirements.

According to the semantic rules, indicators that return a price or volume-based value can be used as input to indicators that take price or volume as an input. Examples of price-based indicators include Average, XAverage, Highest, Lowest, ROC, RSI, Momentum, and MACD. For example, Builder can produce conditions that include statements such as Average(Highest(XAverage(C, N1), N2), N3).

# Order Types

The following types of **entry orders** are available in Builder:

- Market Entry
- Stop Entry

- Limit Entry

A market entry means the trade enters at the open of the next bar. Stop and limit entries are placed at a specified price away from the market. Stop orders are intended to be placed above the market for a long entry and below the market for a short entry. Limit orders are intended to be placed below the market for a long entry and above the market for a short entry.

Stop and limit entry prices in Builder are calculated as follows:

```
EntryPrice = PriceValue +/- Fr * PriceDiff
```

where:

`PriceValue` is one of: price, price[N], Highest(price, N), Lowest(price, N), Average(price, N), XAverage(price, N), or DayPrice;
price is one of: O, H, L, or C;
price[N] is the price N bars ago;
DayPrice is one of: OpenD(0), HighD(0), LowD(0), or CloseD(1);
`Fr` is a constant multiplier; and
`PriceDiff` is one of: true range, ATR, $\text{PriceValue}_1 - \text{PriceValue}_2$, or $\text{AbsValue}(\text{PriceValue}_1 - \text{PriceValue}_2)$.

The listed functions, such as OpenD(0) and Highest(price, N), are shown as written in TradeStation's EasyLanguage. PriceValue, Fr, PriceDiff, and the associated function parameters are chosen by Builder during the build process.

The following are examples of long stop entry prices:

```
EntryPrice = Average(C, 10) + 3.5 * AbsValue(C[5] - H[14])

EntryPrice = H + 2.1 * AbsValue (Average(C, 20) - Lowest(H, 15))
```

These could also be short limit entries since short limit entries are also above the market and therefore use a "+" sign to add the price difference to the price value.

Examples of short stop entry prices are shown below:

```
EntryPrice = OpenD(0) - 1.7 * AvgTrueRange (11)

EntryPrice = C[16] - 4.3 * AbsValue(XAverage(L, 5) - Xaverage(C, 2))
```

These could also be long limit entries since long limit entries are also below the market and therefore use a "-" sign to subtract the price difference from the price value.

The following types of **exit orders** are available in Builder:

- Target Exit
- Trailing Stop Exit
- Exit N Bars From Entry
- Money Management (Protective) Stop
- Exit at Time-of-Day
- Exit at Market
- Exit at End-of-Day

Target and protective stop exits are constructed in much the same way as stop and limit entry orders. Specifically, the exit prices for these orders are calculated as follows:

```
ExitPrice = EntryPrice -/+ Fr * PriceDiff
```

where `EntryPrice` is the entry price for the trade, and `Fr` and `PriceDiff` are as defined above.

The following is an example of the exit price for a long money management stop:

```
ExitPrice = EntryPrice – 4.3 * AbsValue(C[10] – Xaverage(C, 2))
```

This could also be the exit price for a short target exit. Similarly, if the first minus sign were changed to a plus sign, this could be the exit price for a short money management stop or a long target exit.

Trailing stops in Builder are activated when the open profit on a closed-bar basis is above a threshold equivalent to a multiple of the average true range. Once the threshold has been reached, the trailing stop is placed so that a percentage of the open profit is locked in. The stop remains active until the trade exits. The multiple of the ATR and the percentage of profit to lock in are inputs chosen by Builder during the build process.

The “Exit N Bars From Entry” exit order causes the trade to exit at the open of the next bar when the number of bars since entry is greater than or equal to N, which is a strategy input chosen by Builder.

The “Exit at Time-of-Day” exit causes the trade to exit at the open of the next bar when the time of the current bar is greater than or equal to an input time chosen by Builder. For example, the exit order might read “If time >= 1030 then sell next bar at market”.

The “Exit at Market” exit causes the trade to exit at the open of the next bar when the exit condition is true on the current bar.

The “Exit at End-of-Day” exit causes the trade to exit at the close of the last bar of the current day if intraday or daily bars are used. On weekly or monthly data, this exit causes the trade to exit at the close of the current bar.

## Trading Strategy Structure

Trading strategies in Builder have the following general form, shown below in pseudo-code:*

```
Inputs: N1, N2, N3, …

LongEntryCondition = …

ShortEntryCondition = …

LongExitCondition = …

ShortExitCondition = …


If [position is flat and] LongEntryCondition is true then
```

```
      Long entry order…
      Initialize long exit orders as necessary…

If [position is flat and] ShortEntryCondition is true then
      Short entry order…
      Initialize short exit orders as necessary…

If position is long then
      Long exit order 1…
      Long exit order 2…
      …

If position is short then
      Short exit order 1…
      Short exit order 2…
      …

[End-of-day exit]
```

* Code shown in brackets [] is optional.

Strategies in Builder start with the list of inputs. An input is provided for any indicator parameter, price pattern look-back length, and any parameters required by the entry and exit orders, such as the look-back length for the ATR.

The LongEntryCondition through ShortExitCondition variables are the true/false conditions evolved by the genetic programming process, such as shown in Fig. 3. A long entry order is placed if the long entry condition is true, subject to the optional condition that the position is currently flat (out of the market). Likewise, a short entry order is placed if the short entry condition is true, subject to the optional condition that the position is currently flat. An open long trade is exited at the next open if the long exit condition is true on the current bar. An open short trade is covered (exited) at the next open if the short exit condition is true is on the current bar.

Only one type of entry order is allowed for each side of the market (long/short), although they can be different for each side unless the symmetry option is selected. When an entry order is placed, one or more variables for the exit orders may be initialized within the entry order code block.

The statements for the exit orders follow the entry orders. One or more exit orders may be chosen by the program, with a maximum of one exit order of each type listed above. Unless the necessary orders have been excluded from the build set by the user, the program will ensure that each strategy has an exit-at-a-loss and an exit-at-a-profit. This prevents trades from remaining open indefinitely.

The optional end-of-day exit must be specified by the user on the Strategy Options tab in order to be included in the strategy.

## Example

As an introductory example, Builder was run on daily bars of a stock index futures market for a small population and a limited number of generations. The performance metrics chosen to guide the process are shown below in Fig. 4. These settings imply that the fitness function was a weighted average of the net profit, number of trades, correlation coefficient, statistical

significance, and the return/drawdown ratio. Specific targets were set for the number of trades and the return/drawdown ratio. The other selected metrics were maximized.

The build options were set mostly to the defaults, as shown in Fig. 5. The population size was set to 100 and the number of generations to five. All members of the population were saved ("Save 100 best strategies"), and the Reset on Build checkbox was unchecked. This means that when the build stops after five generations, the build process will continue where it stopped when the Build button is clicked again. Specifically, the program will initialize the new population with the prior population, rather than starting over. That way, if the build process is progressing well, the population can be evolved beyond the initial five generations.


**Figure 4. Build metrics for example build.**

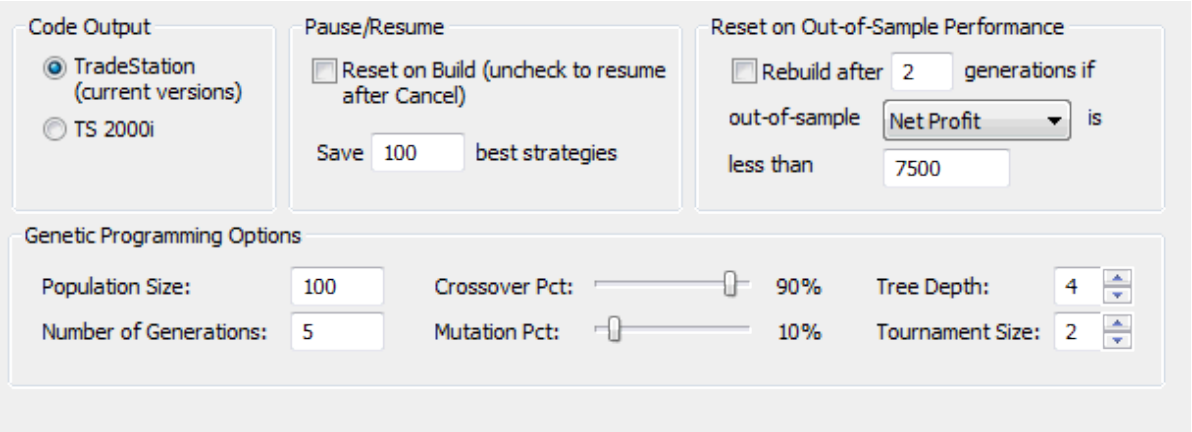



**Figure 5. Build options for example build.**

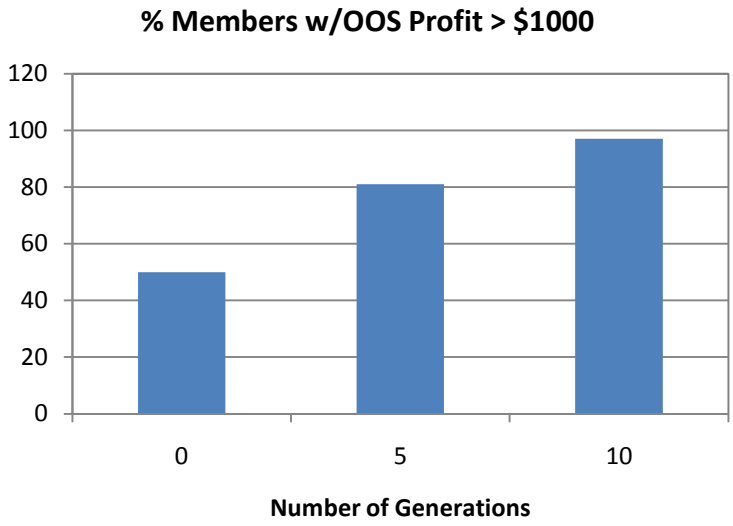



**Figure 6. Percentage of population members with out-of-sample net profit greater than $1,000.**

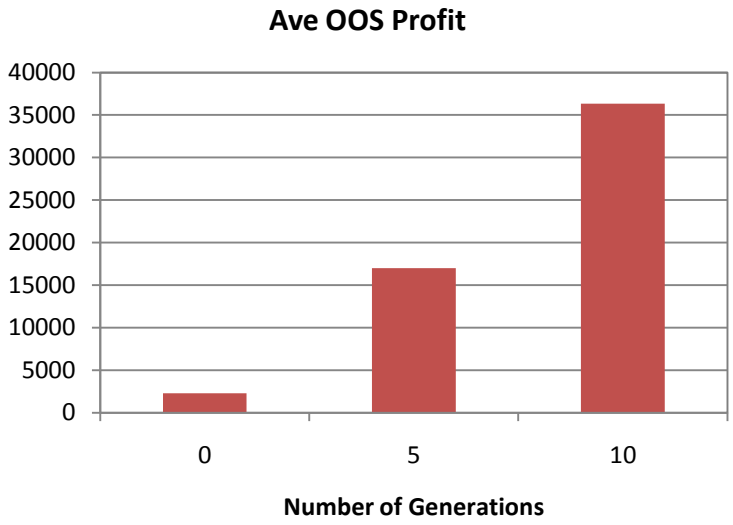



**Figure 7. Average out-of-sample net profit of population members.**

The in-sample/out-of-sample division of data was set to 80% in-sample and 20% out-of-sample (OOS), with the OOS period following the in-sample period. The build process was run over a total of 10 generations. To illustrate how the results evolved during the build, the OOS net profit was recorded after the initial population was generated and after five and 10 generations. Fig. 6 demonstrates that the number of members of the population with OOS net profits of at least $1,000 increases after five and 10 generations.

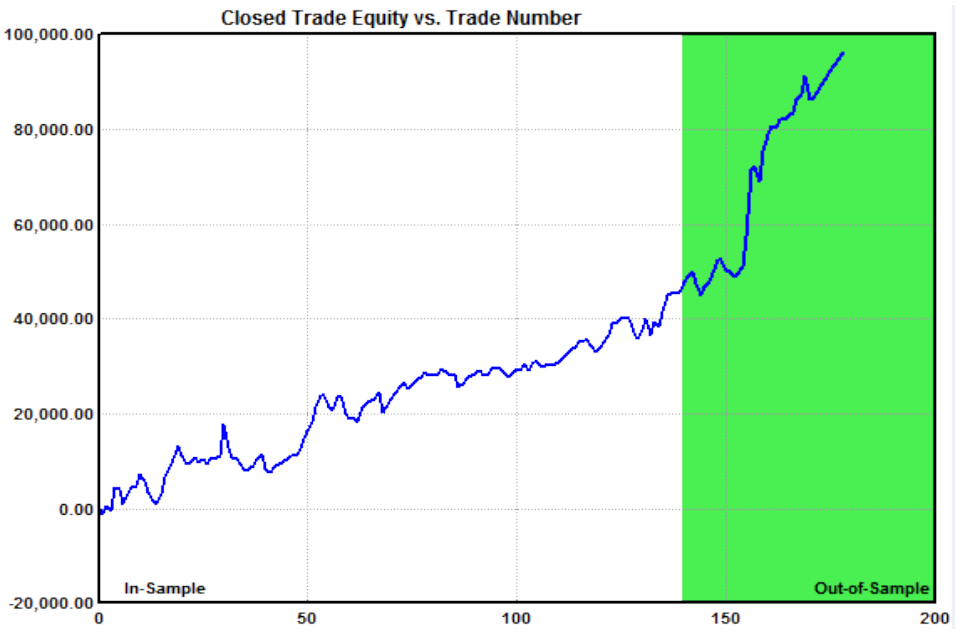



**Figure 8. Closed trade equity curve after 10 generations.**

Similarly, the average OOS net profit of the population increases after five and 10 generations, as shown in Fig. 7. Note that these results are for the OOS net profit. By definition, the out-of-sample data is not used in the build, so OOS results are unbiased; they don't benefit from hindsight. This implies that the GP process not only tends to improve the in-sample results over successive generations, which is a direct effect of the GP algorithm, but the OOS results also tend to improve as the strategies are evolved.

The equity curve for one of the top strategies is shown above in Fig. 8 after 10 generations. Finally, the EasyLanguage (TradeStation) code for the corresponding strategy is listed below.

```
{
 EasyLanguage Strategy Code for TradeStation
 Population member: 46

 Created by: Adaptrade Builder version 1.1.0.0
 Created:    10/19/2010 2:19:52 PM

 TradeStation code for TS 6 or newer

 Price File:  C:\TestData.txt
 Build Dates:
}

{ Strategy inputs }
Inputs: NL1 (74),
        NL2 (20),
        NL3 (85),
        NBarEnL1 (59),
        NATREnL (84),
        EntFrL (3.8189),
        NATRTargL (57),
        TargFrL (1.6168),
        NBarExL (100),
        NBarEnS1 (40),
        NBarEnS2 (49),
        NBarEnS3 (7),
        EntFrS (0.6971),
        NBarExS (6),
```

```
        NATRTrailS (33),
        ATRFrTrailS (1.4126),
        TrailPctS (50.0000);

{ Variables for average true range for entry and exit orders }
Var:    ATREnL (0),
        ATRTargL (0),
        ATRTrailS (0);

{ Variables for money management and/or trailing stop exit orders }
Var:    SStop (0),
        NewSStop (0),
        STrailOn (false);

{ Variables for entry conditions }
Var:    EntCondL (false),
        EntCondS (false);

{ Average true range }
ATREnL = AvgTrueRange(NATREnL);
ATRTargL = AvgTrueRange(NATRTargL);
ATRTrailS = AvgTrueRange(NATRTrailS);

{ Entry conditions }
EntCondL = (Highest(Volume, NL1) >= Lowest(Volume, NL2)) or (Volume < Average(Volume,
NL3));

EntCondS = true;

{ Entry orders }
If MarketPosition = 0 and EntCondL then begin
   Buy next bar at XAverage(L, NBarEnL1) + EntFrL * ATREnL stop;
end;

If MarketPosition = 0 and EntCondS then begin
   Sell short next bar at Highest(H, NBarEnS1) - EntFrS * AbsValue(Lowest(L,
NBarEnS2) - Lowest(H, NBarEnS3)) stop;
   STrailOn = false;
   SStop = Power(10, 10);
end;

{ Exit orders, long trades }
If MarketPosition > 0 then begin

   If BarsSinceEntry >= NBarExL then
      Sell next bar at market;

   Sell next bar at EntryPrice + TargFrL * ATRTargL limit;
end;

{ Exit orders, short trades }
If MarketPosition < 0 then begin

   If EntryPrice - C > ATRFrTrailS * ATRTrailS then
      STrailOn = true;

   If STrailOn then begin
      NewSStop = EntryPrice - TrailPctS * (EntryPrice - C)/100.;
      SStop = MinList(SStop, NewSStop);
   end;

   If BarsSinceEntry >= NBarExS then
      Buy to cover next bar at market;

   If STrailOn then
      Buy to cover next bar at SStop stop;
end;
```

# Chapter 2
## Getting Started

### Installation

**Minimum System Requirements:**

1.2 GHz processor
512 MB RAM
1 GB hard disk space
Windows XP, Vista or Windows 7 operating system
Monitor: 17 inch or larger

Adaptrade Builder can be downloaded at any time from the download page of the Adaptrade Software web site (www.Adaptrade.com). The program initially installs as a trial, which can be activated (i.e., converted to a licensed copy) following purchase. The download file is a self-extracting installation file. To install Builder, browse to the location of the downloaded file via Windows Explorer (also known as "My Computer") and double-click on the file to open it. Alternatively, select Run from the Accessories menu under the list of programs in the Start menu, browse to the location of the downloaded file, click Open, then click OK in the Run window. The installation should begin.

The installation program will prompt you for the location to install the program files. The default location is the folder Program Files\Adaptrade Software\Adaptrade Builder x.x\, where x.x is the version number (e.g., 1.2). You can install the program in another location if you wish.

When installing a new version of Builder on a computer than currently has a prior version installed, please note the following:

- It's usually unnecessary to uninstall a prior version of Builder before installing a newer one. However, if the version number is the same through the second decimal (e.g. 1.2.1 and 1.2.2 are both version 1.2), the newer version will be installed in the same folder as the current version by default and will over-write the older version, making it impossible to use the older version.
- Provided the installation is on the same computer as the prior installation, no new activation code should be required. The new version should install already activated. If not, it may be necessary to enter your license ID and password, as provided on your purchase receipt. If you need an additional activation, please contact Adaptrade Software.
- Uninstalling an older version will not affect any Builder files (.gpstrat files) you may have created or saved.
- New versions of Builder are designed to read files (.gpstrat files) from prior versions. However, once a file is saved in the new version of Builder, you will not be able to open it in an older version.
- The window layout stored with a currently installed version of Builder is not removed when that version is uninstalled. This means that if a new version, such as 1.1.0, has a different window layout than the prior installed version, the layout may need to be adjusted when the program is first run. All panes in Builder can be moved and resized by clicking and dragging.

**Note about 32 vs. 64-bit versions:** The 32-bit version of Builder will run on both 32 and 64-bit versions of Windows. However, the 64-bit version will be able to take advantage of all available installed memory to process larger data files. If you're not sure which version of Windows you have, you can check the System settings under the Control Panel. Both the 32 and 64-bit versions are compiled from the same code and are functionally identical, including the project (.gpstrat) files they generate. Because they're the same program, only one may be installed on the same computer. If your computer runs 64-bit Windows, it's strongly recommended that you install the 64-bit version; otherwise, it will be necessary to install the 32-bit version.

Once the installation is complete, you should find the Builder icon, as displayed on the title page of the user's guide, on your desktop and the Builder program in the programs menu. You should also find a folder called Examples in the Adaptrade Builder folder. The Examples folder contains sample files that can be opened by Builder.

To activate Builder, enter the license ID and password provided in the purchase receipt in the spaces provided on the activation screen, which will be displayed when the program is first run. Please note that the licensed version of Builder and the trial version are identical. The trial version is converted to a licensed version after purchase by entering the license ID and password. There is no separate download for the licensed executable. The most recent version of Builder can always be found on the trial download page of the Adaptrade Software web site (www.Adaptrade.com).

Builder can be uninstalled through the Windows Control Panel.

## Windows and Panes

The user interface of Adaptrade Builder consists of a number of different windows. The input data windows, as shown in Fig. 9 below and in the upper left corner of Fig. 1, comprise the main view window of Builder. These tabbed windows always stay together and are always anchored to the outer window frame. The tabs of the main view window (Markets, Strategy Options, Build Goals, and Build Options) can be moved relative to each other by clicking on the tab and dragging it to a different position. For example, to move the Markets tab so that it follows the Build Options tab, click and drag the Markets tab to that location. These tabbed windows can be moved relative to each other, but will always stay in the main view window.

The other windows are referred to as *docking windows* or *panes*. The panes are *floating windows* that can be re-positioned by clicking and dragging. The panes consist of the Equity Curve, Trade List, Strategy Code, Performance (in-sample), Performance (out-of-sample), and Output windows. Panes are moved by clicking and dragging them relative to each other and to the outer window frame. To move a pane, click the title bar and start dragging the window. This action will cause a set of *docking locations* to appear. You can move the mouse cursor over the different docking locations to see a preview of where the window will appear if you release the mouse at that location. You will usually see four docking locations (top, bottom, left, right) for the entire program window (main frame) and four others relative to the current pane location.

When two or more panes are moved to the same location, they appear with tabs, as with the in-sample and out-of-sample performance windows. The tabs can be moved relative to each other by clicking and dragging, as was discussed above for the tabs of the main view window.

To rearrange the pane windows, sometimes it's necessary to move them in multiple steps. For example, to move the equity curve window so that it sits above the output window, you might need to move the output window to the bottom first, and move the equity curve and other windows to the right. Because the main view window is not a docking window, it can't be moved directly. If you want the tabs of the main view window (Markets, Strategy Options, etc.) in a different location, you can move the other panes relative to them. For example, if you want the main view window on the right, you could drag the equity curve pane, along with the other panes in that tab group, to the left side. This will force the main view window to the right. Any pane can be closed using the close button in the upper right-hand corner of the title bar. Use the View menu to restore a closed pane.

When Builder is first installed, it arranges the panes in the default configuration shown in Fig. 1. However, if you're installing the program on top of an older version, the stored locations for windows and panes in the older version may take precedence. Uninstalling Builder won't affect the layout because the layout is stored in the registry, which is not deleted during uninstall. If you're comfortable working with the registry, you can delete it yourself. That will cause the entire layout to be reset the next time you run the program. Otherwise, the windows can be rearranged manually as described above.

# Working with Project Files

The settings made in Builder can be saved for later use. Builder project files end with the extension .gpstrat; e.g., MySampleFile.gpstrat. To save your work, select Save Project from the File menu. This will prompt you to select or enter a file name. This file will not include the price data but will save the name and location of the text file containing the price data. To return to the project file later, open it by selecting it from the list of recently used files on the File menu or select it using the Open Project command of the File menu. Project files also store the strategy code for all the saved strategies, as shown in the Performance tables, along with the performance results for each strategy. Only one project file can be open at a time.

# Quick Start Steps

Adaptrade Builder is a stand-alone Windows application. The only external data it requires is a text file of price data for the market of interest. The input settings are made in a series of tabbed windows, each of which represents a different category of settings: Markets, Strategy Options, Build Goals, and Build Options. After making the desired settings, press the Build button to begin the build process. While the build process is ongoing, progress messages are displayed in the Output window. Additionally, the results windows (Performance table, Equity Curve, Trade List, and Strategy Code) are updated after each generation is complete.

**Step 1. Obtain the Price Data**

Builder is designed to read text files of price data saved in comma-delimited format, such as price data saved from the TradeStation Data Window. To display the Data Window from a TradeStation price chart, go to the View menu in TradeStation and select Data Window. To save the price data to a text file, click the disk icon on the data window, or, for TS 2000i, right-click and select "Send to File". In the Save-As dialog, select a file, then click the Save button. This will save the price data for the chart window to the text file you've selected.

**Step 2. Select the Price Data File Created in Step 1**

Open Builder and click on the Markets tab, shown below in Fig. 9. Click the Add button to add a price file or click Remove to delete one from the table. Although strategies can only be built over one file at a time, multiple files can be added to the table. The strategies will be built using the data from the file selected in the table, as indicated by the check box in the Market column.

When you select a file of price data, Builder opens the Price File Format window, as shown in Fig. 10. The top table displays the contents of the price file and allows you to identify the columns of data in the file. Click a column heading in the table to change, add, or remove a column label. If the price file was generated from the TradeStation data window, the default headings for the date, time, open, high, low, and close should be correct. Change the heading for the volume column(s) as necessary, depending on whether the price file contains one column of volume data or separate columns for up-tick and down-tick volume.

Once you've selected the format for the file of price data, clicking OK will return you to the Markets tab. You can click directly on entries in the Market Data table to change their values. Builder will try to properly infer the bar type, bar size, and session times from the data file. If any of these are incorrect, they should be manually corrected here.

Note that changing the settings for bar type, bar size or session times will not change the data used by Builder. For example, changing the session times so that the session ends earlier than the data in the file will not restrict trades to the earlier times. These settings are used by the program to properly interpret the input data. If different bar type, bar size or session times are needed, a new data file should be created with data that match the required settings.

**Step 3. Select the In-sample/Out-of-Sample Dates**

The Markets tab also includes controls for changing the start and end dates for analysis as well as the in-sample/out-of-sample dates. You can change the start and end dates for building or evaluating a strategy by clicking on the calendar controls at either end of the in-sample/out-of-sample slider. Move the slider to change the dates for in-sample and out-of-sample analysis. The in-sample data are used in building the strategy. Out-of-sample (OOS) data are used in evaluating the strategies following the build. Because the OOS data are not used during the build process, the results achieved on the OOS segment are unbiased. Generally speaking, a ratio of four or five to one for in-sample to OOS data is recommended.

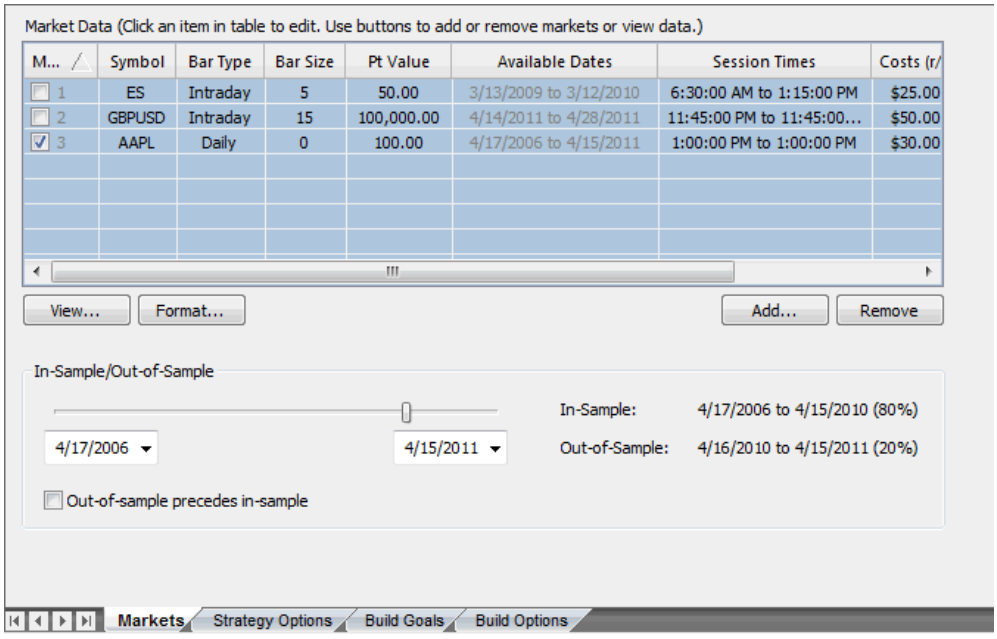



**Figure 9. Settings for the price data are made on the Markets tab.**

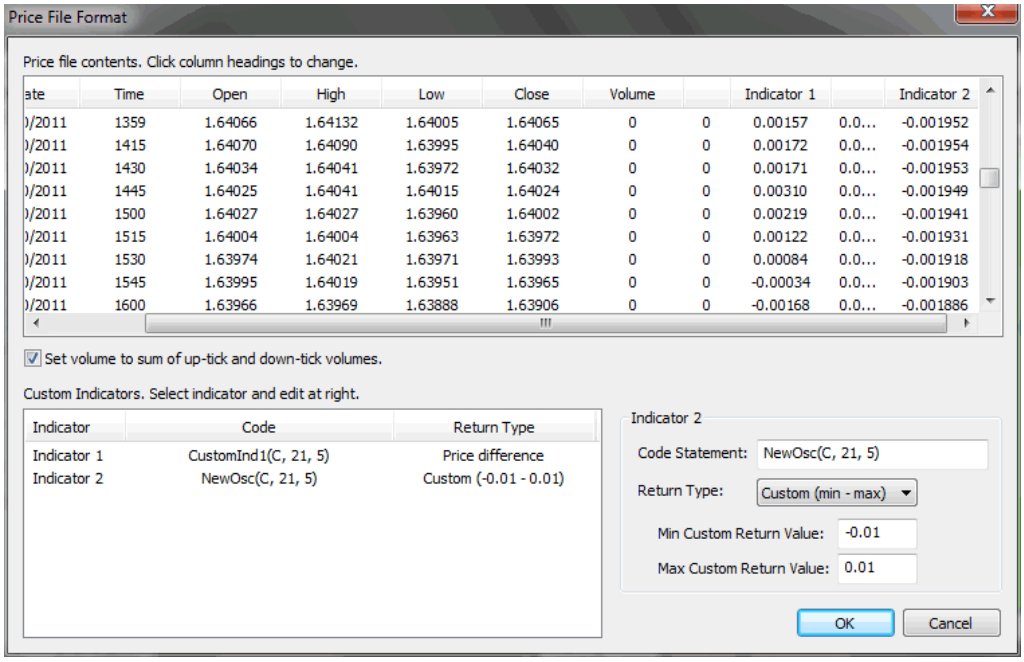



**Figure 10. The format of the price file is selected on the Price File Format window.**

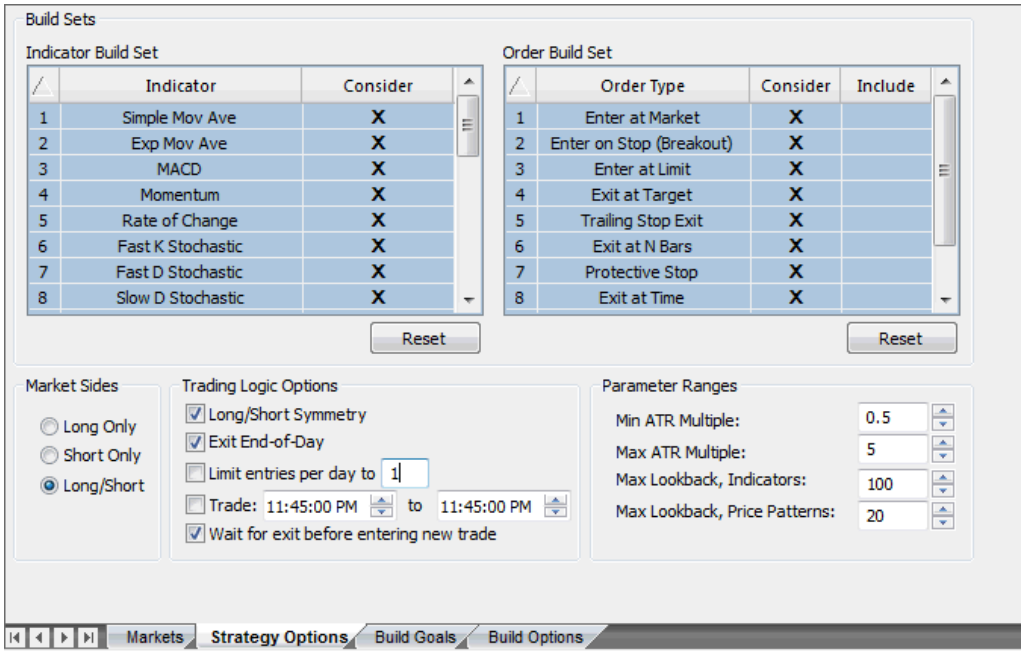



**Figure 11. Settings that control the types of strategies generated are made on the Strategy Options tab.**

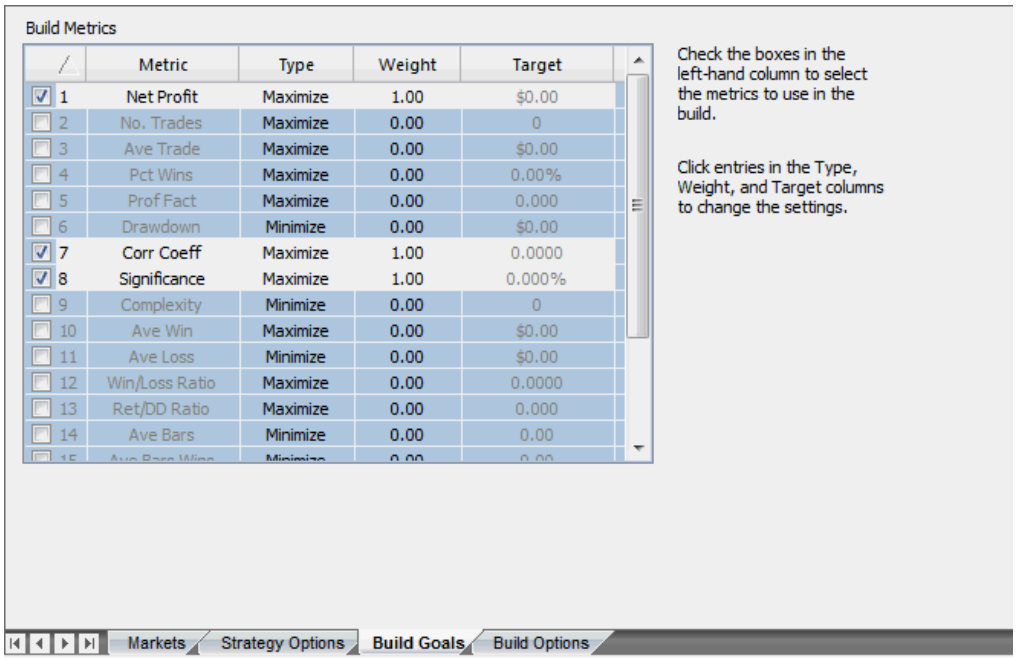



**Figure 12. Build goals are specified on the Build Goals tab by selecting weighting factors and targets for key performance metrics.**

**Step 4. Select Options on the Strategy Options Tab**

The Strategy Options tab is shown in Fig. 11. The settings on this tab control the types of strategies generated by Builder. The two tables represent the *build set*, which contains the list of possible indicators and the list of order types that the program draws from during the genetic programming process. To remove a specific indicator or type of order from the build set, click the corresponding X in the Build Set column of the table. Removing an indicator or order type from the build set means it won't be considered by the program when constructing strategies. Removing too many items may reduce the likelihood of finding viable strategies. Clicking an order type in the Include column in the Order Build Set table ensures that the order type will be included in each generated strategy.

Other options on this tab include Market Sides, which allows you to restrict entries to either long or short trades only, several Trading Logic Options, such as exiting at the end-of-day, and minimum and maximum values for constants used by the program, such as the multipliers for the average true range (ATR) and look-back lengths for indicators and price patterns.

**Step 5. Select the Build Goals**

The Build Goals tab is shown in Fig. 12. This is where you select the key performance metrics that will guide the build process. Strategies are ranked during the genetic programming process according to the *fitness*, which is a weighted average of the performance metrics selected on this tab. Click the check box in the left-hand column to select or deselect a metric. Enter a weight value in the Weight column for every selected metric. The weight values are all relative to each other; any positive number may be used. To select a specific target value, click on the entry in the Type column and select Target. Then enter a target value in the Target column. For example, to build strategies with 300 trades over the in-sample period, change the Type to "Target" and enter a value of 300 in the Target column for the No. Trades metric.

**Step 6. Choose the Build Options**

The Build Options tab is shown in Fig. 13. Different options that influence the build process are selected on this tab. For example, this is where you enter the population size and the number of generations. You can also choose to have the build process automatically check the out-of-sample results periodically after each specified number of generations and reinitialize the population if the selected metric is not met. For example, the selection shown in Fig. 13 will rebuild if the out-of-sample net profit is not positive after each five generations.

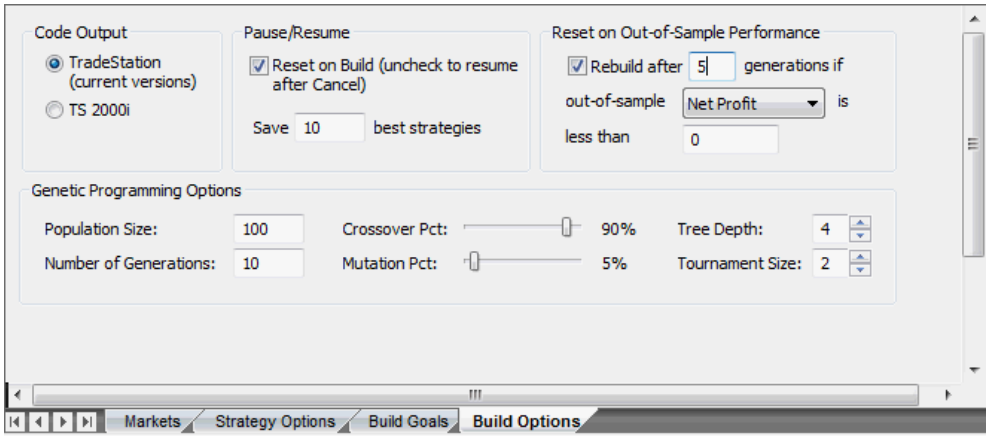



**Figure 13. Options that affect the build process are specified on the Build Options tab.**

You can also modify the crossover and mutation percentages, tree depth, and tournament size on this tab. These options are discussed further in the next chapter. To incorporate the best strategies from the prior build into the next build, uncheck the Reset on Build box before building. This will cause Builder to initialize the population with the saved strategies from the prior build. The number of saved strategies is specified using the "Save [] best strategies" option. If the Reset on Build box is checked, the population will be initialized randomly.

After making the preceding settings, it's good practice to save them by selecting Save Project from the File menu. This will prompt you to select or enter a file name for saving all the settings you've made up to this point. The resulting file will have the file extension ".gpstrat". This file will not include the price data but will save the name and location of the text file containing the price data. To return to the project file later, open it by selecting it from the list of recently used files on the File menu or select it using the Open Project command of the File menu.

**Step 7. Start the Build Process**

Click the Build button in the message bar above the main window to start the build process. A message is displayed in the Output window near the bottom of the window during each step of the process. After each generation is completed, the results windows (Performance table, Equity Curve, Trade List, and Strategy Code) are updated. The summary performance metrics for each of the top strategies are displayed in the Performance tables (in-sample and out-of-sample). Click on a row in the Performance table to display the results for the corresponding strategy in the Equity Curve, Trade List, and Strategy Code windows. The build process can be cancelled at any time by clicking the Cancel button. To restart the process after cancelling, make sure the Reset on Build option is unchecked and that the number of saved strategies is at least as large as the population size.

**Step 8. Evaluate the Generated Strategies and Adjust if Necessary**

After the specified number of generations is completed or the Cancel button is clicked, the build process stops. The generated strategies are listed in the Performance tables in order of decreasing fitness. Builder will save any number of strategies you specify on the Build Options tab up to the population size. Evaluate the saved strategies by reviewing the summary performance metrics in the Performance tables (in-sample and out-of-sample), viewing the equity curves, and examining the trade-by-trade results in the Trade List table. If none of the saved strategies meet your requirements, change the weights and/or targets in the Build Goals tab to address the deficiencies observed in the generated strategies. For example, if the drawdown is too large, you could try increasing the weight for the drawdown metric or increasing the weight for the correlation coefficient. Click the Build button to rebuild. If the Reset on Build box is checked, the current population will be discarded, and the population will be reinitialized randomly. If the box is unchecked, the saved strategies will be used to reinitialize the population.

**Step 9. Transfer the Code to TradeStation**

When you're satisfied with the results in Builder, transfer the strategy code to TradeStation or MultiCharts for any additional testing, such as real-time tracking, and for trading. The EasyLanguage code in the Strategy Code window can be copied by right-clicking and selecting "Copy Strategy". To transfer the code to your trading platform, open a new stragtegy window in TradeStation or MultiCharts, choose a name, then paste the code into the empty strategy window. Finally, compile the EasyLanguage code.

To test the strategy, insert it into the corresponding chart, such as the chart used to generate the price data file for analysis, and set the "Maximum number of bars study will reference" (in TradeStation, Format Strategies, Properties for All) to the MaxBarsBack value listed in the Performance table. This ensures that the strategy has enough data to start calculating the indicators at the beginning of the chart.

To verify that you're getting the same results in TradeStation as in Builder, make sure that the costs are set to one-half the value in Builder since Builder deducts costs on a round-turn basis, whereas TradeStation deducts costs per-side. Also make sure the start and end dates of the chart are the same as in Builder and that the MaxBarsBack value is the same. Slight differences may still occur, due primarily to rounding error differences and to initial differences in the values of some indicators near the beginning of the chart.

To get help on any feature or command while using Builder, press F1 at any time.

The remaining chapters discuss the settings and features of Builder in detail.

# Chapter 3
## Input Data and Settings

The input settings in Builder are made in a series of tabbed windows, each of which represents a different category of settings. These windows are titled Markets, Strategy Options, Build Goals, and Build Options.

## Markets

The only external data source required to use Builder is a text file of price data. Builder uses the price data in the trading simulations that are part of the strategy building process. All settings related to the price data are made via the Markets tab, shown below in Fig. 14.

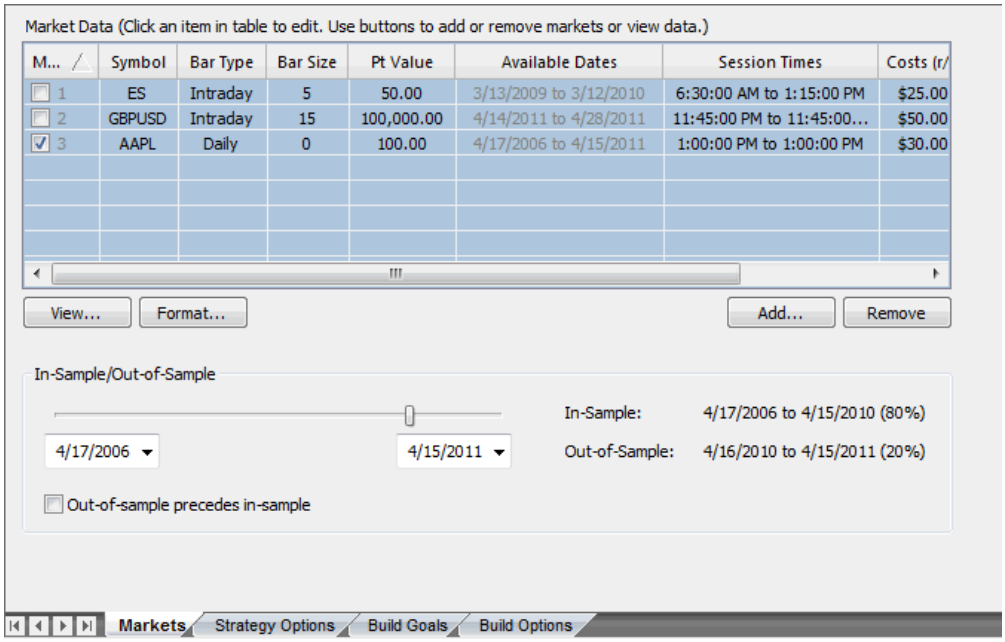



**Figure 14. The Markets tab in Builder.**

### Add Button

Click the Add button below the Market Data table to select a file of price data. The corresponding market will be added to the table. Clicking this button brings up the Price File Format window, as shown below in Fig 15.

### Price File Format Window

The Price File Format window is used to specify the format of the file of price data. Builder displays the contents of the price file in the table at the top of the window. The column headings identify the type of data contained in each column, which can be one of the following: date, time, open, high, low, close, volume, up-tick volume, down-tick volume, indicator, or "don't read/ignore". At a minimum, a price file should contain columns for the date and the closing price of each bar. To take advantage of all the features of Builder, a price file should contain the date, time, open, high, low, close, and volume (or up-tick and down-tick volumes).

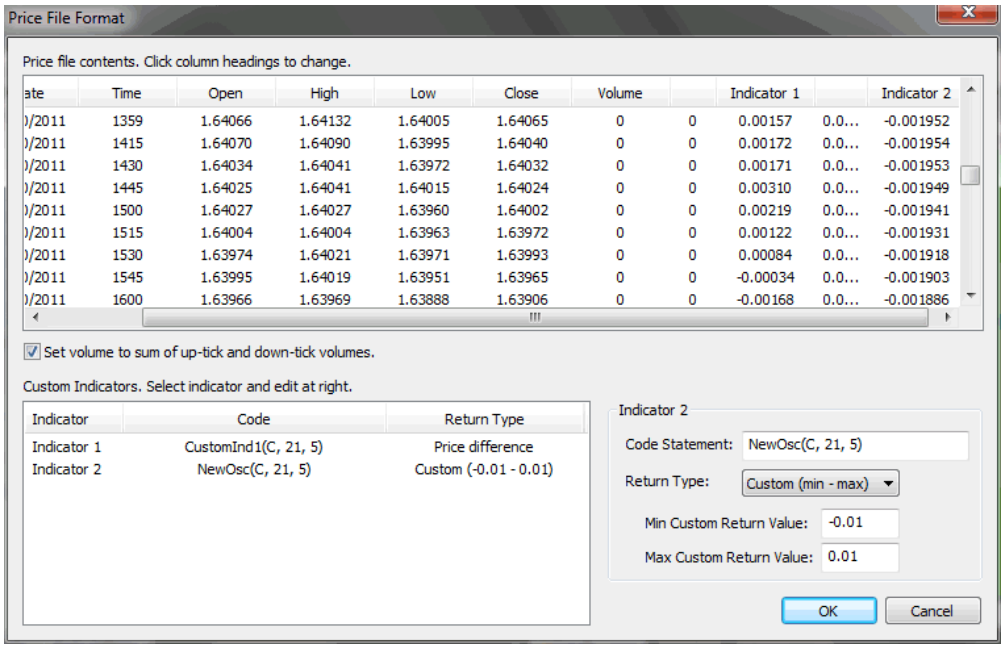



**Figure 15. The Price File Format window allows you to specify the format of the file of price data.**

Builder is designed to read text files of price data saved in comma-delimited format. An example of this data format, as obtained from the TradeStation Data Window, is shown below.

"Date","Time","Open","High","Low","Close","Up","Down"
06/15/2009,0640,504.60,505.10,500.90,503.00,3655,3577
06/15/2009,0650,503.00,503.80,496.60,497.60,5245,6854
06/15/2009,0700,497.70,499.10,496.70,499.00,2275,2029
06/15/2009,0710,498.80,498.80,497.00,497.80,1980,2202
06/15/2009,0720,497.90,498.70,496.50,497.30,2153,2274
06/15/2009,0730,497.20,497.60,496.60,497.00,1453,1411
06/15/2009,0740,496.90,497.00,493.40,493.80,3057,4165

The “Up” and “Down” fields are the up and down tick data for intraday bars. For daily bars of futures, TradeStation uses “Vol” and “OI” for volume and open interest, respectively. Builder does not use open interest.

To change the label for any column, click on the column heading and select the appropriate label from the pop-up window. Select "don't read/ignore" to have the program skip over a column of data; this corresponds to a blank column heading.

If the volume is not specified, it is set to zero. If any price field is not specified, it is set to the closing price. For example, if only the close is available, the open, high, and low are set to the close on each bar.

If the price file includes columns for both the up-tick and down-tick volume, check the box "Set volume to sum of up-tick and down-tick volumes" below the table of price data to combine the two volume fields for the total volume. Generally speaking, the two fields should be combined in order for volume-based indicators to evaluate properly. If it appears that a volume-based indicator is not evaluating properly, try deselecting this option.

The first line of the price file can contain an optional line of text labels, which is skipped over if found.

### Custom Indicators

Builder can include custom indicators in the build process. To use this feature, it's necessary to provide a column in the price file with the custom indicator value for each bar. Typically, this is done by plotting the indicator on the price chart in TradeStation or MultiCharts and saving the chart data to a text file. This adds a column to the text file with the indicator values.

To include the indicator in the build process, click on the column heading in the "Price file contents" table of the Price File Format window and select "Indicator" as the column heading. This will label the corresponding column as "Indicator 1", "Indicator 2", etc., as shown in Fig. 15. This also adds an entry to the Custom Indicators table, located below the table of price data in the Price File Format window.

Builder needs to know two things about each custom indicator you want to use: (1) the code statement needed to calculate the indicator values in EasyLanguage, and (2) the return type of the indicator. For example, in Fig. 15, indicator #1 uses the code statement "CustomInd(C, 21, 5). This is the EasyLanguage code that will be included in the strategies generated by Builder in which the custom indicator is used. This code should correspond to the indicator values in the price file. For example, evaluating the code statement CustomInd(C, 21, 5) in a strategy should produce the values shown in the price file under the column labeled Indicator 1.

**Note**: If a custom indicator cannot be represented in EasyLanguage using a single function call, there are two options for using the custom indicator in Builder: (1) re-write the indicator so that it's values are generated by a single function call, or (2) edit the strategy code generated by Builder after the build process is complete.

The return type for Indicator 1 in Fig. 15 is "Price difference". This means the values produced by the indicator are equivalent to the difference between two prices, such as the difference between two moving averages. The table below lists the different allowable return types with examples of common indicators that return that type.

| **Return Type** | **Definition/Examples** |
|---|---|
| Price | Any market price; C, Average(C, N), Highest(L, N), Keltner |
| Price difference | Difference between two prices; MACD, momentum, H - L, ATR |
| Price rate of change | Ratio of two prices (price 1/price 2); ROC |
| Oscillator, 0 - 100 | Indicators scaled to return between 0 and 100; FastK, SlowD, RSI |
| Oscillator, -50 - +50 | Indicators scaled to return between -50 and +50 |
| ADX | ADX, DMI |
| Volume | volume, Average(volume, N) |
| Day of week | 0 for Sunday, 1 for Monday, etc. |
| Time of day | HHMM format; e.g., 1425 for 2:25 pm |
| True/false | 0 for false, 1 for true |
| Custom | User-specified min to max range; e.g., -0.1 to +0.4 |

The return type is used by Builder to determine which indicators can be meaningfully compared to other indicators. It's necessary to specify the correct return type in order for the program to properly incorporate the custom indicator into the logical conditions for entry and exit.

For each custom indicator in the table of custom indicators, select the indicator in the table and enter the code statement in the edit field to the right of the table and select the return type; see Fig. 15. For any indicator with a custom return type, enter the minimum and maximum values that the indicator can return. *It's best to avoid selecting a custom return type unless necessary because this will limit the ways in which the custom indicator can be used in the logical conditions.*

**Note:** Once a custom indicator has been included in a built strategy, changing the market and re-evaluating the strategy may cause errors if the data columns for the same custom indicators are not available in the new price file.

#### Remove Button

Clicking the Remove button will remove the selected market from the Markets Data table.

#### View Button

Click the View button to confirm that the file of price data has been read correctly. This will display a window listing the data that have been read.

#### Format Button

Clicking the Format button brings up the Price File Format window, as discussed above. This can be used to change how the program reads the price file and/or to change the specifications of any custom indicators being used in the program.

#### Obtaining Price Data

Builder can utilize price data obtained from any source provided the file is in comma-delimited format and can be formatted under the Price File Format window. For TradeStation users, the easiest way to obtain the price data file for use in Builder to is save the data from a TradeStation chart window. This can be done through the Data Window. To display the Data Window, go to the View menu in TradeStation and select Data Window or, in TS 8 or above, right-click on a chart and select View Data Window. To save the price data to a text file, click the disk icon on the data window, or, for TS 2000i, right-click and select "Send to File". In the Save-As dialog, select a file, then click the Save button. This will save the price data for the chart window to the text file you've selected.

#### Market Settings

Multiple price files can be selected in the Markets tab. While Builder currently builds strategies using only one price file at a time, it's sometimes desirable to open multiple files under Markets, as shown in Fig. 14, where three price files have been opened. The strategy will be built using the price file selected in the table, as indicated by the check box in the Market column; e.g., market #3 in Fig. 14. Once the strategy is built, it can be evaluated on another market by clicking the corresponding row in the table and selecting Evaluate from the Build menu.

Click directly on entries in the table to change their values. A text label can be entered in the **Symbol** column to identify the market. Values should be entered in the **Pt Value** and **Costs** columns, respectively, for the point value and round-turn trading costs. For example, the E-mini S&P 500 futures have a point value of $50. This is the value that is multiplied by price to determine position value. For stocks, you can set this value to the number of shares per trade. For example, set the point value to 100 to simulate trading 100 shares per trade. Forex symbols generally require a point value equal to the value of the position, such as 100,000 for one standard lot.

The cost entry represents the total trading costs (commissions, fees, and slippage) deducted from each trade on a round-turn basis (i.e., deducted once per trade). For stock trades, where the point value has been set to the number of shares per trade, this value should be "per trade", rather than "per share". For example, if the point value is set to 100, representing 100 shares per trade, entering a cost of $25 would cause a deduction of $25 per 100 share trade. *Note that TradeStation typically deducts trading costs on a per-side basis, so you should use one-half the trading cost amount in TradeStation that you enter in Builder.*

Trading costs should always be included though they need not be precise. The purpose is to insure the strategies generated during the build process overcome trading costs. Entering a slightly higher cost value than anticipated in reality may guide the process towards more robust strategies. However, setting trading costs too high may prevent the program from finding viable strategies that have a low average trade profit/loss.

Clicking on the **Bar Type** entry will open a pull-down menu with the following choices: Intraday, Daily, Weekly, Monthly, Tick, Range, Unknown. Intraday bars are specified in minutes, such as 5 min bars or 60 min bars; the smallest allowable value is 1 minute. The **Bar Size** column should contain the number of minutes for intraday data. For all other types of data, entries in the Bar Size column won't be used by the program and can be left blank. Daily, weekly, and monthly Bar Type selections represents bars that span a day, week, and month, respectively. Tick bars are built up from a specified number of transactions or ticks and are typically used in futures trading. A single tick represents a transaction that took place at a certain price and time. It could be for any number of contracts. Tick bars are specified in terms of the number of ticks per bar, such as 500 tick bars. Range bars represent trading within a specified point range, such as five points. All bars in a range bar chart have the same range.

Builder will try to properly infer the bar type, bar size, and **session times** from the data file. If any of these are incorrect, they should be manually corrected in the table. *Note that changing the settings for bar type, bar size or session times will not change the data used by Builder.* For example, changing the session times so that the session ends earlier than the data in the file should not be done and will not restrict trades to the earlier times. These settings are used by the program to properly interpret the input data. If different bar type, bar size or session times are needed, a new data file should be created with data that match the required settings. For example, a *custom session* can be created in TradeStation to restrict the data to a specified time range.

The **Available Dates** column in the Markets table displays the date range for data read from the price file. This column is for informational purposes only; it cannot be edited. To change a file identified in the table, click on the entry in the **File Name** column. This will open the same file dialog as clicking on the **Add** button. To delete a price file from the table, select the file to delete, and click the **Remove** button. The entries in the table can be sorted by clicking on a column heading. For example, to sort by symbol, click on the Symbol heading. Any of the tables described in this chapter can be sorted in the same manner.

The Markets tab also includes controls for changing the **start and end dates** for analysis as well as the **in-sample/out-of-sample dates**. To change the start and end dates for building or evaluating a strategy, click on the calendar controls at either end of the in-sample/out-of-sample slider or type the desired date directly into the date box.

Move the slider to change the dates for in-sample and out-of-sample analysis. The in-sample data are used in building the strategy. Out-of-sample (OOS) data are used in evaluating the strategies following the build. Because the OOS data are not used during the build process, the results achieved on the OOS segment are unbiased. Generally speaking, a ratio of

between three and five to one for in-sample to OOS data is recommended. When selecting Evaluate or Evaluate All from the Build menu, both in-sample and OOS data are used.

The check box below the slider control (**Out-of-sample precedes in-sample**) allows you to specify whether the OOS period follows the in-sample period or precedes it. Some traders prefer to perform OOS testing on data following the in-sample period, whereas others prefer to perform the OOS testing on data prior to the in-sample period. Leave the box unchecked to perform OOS testing on data following the in-sample period or check the box to perform OOS testing on data prior to the in-sample period.

There can only be one OOS period at a time. Builder stores the dates for the in-sample period over which the strategies were built. Once the strategies in a given project file (.gpstrat) have been built, these build dates cannot be changed unless you rebuild the strategies. If you change the start and end dates after building and select Evaluate from the Build menu, Builder will use the stored build dates to determine the dates for the in-sample and OOS periods. If by changing the start and end dates, the date range is extended both before and after the build (in-sample) period, there may be two OOS periods, one prior to the build dates and one following them. Because the program can only have one OOS period, it will use the larger of the two periods when reporting and plotting the OOS results.

## Strategy Options

The Strategy Options tab is shown in Fig. 11 in the previous chapter. The settings on this tab control the types of strategies generated by Builder. The two tables represent the *build set*, which contains the list of possible indicators and the list of order types that the program draws from during the genetic programming process. To remove a specific indicator or type of order from the build set, click the corresponding X in the Build Set column of the table. To restore the previous settings, click the Reset button. Removing an indicator or order type from the build set means it won't be considered by the program when constructing strategies. Removing too many items may reduce the likelihood of finding viable strategies. The list of indicators is shown in Table 1 in Chapter 1. The indicators themselves are described in the appendix. The different types of entry and exit orders were discussed in Chapter 1.

The Order Build Set includes a column labeled "Include". Clicking an order type in the Include column ensures that the order type will be included in each generated strategy. For example, to make sure each strategy includes a protective (money management) stop, click the Include column entry for the Protective Stop order type.

#### Market Sides

This option allows you to restrict entries to either long-only, short-only, or both long and short trades.

#### Trading Logic Options

The **Long/Short Symmetry** option, if checked, constructs the short-side entry condition by logically reversing the rule for the long side. This will reduce the system complexity, which may increase the reliability of the strategy. However, some markets have a clear directional bias, such as stock indexes, which tend to have an upward bias. In such cases, it may be better to use different rules for long and short trades.

The **Exit End-of-Day** option was discussed in the previous chapter under exit orders. In order to apply this exit to your strategies, the check box for this option must be checked and the corresponding exit order must be included in the build set.

For intraday strategies, you can specify the maximum number of trade entries per day by checking the box **Limit entries per day to []** and entering the desired maximum number of daily entries in the box. Limiting the number of entries per day can often improve results for intraday strategies because it reduces the number of different market conditions that must be accommodated.

Intraday trades can be restricted to certain times of day using the **Trade only from** option. Enter the starting and ending times using the time selectors. This will restrict both entries and exits to the time range chosen. If a trade is open at the end of the time range, it will be closed at market on the next bar.

The option to **Wait for exit before entering new trade** is checked by default. This adds an entry condition that only allows an entry if the current position is flat. If this option is unchecked, a long entry may reverse a short position and vice-versa.

#### Parameter Ranges

This section allows you to optionally change the minimum and maximum values for constants used by the program, such as the multipliers for the average true range (ATR) and look-back lengths for indicators and price patterns. For example, if you find that the strategies generated by the program contain stops that are too large, you can reduce the value for the **Max ATR Multiple**.

The lookback length for indicators is chosen by the program from values that range from 1 to the value in **Max Lookback, Indicators**. This applies to all indicators requiring a lookback length with the exception of price patterns.

The lookback length for price patterns, which is chosen from values that range from 1 to the value in **Max Lookback, Price Patterns**, applies to individual prices; i.e., it's the "N" in O[N], H[N], L[N], and C[N].

## Build Goals

The Build Goals tab is shown in Fig. 12 in the previous chapter. This is where you select the key performance metrics that will guide the build process. Strategies are ranked during the genetic programming process according to the *fitness*, which is a weighted average of the performance metrics selected on this tab. To add or remove a metric from the fitness function, click the check box in the left-hand column of the Build Metrics table.

Metrics can be minimized, maximized, or a target value can be set, as determined by the selection in the Type column. To change the type, click the entry in the Type column and select Minimize, Maximize, or Target from the drop-down menu. If the type is either Minimize or Maximize, only a weight value is required. Each metric you select must have a nonzero weight value, which is entered in the Weight column. The weight values are relative to each other; any positive number may be used. Additionally, each performance metric is normalized to the range 0 to 1 so that it will contribute equally to the fitness calculation for equal weight values. For example, if all weight values are equal to 1, each performance metric will contribute equally to the calculation of the fitness.

If the type is set to Target, both a weight and a target value are required. To set a specific target value, first select "Target" as the type, and enter a target value in the Target column. For example, to build strategies with 300 trades over the in-sample period, change the Type to "Target" and enter a value of 300 in the Target column for the No. Trades metric. Finally, enter a weight value in the Weight column.

The build metrics are calculated over the in-sample period, which is the data segment used in building the strategies. The metrics in the Build Metrics table are the same ones listed in the Performance tables. By monitoring the results in the performance tables, as well as the equity curve, you can get a good idea of which metrics may need to be added or changed to improve the results. It's usually best to start with a small number of metrics, such as three or four. The default selections are often a good starting point. After a few generations, you can add or remove metrics, change the weights, and add or remove target values based on the reported results. An incremental approach usually works better than setting the build metrics arbitrarily and letting the build process run for an extended period of time without intervention.

### Performance Metrics

The different performance metrics calculated by Builder are described below. These metrics are used in the Build Metrics table and are reported in the Performance tables (see next chapter). When used to calculate the fitness function, the metrics are calculated over the in-sample data segment.

**Net Profit.** The net profit is the total profit (gross profit minus gross loss) for all trades, both wins and losses in the trading period.

**No. Trades.** Total number of trades in the trading period.

**Ave Trade.** Average trade. Sum of all trades, wins and losses, divided by the total number of trades. When choosing the build metrics, keep in mind that the average trade is related to the number of trades. Setting a weight value for this metric will tend to result in strategies with a higher average trade and fewer trades.

**Pct Wins.** Percentage wins. Percentage of trades with a profit greater than zero.

**Prof Fact.** Profit factor. Gross profit divided by the absolute value of gross loss. Profit factors of 1.5 or more suggest a strong system.

**Drawdown.** Largest decline in equity from the highest prior peak on an intraday basis.

**Corr Coeff.** Correlation coefficient of the equity curve. The correlation coefficient varies from -1 to +1, with values close to +1 indicating a straight, upward sloping curve. Values above 0.9 indicate a fairly straight equity curve and therefore less variation in returns over different periods.

**Significance.** Statistical significance of the average trade. This represents the probability in percent that the average trade is greater than zero, taking into account the statistical properties of the distribution of trades and the number of degrees-of-freedom of the trading strategy. The number of degrees-of-freedom is the number of trades minus the number of inputs to the strategy. The more degrees-of-freedom, the better. A significance level of 95% or greater is generally desirable.

**Complexity.** In Builder, strategy complexity is defined as the number of strategy inputs. Builder includes an input for each adjustable parameter used in an entry rule, entry order, or exit order. The total number of inputs is a measure of the complexity of the system. Lower complexity implies a greater number of degrees-of-freedom and a lower likelihood of over-fitting.

In many cases, after some number of generations, the top results will tend to converge so that several strategies with different number of inputs will produce the same performance results.

Setting the weight for complexity to a small value can be a way to "break ties" among these otherwise similar strategies based on the number of inputs. For example, you may find that the top 10 strategies have the same performance results but several of them have extra inputs. If the complexity is part of the fitness function, the strategies with the fewest inputs will be ranked higher.

**Ave Win.** Average of the winning trades over the trading period.

**Ave Loss.** Average of the losing trades over the trading period.

**Win/Loss Ratio.** Ratio of the average win to the absolute value of the average loss.

**Ret/DD Ratio.** Ratio of the net profit to the drawdown.

**Ave Bars.** Average number of bars in all trades, both wins and losses. Consider setting a target and/or weight value for this metric if you want to limit the length of trades in the generated strategies.

**Ave Bars Wins.** Average number of bars in winning trades.

**Ave Bars Loss.** Average number of bars in losing trades.

**Max Win.** Largest winning trade over the trading period.

**Max Loss.** Largest losing trade over the trading period.

**Ave MAE.** Average maximum adverse excursion. The maximum adverse excursion (MAE) is the worst-case intra-trade drawdown; i.e., the largest loss on an open trade. The Ave MAE is the average of the MAE's over all trades. Consider setting a target and/or weight value for this metric if you want to limit intra-trade drawdown in the generated strategies.

**Max MAE.** Maximum value of the maximum adverse excursion. The Max MAE is the largest of the MAE's over all trades. Consider setting a target and/or weight value for this metric if you want to limit intra-trade drawdown in the generated strategies.

# Build Options

The Build Options tab is shown in Fig. 13 in the previous chapter. The options that influence the build process are selected on this tab.

### Code Output

The type of strategy code that is generated can be selected in this section. Select **TradeStation** for TradeStation version 6 and above and for recent versions of MultiCharts or **TS 2000i** for TradeStation version 2000i. The code for all of the top strategies listed in the results tables is stored in the project file (extension .gpstrat). If you want to generate code for a different version of TradeStation, change the selection here, select the desired strategy from the performance table, and select Evaluate from the Build menu.

### Pause/Resume

The Pause/Resume option determines whether the results from the prior build are used to initialize the population for the next build. To incorporate the best strategies from the prior build into the next build, uncheck the Reset on Build box before building. This will cause Builder to initialize the population with the saved strategies from the prior build. If the

number of saved strategies is at least as large as the population size, this is the same as resuming the build after pausing it; hence the Pause/Resume label.

It's not necessary to save as many strategies as the population size. If you have a population size of, say, 500, you might want to save the top 100 strategies. Stopping the build (or waiting for it to end) and clicking the build button again with the Reset on Build box unchecked will then initialize the first 100 members of the population with the saved strategies from the prior build. The other 400 members will be initialized randomly. In this case, it's not a true "pause and resume." Instead, it includes the best results from the prior build into the next one while still allowing for new members to influence the results.

If the Reset on Build box is checked, the current population will be discarded, and the population will be initialized randomly. This is the default setting and is the same as starting a new build, in which there are no existing population members.

The number of saved strategies is specified using the **Save [] best strategies** option. This is the number of members from the population that are stored. Builder selects the strategies to store based on fitness. For example, if the population size is 200 and you enter a value of 100 for this setting, the top 100 strategies ranked by fitness will be stored. You can only store as many members as are in the population. If you enter a number larger than the population size, the entire population will be stored. The stored strategies are from the last completed generation or from the initial population if the build was cancelled before the first generation was completed.

The results for the stored strategies are displayed in the Performance tables. Clicking on any row in the performance tables displays the equity curve, trade results, and strategy code for the corresponding strategy.

#### Reset on Out-of-Sample Performance

You can also choose to have the build process automatically check the out-of-sample results periodically after each specified number of generations and reinitialize the population if the selected metric is not met. For example, the selection shown in Fig. 13 will rebuild if the out-of-sample net profit is not positive after each five generations. The selected metric that is evaluated on the OOS data can be selected from the drop-down menu. You can also enter the value that the metric is compared to. For example, you could rebuild if the OOS profit factor is less than 2.0 by selecting "Prof Fact" from the pull-down menu and entering 2.0 into the space. The available performance metrics are the same as those described in the previous section.

#### Genetic Programming Options

The **population size** is the number of strategies in the genetic programming population. Strategies are generated by evolving the population. The larger the population, the longer the process will take, but the more diversity will be introduced into the solution. More diversity generally increases the probability of finding a good strategy. With large intraday price files, a population as small as 100 members may be necessary to achieve reasonable solution times, depending on the bar size, time span, and computer hardware specifications. Files of daily data generally allow for population sizes of 1000 or more.

The **number of generations** is the number of steps in the evolution of the population of trading strategies. During one generation, the GP process generates a number of new population members (strategies) equal to the size of the population. A higher number of generations generally results in better strategies. However, after a number of generations, the population members may start to converge so that little additional benefit is achieved by

continuing the build process. It's usually better to start with a smaller number of generations, such as 5 - 10, and continue building if the results are promising and have not yet converged.

To explore the variety of strategy logic that can be generated by Builder, try setting the number of generations to zero. This will stop the build process after the initial population is generated. Because the initial population is generated randomly and has not been modified by crossover and mutation, the resulting strategies will display a wide variety of trading logic.

The crossover percentage (**Crossover Pct**) is the percentage of members of the population generated from crossover. Crossover is an operation that takes part of one member and combines it with part of a different member to create a new member of the population. Most members should be generated via crossover. The default setting is 90%. Members not generated via crossover are generated via mutation. To introduce more randomness into the population, try decreasing this percentage, which will increase the percentage of members generated from mutation.

The mutation percentage (**Mutation Pct**) is the probability that a given element of the population member being created by mutation will be modified. Mutation operates on an element-by-element basis. During mutation, each element, such as part of an entry rule or part of the exit logic, is randomly modified with a probability equal to this value. To introduce more randomness into the population, try increasing this percentage.

The entry rules in Builder are represented by tree structures, as explained in Chapter 1; see Fig. 3. The **tree depth** is the number of levels in the tree. For example, the tree in Fig. 3 has a depth of five. The default value of tree depth in Builder is 4. To increase the potential complexity of the entry rules, a higher tree depth can be used. The tree depth will tend to increase over successive generations as a result of the crossover operations, which may result in a tree depth greater than the value specified here. To prevent the entry rules from becoming overly complex, the complexity metric can be included as part of the build goals, as explained in the Performance Metrics section, above.

Setting the tree depth to zero will result in trivial entry conditions that are set to "true". In this case, the entry conditions will have no effect on the trading strategy logic. This might be desirable if you want to generate strategies that have the minimum number of inputs. The entry logic for strategies of this type will be limited to the entry order logic. For example, a strategy might be generated with an entry stop order that contains a stop price calculated from a multiple of the average true range. This would be the only entry logic for that strategy.

As explained in Chapter 1, the parent members for crossover and mutation are chosen by randomly selecting members of the population and choosing the member with the highest fitness score. This is known as tournament selection. The **tournament size** is the number of population members randomly selected for the tournament. The default value is 2, which means that two members of the population are randomly selected, and the one with the highest fitness score is selected as the parent. A so-called *negative tournament* is used to select the member of the population to replace by the newly generated population member. In a negative tournament, the member of the tournament with the lowest fitness score is replaced. The tournament size selected here governs both types of tournaments.

# Chapter 4
## Build Results

## Overview
The results generated from running Builder include the strategy code for each of the top strategies along with the performance results for each one. The docking windows (panes) described below display the different types of program output. Chapter 2 discussed working with windows and panes in Builder.

## Output Window
The Output window displays messages generated by the program while performing build and evaluate operations. For example, when building strategies, a message is displayed in this window several times per second indicating which member of the population has been recently initialized or created. This window will also display an error message if the price file cannot be found.

**Memory Allocation Errors.** The following message will be displayed in the Output window if the program runs out of memory:

```
>> Insufficient memory. Try reducing population size, tree depth, or price
file size.
```

This error occurs when the program's memory requirements exceed the computer's available memory. When this happens, the program is designed to detect the lack of available memory and stop processing before a program crash occurs. When this happens, it can be resolved by reducing either the population size or the size (length) of the file of price data. If the tree depth has been increased beyond the default value, lowering it may also help.

## Performance (in-sample, out-of-sample) Tables
The performance metrics for each of the saved strategies are displayed in the Performance tables (in-sample and out-of-sample). The performance metrics displayed in the tables are defined in the previous chapter. Click on a row in the Performance table to display the results for the corresponding strategy in the Equity Curve, Trade List, and Strategy Code windows. The strategies in the Performance tables are listed in order of decreasing fitness. Builder saves and displays the number of strategies you specify on the Build Options tab up to the population size.

Note that the performance metrics are calculated only for closed trades. If the data segment (in-sample or out-of-sample) ends with an open trade, that trade will not be included in the results for that segment. If the in-sample period ends with an open trade that exits in the out-of-sample period (or vice-versa if the in-sample period follows the out-of-sample period), then the trade will be included in the metrics for the out-of-sample period.

While the build process is ongoing, the program updates the results in the performance tables after each completed generation. If the build is cancelled before the first generation is completed, the initial population will be displayed. The table results can be sorted by any

column by clicking on the column heading. For example, to sort the out-of-sample results by net profit, click on the Net Profit column label in the Performance (out-of-sample) table.

The results from either of the performance tables can be copied to a spreadsheet by right-clicking on the table and selecting “Copy”. Open a blank spreadsheet page and select Paste to insert the table results into the spreadsheet.

## Strategy Code Window

The code for the strategy selected in the performance table is displayed in the Strategy Code pane. To view the code for a different strategy of the population, select the strategy in the performance table. The code in the Strategy Code window can be copied to the clipboard by right-clicking and selecting **Copy Strategy** or by selecting **Copy Strategy** from the **Edit** menu. To copy only part of the strategy code, click and drag the mouse over the part you want to copy, then right-click and select **Copy** to copy the selected text to the clipboard. The strategy code cannot be edited directly in Builder. To edit the code, copy the code to your trading platform (e.g., TradeStation or MultiCharts), and use the editor in the trading platform.

To transfer the code to your trading platform, open a new stragtegy window in TradeStation or MultiCharts, choose a name, then paste the code into the empty strategy window. Finally, compile the EasyLanguage code. To test the strategy, insert it into the corresponding chart, such as the chart used to generate the price data file for analysis, and set the ”Maximum number of bars study will reference” (in TradeStation, Format Strategies, Properties for All) to the **MaxBarsBack** value listed in the Performance table. This ensures that the strategy has enough data to start calculating the indicators at the beginning of the chart. In order for the results in your trading platform to match those in Builder, it’s necessary to use the correct MaxBarsBack value.

## Equity Curve Window

The closed-trade equity curve for the strategy selected in the performance table is displayed in the Equity Curve pane. To view the equity curve for a different strategy of the population, select the strategy in the performance table. The equity curve is based on the most recent evaluation. If you want to see the curve for a different range of dates, change the date range on the Markets tab, then select **Evaluate** from the Build menu. Similarly, the selected strategy can be evaluated over a different market by selecting the market in the Market Data table and re-evaluating the strategy.

The out-of-sample (OOS) results are shown in the equity curve plot in a shaded section. The OOS section is shaded green if the OOS results are profitable or red if they’re not.

## Trade List Table

The Trade List table displays the trade-by-trade results for the strategy selected in the performance table. To view the trade list for a different strategy of the population, select the strategy in the performance table. The trade list table includes the following fields: entry date, entry price, exit date, exit price, stop price, direction, profit/loss, risk and equity. The stop price field will read “NA” if there was no money management (protective) stop for that trade. The direction field is either “Short” or “Long”. The profit/loss field is the closed trade profit or loss after costs. The risk is the potential loss corresponding to the protective stop, if any. The equity field displays the cumulative closed trade equity, as shown in the equity curve plot. Note that a trade that enters during the in-sample period but exits in the OOS period will be listed in the OOS section of the list (or vice-versa if the in-sample and OOS periods are reversed).

In-sample results in the trade list table are shown against a white background, with profitable trades written in green text and unprofitable trades in red text. The out-of-sample trades are written in black text with a colored background. The background is green if the OOS results are net profitable or red if the OOS net profit is negative.

The trade list table can be copied to a spreadsheet by right-clicking on the table and selecting "Copy". Open a blank spreadsheet page and select Paste to insert the table results into the spreadsheet. To save the table contents to a comma-delimited text file (.csv file), right-click and select "Save to file". This file is formatted so that it can be read by **Market System Analyzer (MSA)** position sizing software as a trades file. In MSA, select the WriteTrades format when selecting the file under Data Source or when importing the file.

# Chapter 5
## Usage Topics

## Overview
The basic steps to using Adaptrade Builder were outlined in the Quick Start Steps section in Chapter 2 (Getting Started). This chapter delves into various topics related to using Builder, including a discussion of out-of-sample performance, factors that affect the build time and how to reduce it, considerations for after you've built a strategy, and various tips and hints for strategy building.

## Out-of-Sample Performance
Genetic Programming (GP) is type of optimization. Most systematic traders are probably familiar with parameter optimization, in which the inputs to a strategy are optimized. Unlike parameter optimization, GP optimizes the strategy's trading logic. Nonetheless, the risk of over-optimization, or "over-fitting", is also a concern for GP, just as it is for parameter optimization.

Typically, optimization is performed over one segment of data, called the optimization or in-sample segment, and tested on different data, called the test or out-of-sample segment. Over-fitting refers to the problem of optimizing a strategy so that fits the in-sample segment well but doesn't work well on any other data, including the out-of-sample data.

Poor out-of-sample performance is usually caused by one of several factors. One important factor is the so-called *number of degrees-of-freedom* in the in-sample segment. The number of degrees-of-freedom, which is equal to the number of trades minus the number of rules and conditions of the strategy, determines how tightly the strategy fits the data. Provided inputs are added for each parameter in the strategy, the number of strategy inputs can be used as a proxy for the number of rules and conditions. For example, if a strategy has 100 trades and 10 inputs, it has 90 degrees-of-freedom. The more degrees-of-freedom, the less likely it is that the strategy will be over-fit to the market and the more likely it is that it will have good out-of-sample performance.

The number of degrees-of-freedom can be increased during the build process by adjusting the weights for the number of trades and/or the strategy complexity. All other things being equal, increasing the performance weighting for the number of trades will result in strategies with more trades and therefore more degrees-of-freedom. Likewise, increasing the performance weighting for the complexity metric will result in strategies with fewer inputs, which will also increase the number of degrees-of-freedom.

Builder also incorporates the degrees-of-freedom into the build process via the significance performance metric. In Builder, "significance" is based on the Student's t test applied to the average trade. It measures the statistical significance of the average trade; that is, the probability that the average trade will be greater than zero. The t test is based on the number of degrees-of-freedom but is a more complete measure of whether a strategy is over-fit than the number of degrees-of-freedom alone. One way, then, to improve out-of-sample performance is to use the significance metric to generate strategies that have a high statistical significance.

Another important factor affecting out-of-sample performance is the variety of market conditions in the in-sample segment. Generally speaking, it's better to optimize over data that includes a wide variety of market conditions, such as up trending and down trending markets, periods of consolidation, high and low volatility, etc. The more variety in the in-sample segment, the more likely it is that the strategy will perform well on other data, including out-of-sample data and in real-time trading. While the future never exactly duplicates the past, provided the future (or out-of-sample data) is similar enough to at least part of the in-sample segment, the strategy should perform well on new data.

The value of optimizing over a variety of market conditions presumes that good performance is achieved over each part of the in-sample segment. One way to measure this is with the correlation coefficient of the equity curve, which measures how closely the equity curve approximates a straight line. If the equity curve is a straight line, it implies that the performance is uniform over all segments of the data. Obviously, this is desirable if the goal is to achieve good performance over as many different types of market conditions as possible. The correlation coefficient for the strategies generated by Builder can be increased by increasing the performance weighting for this metric.

Unfortunately, there will be cases where even with a high significance, a correlation coefficient close to 1, and a wide variety of market conditions in the in-sample segment, the out-of-sample performance will be poor. This can happen for several reasons. First, even a simple strategy with few parameters can in some cases fit the *noise* rather than the *signal*. By definition, noise is any part of the market data that does not contribute to profitable trading signals. Secondly, the market dynamics on which the strategy logic is based (i.e., the signal) may have changed in the out-of-sample segment enough to negatively impact performance. This is sometimes due to a fundamental change in the market, such as the switch from floor-based to electronic trading. However, more subtle changes, often related to the trading patterns of market participants, are also possible, particularly for shorter-term trading.

If this appears to be the problem, the solution may be as simple as rebuilding the strategy with new trading logic. Another possible solution is to include the most recent data in the optimization segment and test it out-of-sample by tracking the performance in real-time. In most cases, a strategy that has a large number of trades, a high significance value and good performance on the in-sample segment will continue to perform well for some period of time post-optimization. The more important question is how long it will continue to perform before requiring re-optimization or rebuilding. This issue is discussed further in the section "Post-Build Testing and Optimization".

## Build Time

The GP algorithm is very computationally intensive. There are several factors that can affect the time it takes to build a strategy:

- Length of the price data file. A file of 10 years of daily data for the EURUSD currency is about 138k in size. One year of 5 min data for the E-mini S&P is ten times as large. The build time is proportional to the file size, so a file that is 10 times as large will generally take about 10 times as long to process.
- Population size and number of generations. These parameters determine how many strategy simulations are performed. One simulation is performed for each member of the population for each generation.
- Tree depth. This setting determines in part the complexity of the entry conditions for the strategies. Strategies that are more complex take longer to evaluate.

- Long/short symmetry. If the short side is the mirror image of the long side, Builder can save time both in building the strategy and evaluating it. Strategies in which the long and short sides are different are more complex and take longer to build.
- Hardware and software considerations. Obviously the build process will run faster on a faster processor. The build algorithm is a parallel processing algorithm that takes advantage of multi-core processors. The more cores the better. It's also the case that if multiple programs are running together on the same computer at the same time as Builder, fewer processor cycles will be available for Builder, and the build process will take longer.

Depending on these factors, the build process may take anywhere from several minutes to several hours or longer. If the build process seems to be taking too long, it can be cancelled and restarted at a later time. To restart the build process later, make sure to save the current results before exiting the program by selecting Save Project from the File menu. Upon opening the file, uncheck the Reset on Build box (Build Options tab). This will instruct the program to initialize the population using the saved strategies from the prior session. Builder saves the top number of strategies specified by the Save Best Strategies setting on the Build Options tab. To restart the build process exactly where you left off, the Save Best input should be set to the population size prior to the initial build. That way, Builder will save the entire population for the last completed generation. Note that at least one generation must be completed for the results to be saved.

## Post-Build Testing and Optimization

As discussed above, genetic programming (GP) is an optimization process. As such, it's important to perform out-of-sample testing on any strategy generated by the program that you intend to trade. Builder was designed to perform these calculations automatically. All you need to do is decide how much data to use for in-sample and how much for out-of-sample testing. As mentioned elsewhere, a good ratio of in-sample to out-of-sample data is often between three and five to one. For example, setting the slider control on the Markets tab to 80% in-sample is often reasonable.

Also keep in mind that the most unbiased form of out-of-sample testing is real-time tracking, in which you follow the strategy for some period of time following the optimization and record the performance as new price data are generated in real time. This could either be done in Builder by loading new price data, or you could track the strategy directly in TradeStation.

Another approach that some traders may prefer is to load a different price series into Builder and test the strategy on that file. This can easily be done by adding another market to the table on the Markets tab. After adding the new market, select the strategy to evaluate and select **Evaluate** from the Build menu. For example, you might want to build a strategy for wheat futures then test it on corn futures. Similarly, you could build a strategy on the SPY ETF then test the strategy on various large-cap stocks. Keep in mind, however, that this approach is more demanding of your strategy and may not always produce acceptable results. The suggestions in the previous section can be used to help generate more robust strategies that may have a greater likelihood of performing well on multiple markets.

In addition to out-of-sample testing, the strategies generated by Builder can be optimized in the traditional way; i.e., parameter optimization. As noted above, an input has been added for each parameter in the strategy. These inputs are assigned values during the GP process and are modified by the mutation operator, but in some cases, further improvement may result from optimizing these inputs separately. This can be done in TradeStation using the built-in genetic optimizer of TS 8.5 or later. To set the range for each input, refer to the settings in

the Strategy Options tab in Builder. For example, if an input represents a multiple of the average true range, you can use the Min ATR Multiple and Max ATR Multiple settings. Likewise, if an input is a lookback length for a price pattern, you can use the range 1 to the Max Lookback, Price Patterns setting. As with GP, any results generated from optimizing the strategy inputs should be tested out-of-sample.

The other important issue regarding post-build testing and optimization is monitoring the strategy during real-time trading and re-optimizing or rebuilding if necessary. As discussed in the section on out-of-sample performance, market dynamics can change in such a way that the strategy logic may no longer work. This will tend to happen more frequently for strategies that are optimized over shorter time periods. However, performance can start to decline at any time for any strategy, regardless of its back-tested performance or how long it has been used successfully in real time trading. This is simply one of the risks of trading.

To address this risk, the performance can be monitored over the most recent trade history and compared to the long-term average performance of the strategy. This approach is sometimes referred to as *statistical process control*, which is a method used in manufacturing to make sure the manufacturing processes stay within prescribed error limits. In the context of trading, you might, for example, calculate the percentage of winning trades for the most recent 30 trades and compare this to the percentage for the entire trade history. If the percentage for the most recent period deviates from the long-term average by more than two or three standard deviations, the strategy can be rebuilt or re-optimized. Any performance metric could be used for this calculation, including the average trade, profit factor, and so on.

If a strategy starts to show declining performance, it might be possible to re-optimize the inputs in TradeStation rather than rebuilding in Builder. If that doesn't work, a new strategy can always be built in Builder.

## Tips and Hints

The best way to become proficient at using Adaptrade Builder is to spend some time using it to build strategies. Don't expect that each and every build process will result in a strategy that meets your requirements. It's more likely that after the first run, you'll want to adjust some of the performance weights and build again. Nonetheless, you'll soon find the combination of weights and targets that generates the performance results you want.

The following tips and hints for building strategies with Adaptrade Builder may help you come up with better strategies in less time. Some of these suggestions were also mentioned in previous chapters.

**Good strategy results for small intraday time intervals**, such as 1 min or tick bars, over long time periods, such as several years, can be difficult to find and take multiple days to build. A better approach may be to build over a shorter time period and be prepared to rebuild the strategy every few months to keep up with changing market dynamics.

**The option of removing indicators and/or order types from the build set** is intended to allow users with specific preferences for or against specific strategy elements to restrict the build set to the desired elements. Removing too many items from the build set may reduce the likelihood of finding viable strategies.

**The complexity weight can be used to "break ties" among strategies** that give the same performance results with different numbers of inputs. Setting this weight to a relatively small value can help to keep the entry conditions from getting too verbose.

Whether or not you use the weight for statistical significance, **a significance value of at least 95% is generally desirable**. This implies adequate average trade size, low standard deviation of the trades, and a sufficient number of trades.

**A correlation coefficient of at least 0.9** will help ensure that the performance is fairly uniform over the price history. Lower values often indicate extended flat or down periods.

**If you'll be running the program overnight or while it's unmonitored** for an extended period of time, consider using the "Rebuild on out-of-sample performance" option so that the program will keep working if it doesn't find good results initially.

**The settings in the Build Process category for Crossover Pct, Mutation Pct, Tree Depth, and Tournament Size** can be considered advanced settings. The default values usually suffice. However, don't hesitate to experiment with other values.

**Limiting the number of entries per day** can often improve results for intraday strategies because it reduces the number of different market conditions that must be accommodated by a single strategy.

**To get a feel for how difficult it will be to find viable strategies for your market**, consider setting the number of generations to zero, and save the entire population. The GP process will stop after the initial population is randomly generated. You can then sort the OOS results by net profit to see how many members of the initial population are profitable. The more profitable results you see from these randomly-generated members, the more likely it is that you'll get good results from the build process.

**To maximize the chance of finding viable strategies, it's usually better to start with the most general set of strategy options** and narrow the options only after finding a good solution. For example, start with the symmetry option unchecked (off), so that you can get different entry conditions and order types for long and short trades. Provided that works, try applying the symmetry option to simplify the logic if desired.

**To find the simplest possible strategies**, set the tree depth to zero. This will effectively remove the entry conditions, so that the only entry logic is in the entry orders themselves.

**When choosing the build metrics, keep in mind that some metrics are related to each other.** For example, the average trade size is related to the number of trades. If you set a weight value for the average trade, you'll probably end up with fewer trades. Likewise, both drawdown and correlation coefficient are related to the straightness of the equity curve. Similarly, minimizing the average losing trade could tend to bias the results towards strategies with a small average trade (i.e., both wins and losses are smaller) or towards strategies with a higher profit factor.

**The maximum adverse excursion (MAE)** tells you how far trades go against you before they exit. Minimizing the Ave MAE or Max MAE may result in tighter stops. This can be a good build goal if you have trouble trading strategies that have large open losses, even if the trades usually exit profitably.

**When setting the build goals, it's usually best to start with a small number of metrics**, such as three or four. The default selections are often a good starting point. After a few generations, review the performance results and add or remove metrics, change the weights, and add or remove target values in order to correct any shortcomings you see in the reported results. An incremental approach usually works better than setting the build metrics once and letting the build process run for an extended period of time without intervention.

# Chapter 6
## Menu Commands

## File Menu Commands

The **File** menu of Adaptrade Builder offers the following commands:

| | |
|---|---|
| **New Project** | Creates a new project file. |
| **Open Project** | Opens an existing project file. |
| **Close Project** | Closes an open project file. |
| **Save Project** | Saves an opened project file using the same file name. |
| **Save Project As** | Saves an opened project file to a specified file name. |
| **Print** | Not currently active. |
| **Print Preview** | Not currently active. |
| **Print Setup** | Selects a printer and printer connection. |
| **Exit** | Exits Builder. |

Most of the File menu commands will be inaccessible while the build process is running.

#### New Project Command

Use this command to create a new project file in Builder. The new project will be blank, and all input setting will be set to their default values. If a project file is currently open, you will be prompted to save it if necessary before the new project file is initialized.

You can open an existing project file with the **Open Project** command.

#### Open Project Command

Use this command to open an existing project file. These documents have the file extension .gpstrat; e.g., MyBuilderFile.gpstrat.

You can create new project files with the **New Project** command.

**File Open Window**

The following options allow you to specify which file to open:

**File Name**

Specifies the file you want to open. This field lists files with the extension you select in the pull-down menu to the right of this field.

**Files of Type**

Specifies the type of file you want to open. Builder creates and opens files of type .gpstrat. You can also select “All Files” in this box, but only files of type .gpstrat can be opened.

### Close Project Command

Use this command to close the project file. When you close a project file, Builder prompts you to save the document if it has any unsaved changes. After a project file is closed, the current project is initialized, as if the New Project command had been selected.

### Save Project Command

Use this command to save the active project to its current file name and directory. When you save a document for the first time, Builder displays the **Save As window** so you can name your project file. If you want to change the name and directory of an existing document before you save it, choose the **Save Project As** command.

### Save Project As Command

Use this command to save and name the active project. Builder displays the **Save As window** so you can name your document.

To save a project with its existing name and directory, use the **Save Project** command.

**File Save As Window**

The following options allow you to specify the name and location of the file you are about to save:

**File Name**

Specifies a file name to save a document with a different name. Builder adds the extension you specify under **Save As Type**.

### Print Setup Command

Use this command to select a printer and a printer connection. This command opens a **Print Setup window**, where you specify the printer and its connection.

**Print Setup window**

The following options allow you to select the destination printer and its connection.

**Printer**

Specifies the printer you want to use. Choose the default printer; or choose the **Specific Printer** option and select one of the current installed printers shown in the pane. You install printers and configure ports using Control Panel.

**Orientation**

Specifies **Portrait** or **Landscape**.

**Paper Size**

Specifies the size of paper on which the document is to be printed.

**Paper Source**

Specifies the tray for the paper source, because some printers offer multiple trays for different paper sources.

**Properties**

Displays a window where you can make additional choices about printing, specific to the type of printer you have selected.

**Network**

Click this button to connect to a network location, assigning it a new drive letter.

### 1, 2, 3, . . . Command

Use the numbers and filenames listed at the bottom of the File menu to open the last ten documents you closed. Select the number that corresponds to the document you want to open.

### Exit Command
Use this command to end your Builder session. Builder prompts you to save the open document if it has unsaved changes.

## Edit Menu Commands
The **Edit** menu of Adaptrade Builder offers the following commands:

| | |
|---|---|
| **Undo** | Not currently active. |
| **Cut** | Not currently active. |
| **Copy** | Copies the selected strategy code to the clipboard. |
| **Paste** | Not currently active. |
| **Copy Strategy** | Copy the entire strategy code to the clipboard. |

### Copy Command
Use this command to copy the part of the strategy code that is currently selected in the Strategy Code pane. The selected code will be copied to the clipboard in text format.

### Copy Strategy Command
Use this command to copy the entire strategy code shown in the Strategy Code pane to the clipboard in text format.

## View Menu Commands
The **View** menu offers the following commands:

| | |
|---|---|
| **Toolbars and Docking Windows** | Shows or hides the toolbars and docking windows (panes) and provides customization options. |
| **Status Bar** | Shows or hides the status bar. |
| **Caption Bar** | Shows or hides the caption bar. |
| **Application Look** | Presents optional visual styles for the program. |

### Toolbars and Docking Windows Command
Use this command to show or hide the toolbar, which includes buttons for some of the most common commands in Builder, such as **File Open**, and to show or hide the docking windows (panes) in Builder. A checkmark appears next to the item when it is displayed.

See **Toolbar** below for help on using the toolbar.

### Toolbar
The toolbar is displayed across the top of the application window, below the menu bar. The toolbar provides quick mouse access to File menu functions (**New**, **Open**, **Save**) and Help menu functions (About, Help). Place the mouse cursor over a button (without clicking) to display a brief description of the button. The small, downward-pointing button at the right edge of the toolbar provides options for adding or removing toolbar buttons and for opening the Customize menu (see below).

To hide or display the toolbar, select the Standard item from the **Toolbars and Docking Windows** command from the View menu.

**Customize..**
Use this command to customize the menus, toolbars, and keyboard shortcuts.

### Status Bar Command

Use this command to display and hide the status bar, which describes the action to be executed by the selected menu item or depressed toolbar button. A checkmark appears next to the menu item when the status bar is displayed.

See **Status Bar** below for help on using the status bar.

### Status Bar

The status bar is displayed at the bottom of the main window. To display or hide the status bar, click **Status Bar** from the View menu.

The left area of the status bar describes actions of menu items as you use the arrow keys to navigate through menus. This area similarly shows messages that describe the actions of toolbar buttons as you select them, before releasing them. If after viewing the description of the toolbar button command you decide not to execute the command, then release the mouse button while the pointer is off the toolbar button.

The right areas of the status bar indicate which of the following keys are latched down:

| Indicator | Description |
|---|---|
| CAP | The Caps Lock key is latched down. |
| NUM | The Num Lock key is latched down. |
| SCRL | The Scroll Lock key is latched down. |

### Caption Bar Command

Use this command to show or hide the caption bar. The caption bar is the area below the toolbar containing the Build button. After clicking the Build button, the button changes to "Cancel". Clicking the Cancel button stops the build or evaluation process and changes the button back to "Build".

### Application Look Command

Use this command to change the style of the menus, toolbars, and other visual elements of the user interface of the Builder program. To change the program style, select a style from the popup menu. Styles are available to emulate various Windows programs, such as Office XP, Office 2003, Office 2007, Windows 2000, and Windows XP. Once the style has been changed, it will be stored locally, and the program will continue to use that style until a new style is selected.

## Build Menu Commands

The **Build** menu offers the following commands:

| | |
|---|---|
| **Evaluate** | Evaluates the selected strategy. |
| **Evaluate All** | Evaluates all saved strategies. |
| **Build** | Starts the build process. |

### Evaluate Command

Use this command to evaluate the strategy currently selected in the performance table. The selected strategy will be evaluated over the market data selected on the Markets tab, and the results will be displayed in the performance tables and in the Equity Curve, Trade List, and Strategy Code panes. Use this command to test a strategy over different market data than used in the build or to generate strategy code for a different platform than selected during the build. When this command is selected, the Build button on the caption bar changes to "Cancel", and

a message is displayed in the Output window indicating that the selected strategy is being evaluated.

### Evaluate All Command

Use this command to evaluate all strategies listed in the performance tables. The strategies will be evaluated over the market data selected on the Markets tab, and the results will be displayed in the performance tables and in the Equity Curve, Trade List, and Strategy Code panes. Use this command to test the strategies over different market data than used in the build or to generate strategy code for a different platform than selected during the build. This command can also be used to update the performance results when opening a Builder file from an older version of the program. When this command is selected, the Build button on the caption bar changes to "Cancel", and messages are displayed in the Output window to indicate that the strategies are being evaluated.

### Build Command

The **Build** command is the same as clicking the Build button on the caption bar. Use this command to start the build process. After selecting this command, the Build button changes to "Cancel", and progress messages are displayed in the Output window.

## Help Menu Commands

The **Help** menu offers the following commands, which provide assistance with this application:

| | |
|---|---|
| **Help Topics** | Offers an index to topics on which you can get help. |
| **About Builder** | Displays information about this application. |

### Help Topics Command

Use this command to display the table of contents for Help. From the table of contents, you can open various topics for using Adaptrade Builder, look up topics in the index, and perform word searches using Find. Press F1 at any time to open a help topic related to the selected command or active window. The help system contains the complete contents of this user's guide.

### About Builder Command

Use this command to display information about Adaptrade Builder, such as the copyright notice and version number.

# Appendix: Technical Indicators

Adaptrade Builder currently includes the indicators described below, which are also listed in Table 1 in Chapter 1. Subsequent versions of Builder may include other indictors as well. Most indicators include one or more inputs, such as the averaging length for a moving average. The values of these inputs are chosen by Builder during the build process from the range of values specified on the Strategy Options tab.

### Simple moving average

Simple moving average of price over past the most recent N bars, where N is an input. The simple moving average adds up the price over the past N bars and divides by N.

### Exponential moving average

Exponential moving average (XMA) of price. The XMA is an exponentially weighted average of price. As additional prices are added, their contribution decreases exponentially as (1 - alpha)^N, where alpha = 2/(1 + N). Because more recent prices are weighted more heavily, the XMA is considered to be more responsive to price changes than the simple moving average.

### Moving average convergence divergence (MACD)

Moving average convergence divergence indicator. The MACD is calculated as the difference between two exponential moving averages of price.

### Momentum

Momentum is an oscillator calculated as the difference between the current price and the price N bars ago, where N is an input. Positive values indicate that prices are rising, whereas negative values indicator that prices are falling.

### Rate of change (ROC)

ROC is an oscillator calculated as the ratio of the current price to the price N bars ago, subtracted from 1 and multiplied by 100. Positive values indicate that prices are rising, whereas negative values indicator that prices are falling.

### Fast K stochastic

The Fast K stochastic is a price oscillator scaled to the range 0 to 100. It's calculated as the difference between the bar's close and the lowest low, all divided by the difference between the highest high and the lowest low. High values are generally considered to represent so-called over-bought conditions, whereas low values are generally considered to represent so-called over-sold conditions.

### Fast D stochastic

The Fast D stochastic is a price oscillator scaled to the range 0 to 100. It's calculated as a 3-period exponential moving average of the Fast K stochastic. High values are generally considered to represent so-called over-bought conditions, whereas low values are generally considered to represent so-called over-sold conditions.

### Slow D stochastic

The Slow D stochastic is a price oscillator scaled to the range 0 to 100. It's calculated as a 3-period exponential moving average of the Fast D stochastic, in which the exponential moving average uses a smoothing factor of 1/N, rather than the value 2/(1 + N) of the normal XMA. High values are generally considered to represent so-called over-bought conditions, whereas low values are generally considered to represent so-called over-sold conditions.

### Relative strength indicator (RSI)
The RSI is a momentum oscillator that indicates the strength of the market within a range of 0 to 100. The calculation is based on the ratio of up price changes to down price changes. High values are generally considered to represent so-called over-bought conditions, whereas low values are generally considered to represent so-called over-sold conditions.

### Directional indicator (DI+/DI-)
DI+ is the positive directional indicator, whereas DI- is the negative directional indicator. DI+ indicates the presence of an up trend, whereas DI- indicates the presence of a down trend.

### Directional movement index (DMI)
The DMI indicates the strength of the price trend. The DMI is calculated as the absolute value of the difference between DI+ and DI- divided by the sum of DI+ and DI-, multiplied by 100.

### Average directional index(ADX)
The ADX indicates the strength of the price trend. The ADX is calculated as an exponential moving average of the DMI.

### True range (TR)
The TR is the difference between the true high (highest of the current bar's high and the prior close) and the true low (lowest of the current bar's low and the prior close). The TR is intended to more accurately represent the bar's range than the high minus the low when the prior close is outside of the current bar's range.

### Average true range (ATR)
The ATR is the simple moving average of the true range over the past N bars, where N is an input.

### Lowest
The lowest price over the past N bars, where N is an input. Because the Lowest function returns a price, it is used in Builder where a price value is required, such as in the calculation of stop and target prices.

### Highest
The highest price over the past N bars, where N is an input. Because the Highest function returns a price, it is used in Builder where a price value is required, such as in the calculation of stop and target prices.

### Volume
The volume is the number of shares or contracts traded. For tick data, Builder uses the sum of the up and down tick volumes. Builder uses volume in several ways: the volume on the current bar, the volume N bars ago, the moving average or exponential moving average of volume, and the highest or lowest volume over the past N bars. Any of these volume values may be compared to any other.

### Accumulation/distribution
The Accumulation/distribution line is a volume oscillator, calculated by accumulating a portion of the volume of each bar. The amount of volume added at each bar is equal to the difference between the close and the open divided by the range.

### Chaiken oscillator
The Chaiken oscillator is the difference between two exponential moving averages of the accumulation distribution line.

### Crosses Above/Below
The crosses above indicator returns "true" if the first input is above the second input on the current bar and below it on the prior bar. If the two inputs are equal on the prior bar, the indicator looks back up to the maximum look-back length to determine if the first input was below the second. If not, the

indicator returns “false”. The crosses below indicator works analogously to “crosses above” to return “true” if the first input crosses below the second input on the current bar and returns “false” otherwise. The crosses above/below indicators can be used anywhere that inequality operators (>, <, <=, >=) apply, including with moving averages, stochastics, RSI, MACD, momentum, and so on.

### Price patterns

A price pattern in Builder is defined as the comparison between two prices, where the price can any of the following: simple price (O, H, L, C), look-back price (O[N], H[N], L[N], C[N]), a day price (OpenD(0), HighD(0), LowD(0), CloseD(1)), highest, lowest, simple moving average, or exponential moving average. The comparison between the two prices may be <, <=, >, or >=. Builder may construct any number of price patterns, depending on the tree depth and the other indicators selected as part of the build process.

### Day of week

The day-of-week indicator returns an integer value from 1 to 7 to represent the day of week.

### Time of day

The time-of-day indicator returns the time of the price bar N bars ago, where N is an input. Typically, the time of a price bar is the time at which the bar closes.

### Absolute value

The absolute value function is used when computing the difference between prices. Normally, when using the price difference in a stop or target, the absolute value is applied to ensure that the price difference is a non-negative number. You can exclude this function from the build set to remove this restriction.

# Index